ระบบสร้างกฎแบบจัดเรียงตัว กระจาย และปรับตัวได้

A SELF-ORGANIZED, DISTRIBUTED AND ADAPTIVE RULE-BASED INDUCTION SYSTEM

พรเทพ โรจนวสุ

PORNTHEP ROJANAVASU

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2553

KMITL-2010-EN-D-018-011

ระบบสร้างกฎแบบจัดเรียงตัว กระจาย และปรับตัวได้

A SELF-ORGANIZED, DISTRIBUTED AND ADAPTIVE RULE-BASED INDUCTION SYSTEM

พรเทพ โรจนวสุ

PORNTHEP ROJANAVASU

วิทยานิพนธ์นี้เป็นส่วนหนึ่งตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2553

KMITL-2010-EN-D-018-011

A SELF-ORGANIZED, DISTRIBUTED AND ADAPTIVE RULE-BASED INDUCTION SYSTEM

PORNTHEP ROJANAVASU

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF ENGINEERING IN ELETRICAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2010

KMITL-2010–EN–D–018-011

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ ระบบสร้างกฎแบบจัดเรียงตัว กระจาย และปรับตัวได้

Thesis Title A Self-Organized, Distributed, and Adaptive Rule-Based Induction System

นักศึกษา นายพรเทพ โรจนวสุ

รหัสประจำตัว 48060022

ปริญญา วิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.บุญวัฒน์ อัตชู

หมายเลขวิทยานิพนธ์ KMITL-2010-EN-D-018-011

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.บุญธีร์	เครือตราชู	
ดร.วัชระ	ฉัตรวิริยะ	
ดร.บัณฑิต	วรรธนาภา	
รศ.ดร.ชม	กิ้มปาน	
รศ.ดร.บุญวัฒน์	อัตชู	

วัน / เดือน/ ปี ที่สอบ วันศุกร์ที่ 15 มกราคม พ.ศ. 2553 เวลา 09.30-11.30 น.

สถานที่สอบ ณ อาคาร A ชั้น 3 ห้องประชุม 2

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร.กอบชัย เดชหาญ)

คณบดี คณะวิศวกรรมศาสตร์

วันที่ 15 มกราคม พ.ศ. 2553

หัวข้อวิทยานิพนธ์	ระบบสร้างกฎแบบจัดเรียงตัว กระจาย และปรับตัวได้
ชื่อนักศึกษา	นายพรเทพ โรจนวสุ
รหัสนักศึกษา	48060022
ปริญญา	วิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2553
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร. บุญวัฒน์ อัตชู
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	รศ.ดร. เอื้อน ปิ่นเงิน

บทคัดย่อ

ระบบจำแนกประเภทแบบเรียนรู้ได้ (Learning Classifier System – LCS) เป็นจักรกลเรียนรู้ที่อยู่ในรูปแบบของกฎได้รับความนิยมอย่างมากในด้านการจำแนกประเภทข้อมูลและการเรียนรู้แบบเสริมกำลัง วิทยานิพนธ์ฉบับนี้นำเสนอการประยุกต์ใช้ตัวจำแนกประเภทที่มีชื่อว่า UCS (sUpervised Classifier System) ซึ่งเป็นเวอร์ชั่นหนึ่งของระบบ LCS สำหรับการจำแนกประเภท โดยนำเสนออยู่ในรูปแบบของโครงข่ายงานร่วมกับเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ (Self-Organizing Map –SOM) และโกลว์อิ่งนิวรอนแก๊ส (Growing Neural Gas – GNG) ซึ่งจะทำหน้าที่ในการแตกปัญหาออกเป็นปัญหาย่อย แต่ละปัญหาย่อยที่ถูกแตกจะถูกเรียนรู้โดยตัวจำแนกประเภท UCS ที่เชื่อมต่อกับโหนดของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พหรือโกลว์อิ่งนิวรอนแก๊สแยกจากกัน เราได้ทดสอบโครงข่ายงานที่นำเสนอนี้โดยการเปลี่ยนตัวจำแนกประเภท UCS เป็นโครงข่ายประสาทเทียม (Artificial Neural Network – ANN)

เราได้ทดสอบโครงข่ายงานที่นำเสนอกับชุดข้อมูลสังเคราะห์และชุดข้อมูลจริงหลายชุดข้อมูลรวมถึงชุดข้อมูลขนาดใหญ่ด้วย โครงข่ายงานที่นำเสนอสามารถให้ค่าความถูกต้องในการจำแนกประเภทข้อมูลได้เทียบเท่าหรือดีกว่าการใช้ตัวจำแนกประเภทแบบตัวเดียว นอกจากนั้นยังใช้เวลาในการเรียนรู้ชุดข้อมูลได้เร็วกว่า เนื่องจากตัวจำแนกประเภท UCS ที่ผูกติดกับโหนดของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พหรือโกลว์อิ่งนิวรอนแก๊สนั้นจะมีจำนวนประชากรกฎน้อยกว่าการใช้งานตัวจำแนกประเภท UCS แบบตัวเดียวมาก ดังนั้นเมื่อมีอินพุตเข้ามายังโครงข่ายงานที่นำเสนอก็จะถูกเปรียบเทียบกับกฎที่มีจำนวนน้อยกว่า ดังนั้นทำให้เราสามารถประมวลผลข้อมูลในช่วงเวลาที่เท่ากันได้มากกว่าด้วย ผลการทดลองแสดงให้เห็นว่าโครงข่ายงานที่นำเสนอสามารถแตกปัญหาและยังให้ค่าความถูกต้องที่เทียบเท่าหรือดีกว่าโดยใช้เวลาเรียนรู้ที่น้อยกว่า

Thesis Title	A Self-Organized, Distributed and Adaptive Rule-Based Induction System
Student	Mr. Pornthep Rojanavasu
Student ID.	48060022
Degree	Doctor of Engineering
Programme	Electrical engineering
Year	2010
Thesis Advisor	Assoc. Prof. Dr. Boonwat Attachoo
Thesis Co-advisor	Assoc. Prof. Dr. Ouen Pinngern

ABSTRACT

Learning classifier systems (LCSs) are rule-based inductive learning systems that have been widely used in the field of supervised and reinforcement learning over the last few years. This paper employs sUpervised Classifier System (UCS), a supervised learning classifier system, which was introduced in 2003 for classification tasks in data mining. We present an adaptive framework of UCS on top of a self-organized map (SOM) neural network and a growing neural gas (GNG). The overall classification problem is decomposed adaptively and in real time by the SOM/GNG into subproblems, each of which is handled by a separate UCS. The framework is also tested with replacing UCS by a feed-forward artificial neural network (ANN).

Experiments on several synthetic and real data sets, including a very large real data set, show that the accuracy of classifications in the proposed distributed environment is as good or better than in the nondistributed environment, and execution is faster. In general, each UCS attached to a cell in the SOM/GNG has a much smaller population size than a single UCS working on the overall problem; since each data instance is exposed to a smaller population size than in the single population approach, the throughput of the overall system increases. The experiments show that the proposed framework can decompose a problem adaptively into subproblems, maintaining or improving accuracy and increasing speed.

กิตติกรรมประกาศ

วิทยานิพนธ์นี้จะไม่สำเร็จลุล่วงหากปราศจากความช่วยเหลือและการสนับสนุนของบุคคลอีกหลาย ๆ คน ข้าพเจ้าขออนุญาตแสดงความขอบคุณดังต่อไปนี้

ลำดับแรกคือ รศ.ดร.บุญวัฒน์ อัตชู ผู้ควบคุมวิทยานิพนธ์ รศ.ดร. เอื้อน ปิ่นเงิน ผู้ควบคุมวิทยานิพนธ์ร่วม Prof.Dr. Hussein A. Abbass และ Dr. Hai Huong Dam อาจารย์ที่ปรึกษาในขณะที่ข้าพเจ้าทำวิจัย ณ. กรุงแคนเบอร์ล่า ประเทศออสเตรเลีย ท่านเหล่านี้เป็นผู้ให้คำปรึกษา และแนวทางการแก้ปัญหา นอกเหนือจากนั้นยังให้คำแนะนำ ข้อคิดในการทำงานและการดำรงชีวิตที่ตั้งมั่นอยู่บนความวิริยอุตสาหะและความใฝ่รู้ ข้าพเจ้าขอกราบขอบพระคุณเป็นอย่างสูง

ข้าพเจ้าขอกราบเท้าครูบาอาจารย์ที่ได้ประสิทธิ์ประสาทวิชาความรู้ให้แก่ข้าพเจ้าทุกท่านตั้งแต่เล็กจนเติบใหญ่ รวมทั้งสั่งสอนและอบรมให้ข้าพเจ้าเป็นคนดี ข้าพเจ้าขอกราบขอบพระคุณเป็นอย่างสูง

ข้าพเจ้าขอขอบคุณสำหรับกำลังใจ คำแนะนำ และประสบการณ์ที่ดีจาก คุณป้า คุณอา และน้อง ๆ นักเรียนไทยในกรุงแคนเบอร์ล่า ประเทศออสเตรเลีย คุณป้าบุญศรี คุณอาสถาพร Nguyen M.H. น้องเหวย พี่หนิง น้องแกมและน้องก้าว รวมถึงพี่ ๆ เพื่อน ๆ รวมถึงน้อง ๆ นักศึกษาปริญญาเอกและนักศึกษาปริญญาโท ห้องวิจัยศาสตร์ข้อมูลสำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ (ReCCIT) และห้องวิจัยนักศึกษาปริญญาโทและเอกภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่าน พี่สมบัติ ไพฑูลย์ น้องป้อม Thach พี่น้อย น้องซัน น้องนพ และโดยเฉพาะอย่างยิ่ง พี่ต้อง เกรียงศักดิ์ เตมีย์ สำหรับคำแนะนำที่ให้รู้จักกับระบบจำแนกประเภท LCS สามปีที่ลาดกระบังและหนึ่งปีที่ประเทศออสเตรเลียข้าพเจ้าจะไม่มีวันลืม

ลำดับสุดท้ายคือครอบครัวของข้าพเจ้าที่ให้กำลังใจเสมอมา คุณพ่อ คุณแม่ พี่ชาย คุณตา และคุณยายผู้ล่วงลับ ผู้ที่เลี้ยงดู ส่งเสีย อบรม หล่อหลอม ขัดเกลาข้าพเจ้าจนเติบใหญ่ ข้าพเจ้าขอกราบแทบเท้าและขอสำนึกในพระคุณนี้อย่างเป็นที่สุดจนกว่าชีวิตจะหาไม่ คุณสุภารัฐ โรจนวสุ ภริยาสุดที่รักที่คอยผลักดัน กระตุ้นและเป็นแรงใจให้ข้าพเจ้าฝ่าฟันต่ออุปสรรคต่าง ๆ นานา รวมถึงสมาชิกครอบครัวปางวัชรากรทุกท่านที่ร่วมเป็นกำลังใจให้ข้าพเจ้า

สุดท้ายนี้คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับผู้มีพระคุณทุกท่าน หากวิทยานิพนธ์ฉบับนี้มีข้อผิดพลาดประการใดข้าพเจ้าขอน้อมรับไว้เพียงผู้เดียว

พรเทพ โรจนวสุ

สารบัญ

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ระบบจำแนกประเภทแบบเรียนรู้ได้ (Learning Classifier System – LCS) เป็นจักรกลเรียนรู้ที่อยู่ในรูปแบบของกฎ ระบบจำแนกประเภท LCS ถูกออกแบบจากการรวมกันระหว่างการเรียนรู้แบบเสริมกำลัง (Reinforcement learning) และ เจเนติกอัลกอริธึม (Genetic Algorithm) ส่วนของการเรียนรู้แบบเสริมกำลังนั้นจะทำหน้าที่ในการปรับค่าพารามิเตอร์ของกฎโดยอาศัยค่าตอบแทน (Reward) ที่ได้มาจากสภาพแวดล้อม (Environment) ในส่วนของเจเนติกอัลกอริธึมนั้นจะทำหน้าที่ในการสร้างกฎใหม่โดยอาศัยกระบวนวิวัฒนาการต่าง ๆ ข้อดีของระบบจำแนกประเภท LCS คือ ประการแรกความรู้ที่ได้จะอยู่ในรูปแบบของกฎที่เราสามารถทำความเข้าใจได้ ประการที่สองระบบจำแนกประเภท LCS เป็นการเรียนรู้แบบออนไลน์กล่าวคือระบบจะทำการปรับปรุงค่าพารามิเตอร์ต่าง ๆ ทันทีเมื่อมีอินพุตผ่านเข้ามา ประการสุดท้ายระบบจำแนกประเภท LCS อาศัยเจเนติกอัลกอริธึมเป็นเครื่องมือในการค้นหาและสร้างกฎใหม่ให้ครอบคลุมขอบเขตของปัญหาซึ่งทำให้ระบบนั้นมีความทนทาน (Robustness) ต่ออินพุตที่เข้ามา

ในปัจจุบันข้อมูลมีปริมาณที่เพิ่มขึ้นอย่างรวดเร็ว การทำเหมืองข้อมูล (Data mining) โดยอาศัยจักรกลเรียนรู้จำเป็นที่จะต้องปรับให้เหมาะกับการเจริญเติบโตของข้อมูลด้วย ในการใช้งานระบบจำแนกประเภท LCS กับข้อมูลที่มีขนาดใหญ่ โดยธรรมชาติของระบบจำแนกประเภทแบบกฎจะสร้างกฎออกมาเป็นจำนวนมาก โดยเฉพาะเมื่อข้อมูลมีความไม่เป็นเชิงเส้น (Non-linear problem) เนื่องจากกฎจะอยู่ในรูปแบบของระนาบเกิน (Hyper-plane) จำนวนกฎที่เพิ่มขึ้นทำให้ระบบต้องใช้เวลาในการประมวลผลข้อมูลเพิ่มขึ้นด้วย เนื่องจากในแต่ละครั้งที่มีข้อมูลเข้ามาระบบจะต้องทำการค้นหากฎทั้งหมดเพื่อหากฎที่สอดคล้อง

ในงานวิจัยนี้ได้สร้างโครงข่ายงาน (Framework) เพื่อแตกปัญหาก่อนที่จะส่งต่อไปยังตัวจำแนกประเภท กระบวนการแตกปัญหาจะเปรียบเสมือนกับตัวพรีเกท (Pre-gate) ซึ่งข้อมูลจะผ่านกระบวนการแตกปัญหาก่อนเข้าไปสู่กระบวนการจำแนกประเภท ในงานวิทยานิพนธ์ฉบับนี้อาศัยกระบวนการจัดกลุ่มข้อมูลเป็นตัวพรีเกทเพื่อใช้ในการแตกปัญหา ดังนั้นข้อมูลที่เข้ามาจะถูกจัดกลุ่มก่อนส่งต่อไปยังตัวจำแนกประเภท ในวิทยานิพนธ์ฉบับนี้จะใช้แผนภาพเซลฟ์ออร์แกนไนท์ซิ่ง (Self-Organizing Map - SOM) และโกลว์อิ่งนิวรอนแก๊ส (Growing Neuron Gas –GNG) ในส่วนของกระบวนการจำแนกประเภทเราจะใช้ตัวจำแนกประเภท UCS ซึ่งเป็นตัวจำแนกประเภท LCS ประเภทหนึ่ง และตัวจำแนกประเภทนิวรอนเน็ตเวิร์ก(Artificial Neural Network – ANN)

1.2 ความมุ่งหมายและวัตถุประสงค์ของการวิจัย

1. เพื่อศึกษา พัฒนา การใช้งานระบบจำแนกประเภท LCS กับข้อมูลที่มีขนาดใหญ่ ทั้งในเชิงของประสิทธิภาพในการจำแนกประเภท และเวลาที่ใช้ในการเรียนรู้ข้อมูล
2. เพื่อออกแบบโครงข่ายงานเชื่อมต่อระหว่างกระบวนการแตกปัญหาโดยอาศัยการจัดกลุ่มข้อมูลและกระบวนการเรียนรู้ของตัวจำแนกประเภท LCS
3. เพื่อวิเคราะห์และประเมินประสิทธิภาพในการแตกปัญหา ประสิทธิภาพในการจำแนกประเภท รวมถึงความเร็วในการเรียนรู้ของโครงข่ายงานที่นำเสนอ

1.3 สมมติฐานของการศึกษา

โดยปกติข้อมูลมีที่ปริมาณมากความซับซ้อนในการเรียนรู้ข้อมูลจะเพิ่มตามขึ้นด้วย ตามธรรมชาติของการทำงานใด ๆ เมื่องานนั้นมีความซับซ้อนหรือมากจนเกินกำลังของคนเดียวที่จะสามารถจัดการได้ โดยทั่วไปเราก็จะพยายามแตกงานนั้นเป็นงานย่อยหลาย ๆ งานแบ่งให้คนหลายคนรับผิดชอบแต่ละงานแตกต่างกัน แนวคิดนี้ในการทำเหมืองข้อมูลเรียกว่าการแตกปัญหาใหญ่ให้เป็นปัญหาย่อย (Problem decomposition) คือการแตกปัญหาใหญ่ที่มีความซับซ้อนยากเกินกว่าที่ตัวจำแนกประเภทพียงตัวเดียวเรียนรู้ได้ให้เป็นปัญหาย่อยที่มีความซับซ้อนน้อยกว่าและสามารถจัดการ โดยอาศัยตัวจำแนกประเภทที่มีความซับซ้อนน้อยกว่าหลาย ๆ ตัวช่วยกัน เพื่อลดความซับซ้อนของจักรกล รวมถึงเพิ่มความเร็วในการเรียนรู้ของตัวจำแนกประเภท

1.4 ขอบเขตการวิจัย

ในงานวิจัยนี้ทำการศึกษาและวิจัยการแตกปัญหาโดยพัฒนาเป็นโครงข่ายงานสำหรับการจำแนกประเภท โดยอาศัยเทคนิคการจัดกลุ่มข้อมูลของแผนภาพเซลฟ์ออร์แกนไนท์ซิ่งและโกลว์อิ่งนิวรอนแก๊ส ตัวจำแนกประเภทที่ใช้งานคือตัวจำแนกประเภท LCS ซึ่งเป็นตัวจำแนกประเภทหลักและตัวจำแนกประเภท ANN เป็นตัวจำแนกประเภทเพื่อทดสอบโครงข่ายงาน ในการวัดประสิทธิภาพของโครงข่ายงานจะวัดจากประสิทธิภาพในการจำแนกประเภทและความเร็วในการเรียนรู้ข้อมูลเทียบกับการใช้ตัวจำแนกประเภทเพียงตัวเดียว

1.5 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

เพื่อให้เป็นไปตามวัตถุประสงค์ของการออกแบบโครงข่ายงานที่นำเสนอ จะต้องทำความเข้าใจในหลักการและทฤษฎีดังต่อไปนี้

1. หลักการเบื้องต้นของการทำเหมืองข้อมูล รวมถึงพื้นฐานการจำแนกประเภท และการจัดกลุ่มข้อมูล
2. ทฤษฎีระบบจำแนกประเภทแบบเรียนรู้ได้ LCS และนิวรอนเน็ตเวิร์ก
3. ทฤษฎีการจัดกลุ่มข้อมูลแบบแผนภาพเซลฟ์ออร์แกนไนท์ซิ่งและโกลว์อิ่งนิวรอนแก๊ส

4. วิธีทางสถิติที่ใช้ในการวัดประสิทธิภาพในการจำแนกประเภทข้อมูล

1.6 ขั้นตอนการวิจัย

1. ศึกษาทฤษฎีเบื้องต้น งานวิจัยที่เกี่ยวข้องในการพัฒนาระบบจำแนกประเภท LCS กับฐานข้อมูลขนาดใหญ่
2. กำหนดวัตถุประสงค์และขอบเขตของงานวิจัย พร้อมทั้งตั้งสมมติฐานการวิจัยเบื้องต้น
3. ออกแบบโครงข่ายงานในการเชื่อมต่อกันของอัลกอริธึมในการจัดกลุ่มและอัลกอริธึมในการจำแนกประเภท
4. พัฒนาโปรแกรมตามโครงข่ายงานที่ได้ออกแบบไว้ พร้อมทั้งออกแบบการทดลองสร้าง และรวบรวมข้อมูลที่ใช้ในการทดลอง
5. ทดลองและบันทึกผลการทดลอง พร้อมทั้งวิเคราะห์ผลการทดลองเบื้องต้นเพื่อปรับปรุงโครงข่ายที่ออกแบบไว้
6. วิเคราะห์ผลการทดลอง และสรุปผลการทดลอง นำเสนอผลงานวิจัยในงานประชุมวิชาการนานาชาติรวมถึงวารสารนานาชาติ ตามเงื่อนไขที่บัณฑิตวิทยาลัยกำหนดไว้

1.7 เครื่องมือและอุปกรณ์ที่ใช้ในการวิจัย

เครื่องมือและอุปกรณ์ที่ใช้ในการวิจัย ได้แก่

1. เครื่องคอมพิวเตอร์ตั้งโต๊ะและส่วนบุคคล
2. เครื่องคอมพิวเตอร์คลัสเตอร์ Barossa (barossa.ac3.com.au)
3. ระบบปฏิบัติการWindows XP SP3
4. ระบบปฏิบัติการ UNIX
5. ซอฟต์แวร์ MATLAB เวอร์ชั่น R2008a
6. ซอฟต์แวร์ Microsoft Visual Studio 2005
7. คอมไพเลอร์ GCC รันบน UNIX

1.8 รายละเอียดในแต่ละบท

วิทยานิพนธ์ฉบับนี้แบ่งออกเป็น 7 บท แต่ละบทประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมา และความสำคัญของปัญหา ความมุ่งหมาย และวัตถุประสงค์ของการศึกษา สมมุติฐานของการศึกษา ทฤษฎีหรือแนวคิดที่ใช้ในการศึกษา ขอบเขตของการศึกษา และขั้นตอนของการศึกษา และเครื่องมือและอุปกรณ์ที่ใช้ในการวิจัย

บทที่ 2 กล่าวถึงพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย โดยกล่าวถึง การทำเหมืองข้อมูล พื้นฐานการจำแนกประเภท การจัดกลุ่มข้อมูล รวมทั้งวิธีการวัดประสิทธิภาพ

บทที่ 3 กล่าวถึงการสำรวจบทความต่าง ๆ ที่เกี่ยวข้องกับการวิจัยเพื่อเป็นแนวทางในการวิจัย รวมทั้งแนวคิดในการทำวิทยานิพนธ์ฉบับนี้

บทที่ 4 กล่าวถึงอัลกอริธึมต่าง ๆ ที่ใช้ในงานวิจัย คือ ตัวจำแนกประเภทแบบเรียนรู้ได้ LCS ประเภทต่าง ๆ และนิวรอนเน็ตเวิร์ก แผนภาพเซลฟ์ออร์แกนไนท์ซิ่ง และโกลว์อิ่งนิวรอนแก๊ส

บทที่ 5 โครงข่ายที่นำเสนอ ผลการทดลอง และการวิเคราะห์ผลการทดลอง

บทที่ 6 สรุปผล ข้อเสนอแนะ และแนวทางในการพัฒนาต่อไปในอนาคต

บทที่ 2

ความรู้พื้นฐานที่เกี่ยวข้องกับงานวิจัย

2.1 บทนำ

ปัจจุบันจำนวนข้อมูลที่เก็บอยู่ในแต่ละองค์กรมีแนวโน้มที่จะเพิ่มขึ้นไปอย่างไม่มีที่สิ้นสุด สืบเนื่องมาจากการพัฒนาเทคโนโลยีในด้านต่าง ๆ เช่น การพัฒนาเทคโนโลยีของซอฟต์แวร์ที่ทำให้องค์กรต่าง ๆ สามารถสร้างโปรแกรมเพื่อใช้ในการเก็บข้อมูลตามที่ความต้องการได้อย่างง่ายขึ้น การพัฒนาเทคโนโลยีความจุของฮาร์ทดิสก์ในปัจจุบันสามารถเก็บข้อมูลได้ในหน่วยเทราไบต์ (1 เทราไบต์ = 10^{12} ไบต์) ทำให้องค์กรต่าง ๆ สามารถหาซื้อเพิ่มมาเก็บข้อมูลได้ง่าย อีกทั้งการพัฒนาเทคโนโลยีของอุปกรณ์เซนเซอร์ที่สามารถที่จะบันทึกข้อมูลในการตัดสินใจต่าง ๆ การบันทึกการเปลี่ยนแปลงของอุณหภูมิ รวมถึงการเก็บพิกัดทางภูมิศาสตร์ และอุปกรณ์เซนเซอร์อื่น ๆ อีกมากมาย ทำให้องค์กรต่าง ๆ ตกอยู่ในภาวะที่เรียกว่า "ปริมาณข้อมูลมีมหาศาลแต่ขาดองค์ความรู้และความเข้าใจเกี่ยวกับข้อมูล"

ในบทนี้จะกล่าวถึงภาพรวมของการดึงเอาความรู้ออกมาจากข้อมูลหรือเรียกว่าการทำเหมืองข้อมูล (Data mining) โดยหัวข้อ 2.2 กล่าวถึงความหมายและพื้นฐานในการทำเหมืองข้อมูล หัวข้อ 2.3 กล่าวถึงหลักการการจำแนกประเภทข้อมูล และรูปแบบต่าง ๆ ในการแทนความรู้ของโมเดลหลังจากการเรียนรู้แล้ว หัวข้อ 2.4 กล่าวถึงหลักการจัดกลุ่มข้อมูลและประเภทของการจัดกลุ่มข้อมูล

2.2 การทำเหมืองข้อมูล

การทำเหมืองข้อมูลได้รับความสนใจเป็นอย่างมากในช่วงสองทศวรรษที่ผ่านมาทั้งในแวดวงงานวิจัย แวดวงการสารสนเทศ หรือแม้กระทั่งแวดวงอุตสาหกรรมต่าง ๆ เนื่องมาจากในปัจจุบันมีการพัฒนาเครื่องมือการทำเหมืองข้อมูลที่มีประสิทธิภาพ สะดวก และมีเทคนิคต่าง ๆ เพื่อให้เหมาะสมกับข้อมูลที่มีอยู่

2.2.1 ความหมายและขอบเขตการทำเหมืองข้อมูล

ในหนังสือหรืองานวิจัยต่าง ๆ ได้นิยามความหมายของการทำเหมืองข้อมูลไว้อยู่หลายแบบ ซึ่งสามารถรวบรวมนำมาเสนอได้ดังนี้

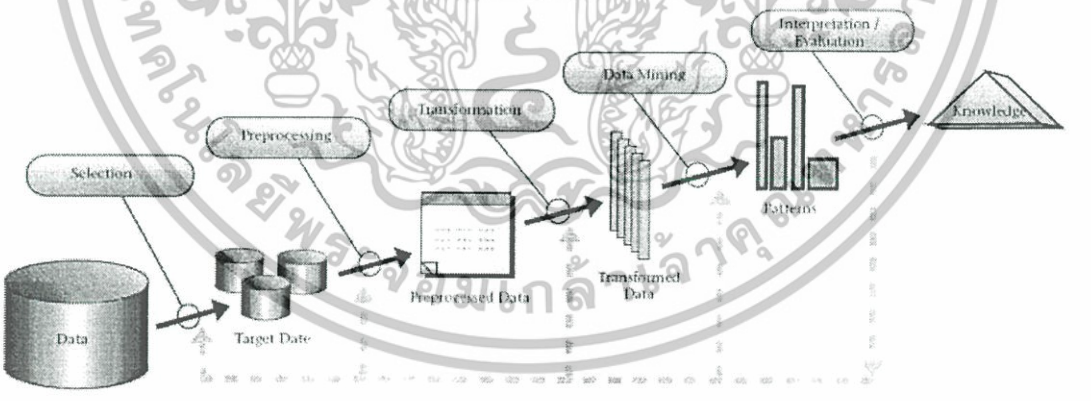
- Frawley et al. [1] นิยามไว้ว่า การทำเหมืองข้อมูล (Data mining - DM) หรือการค้นหาความรู้จากฐานข้อมูล (Knowledge Discovery in Database – KDD) คือ การดึงเอาสารสนเทศ (Information) ที่ซ่อนอยู่ สารสนเทศไม่เป็นที่รู้จัก และเป็นสารสนเทศที่เป็น

ประโยชน์ ออกจากข้อมูลที่มีอยู่โดยอาศัยเทคนิคต่าง ๆ เช่นการจัดกลุ่มข้อมูล (Clustering) การจำแนกประเภท (Classification) การหาความโครงข่ายความสัมพันธ์ (Dependency networks) การวิเคราะห์การเปลี่ยนแปลง (Analyzing changes) และการตรวจจับสิ่งแปลกปลอมต่าง ๆ (Detecting anomalies)

- Fayyad et al. [2] นิยามไว้ว่า การทำเหมืองข้อมูลเป็นกระบวนการหนึ่งในกระบวนการค้นหาความรู้จากฐานข้อมูลโดยอาศัยอัลกอริธึม ซึ่งอัลกอริธึมเหล่านั้นสามารถค้นหารูปแบบ (Pattern) ที่อยู่ภายในข้อมูลโดยใช้เวลาในการคำนวณที่ยอมรับได้
- Hand et al. [3] นิยามไว้ว่า การทำเหมืองข้อมูลเป็นการวิเคราะห์ชุดข้อมูลเพื่อหาความสัมพันธ์บางอย่างที่ซ่อนไว้ สรุปและนำเสนอด้วยวิธีการใหม่ ๆ ซึ่งเพิ่มความเข้าใจและเป็นประโยชน์ต่อผู้ใช้งาน

โดยส่วนใหญ่แล้วความหมายของการทำเหมืองข้อมูลจะหมายถึงกระบวนการหนึ่งในกระบวนการค้นหาความรู้จากฐานข้อมูล ซึ่งจะเป็นกระบวนการค้นหารูปแบบที่เป็นประโยชน์และนำเสนอในรูปแบบที่มนุษย์สามารถทำความเข้าใจได้ รูปแบบเหล่านี้ส่วนใหญ่จะเป็นความสัมพันธ์ระหว่างข้อมูล หรือเป็นการบรรยายลักษณะของข้อมูลด้วยวิธีการใดวิธีการหนึ่ง

โดยทั่วไปแล้วกระบวนการค้นหาความรู้จากฐานข้อมูลเริ่มต้นจากข้อมูลดิบ ข้อมูลดิบคือข้อมูลที่บันทึกอยู่ในรูปแบบของเรคอร์ดทั่วไป เช่น ข้อมูลเกี่ยวกับการเงิน ข้อมูลเกี่ยวกับทางการแพทย์ ข้อมูลเพ็คเกจที่วิ่งในระบบเนตเวิร์ก ผลลัพธ์ที่ได้จากกระบวนการนี้จะเป็นรูปแบบที่เป็นประโยชน์ที่ซ่อนอยู่ในฐานข้อมูลนั้น



รูปที่ 2.1 กระบวนการค้นหาความรู้จากฐานข้อมูล [1]

รูปที่ 2.1 แสดงกระบวนการต่าง ๆ ในการค้นหาความรู้จากฐานข้อมูล จากรูปเราสามารถที่จะแบ่งกระบวนการออกเป็น 5 กระบวนการย่อย

[1] ที่มา http://www.infovis-wiki.net/images/4/4d/Fayyad96kdd-process.png

- การคัดเลือกข้อมูล (Data selection) ขั้นตอนนี้ถือเป็นขั้นตอนเริ่มต้น โดยเราจะต้องทำความเข้าใจจุดประสงค์ของผู้ใช้ (End-user) เพื่อที่จะได้ทำการเก็บข้อมูลดิบได้อย่างถูกต้องตรงกับความต้องการของผู้ใช้ ซึ่งการคัดเลือกข้อมูลที่เหมาะสมจะส่งผลให้สามารถลดเวลาการสกัดความรู้ และทำให้ความรู้ที่ได้มีประโยชน์อย่างแท้จริง
- กระบวนการเตรียมข้อมูลก่อนการประมวลผล (Data preprocessing/Data cleaning) คือ กระบวนการเตรียมข้อมูลด้วยวิธีการต่าง ๆ อาทิเช่น การกำจัดข้อมูลแปลกปลอม (filter) ข้อมูลที่ขัดแย้งกันเอง กระบวนการแปลงรูปแบบของข้อมูลให้เป็นรูปแบบเดียวกัน หรือแม้กระทั้งการหาวิธีจัดการกับข้อมูลที่มีคุณลักษณะบางอย่างขาดหายไป
- กระบวนการแปลงรูปข้อมูล (Data transformation) คือการแปลงข้อมูลไปยังรูปแบบอื่น ๆ เช่น การลดคุณลักษณะของข้อมูล (Feature reduction) หรือแม้กระทั้งการแปลงรูปคุณลักษณะข้อมูลไปยังโดเมนอื่น (Feature transformation) ทั้งนี้เพื่อให้เพิ่มความเข้าใจในข้อมูลนั้น
- กระบวนการทำเหมืองข้อมูล (Data mining) คือการค้นหารูปแบบที่ซ่อนอยู่ในข้อมูล โดยอาศัยอัลกอริธึมต่าง ๆ
- กระบวนการประเมินผลและการแปลความหมาย (Data Evaluation and interpretation) คือ การประเมินผลรูปแบบที่ได้มาเพื่อให้ได้รูปแบบที่เราสนใจจริง ๆ โดยจะอาศัยเทคนิคในการประเมินแบบใดแบบหนึ่ง หลังจากนั้นจะทำการจัดเตรียมรูปแบบที่ได้มาให้อยู่ในรูปแบบที่สามารถเข้าใจได้ง่าย

2.2.2 เหตุผลในการทำเหมืองข้อมูล

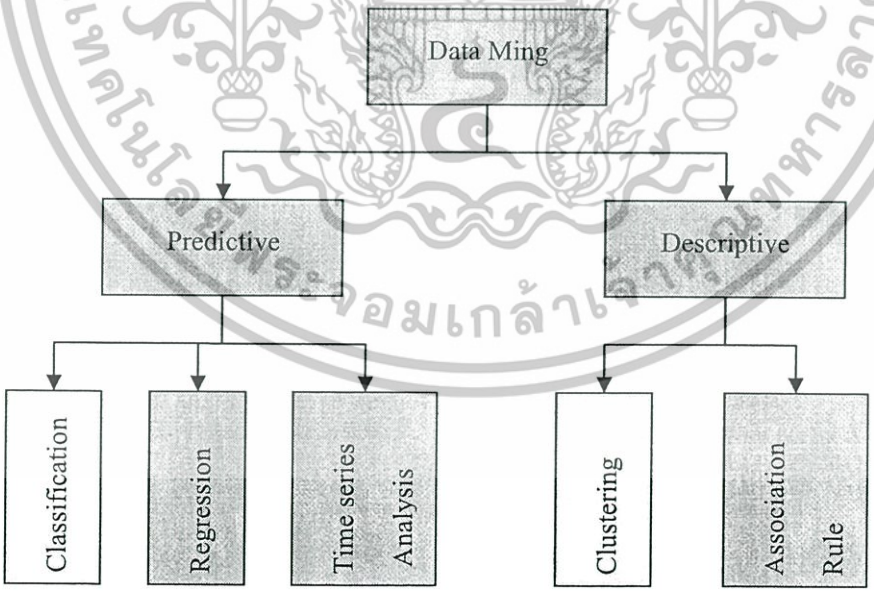
Han [4] กล่าวถึงเหตุผลของการทำเหมืองข้อมูลไว้สองประการ ประการแรกคือการเพิ่มขึ้นของข้อมูลอย่างมหาศาล ประการที่สองคือเราต้องการที่จะแปลงข้อมูลมหาศาลเหล่านั้นออกมาเป็นองค์ความรู้ที่สามารถใช้งานได้

ในราว ๆ สองทศวรรษมีการพัฒนาเทคโนโลยีต่าง ๆ ทั้งด้านซอฟต์แวร์ ฮาร์ดแวร์ และที่สำคัญเทคโนโลยีอินเตอร์เน็ต ทำให้องค์กรต่าง ๆ เริ่มหันมาให้ความสนใจกับข้อมูลเพื่อเพิ่มประสิทธิภาพในการแข่งขัน มีการบันทึกข้อมูลในทุกรายการที่ทำงาน ยกตัวอย่างเช่น บริษัทสื่อสารโทรคมนาคมทำการบันทึกรายการต่างๆ ที่ลูกค้าเข้ามาติดต่อใช้บริการ หรือ บริษัทอินเตอร์เน็ตมีการบันทึกข้อมูลการเข้าชมเว็บไซต์ และกรณีอื่น ๆ อีกมาก ทำให้ข้อมูลที่เก็บมีปริมาณมหาศาล ในกรณีของบริษัทใหญ่ ๆ ข้อมูลเหล่านี้อาจจะมีปริมาณมากเป็นเพทราไบต์ (1 Petabyte = 10^{15}) หรือเอ็กซ์ซาไบต์(1 Exabyte = 10^{18}) ข้อมูลที่มีปริมาณมากขนาดนี้หากเราสามารถที่จะดึงเอาความรู้ออกได้จะสามารถช่วยในการวางแผน การทำตลาด หรือแม้กระทั่ง การตัดสินใจของผู้บริหารระดับสูงเพื่อกำหนดทิศทางของบริษัทในอนาคต

ถึงแม้ว่าการค้นหาความรู้จากฐานข้อมูลอาจจะไม่ใช่เรื่องใหม่มากนัก ในยุคแรกการดึงความรู้จะอาศัยมนุษย์ในการวิเคราะห์และทำความเข้าใจ มนุษย์ก็จะอาศัยโปรแกรมสำเร็จรูปเช่น SPSS หรือ Microsoft Excel ช่วย ซึ่งเมื่อข้อมูลมีความซับซ้อนมากขึ้นมนุษย์ก็จะไม่สามารถที่จะวิเคราะห์ได้ ในบางครั้งเครื่องมือที่ใช้ในการวิเคราะห์ก็ไม่อาจจะสนับสนุนกับฐานข้อมูลที่มีขนาดใหญ่ หรือไม่สามารถวิเคราะห์ในสิ่งที่เราต้องการได้จริง ๆ ในปี ค.ศ. 1989 มีการจัดเวิร์คช็อบขึ้นเพื่อตอบตอบสนองความต้องการในการค้นหาความรู้จากฐานข้อมูลที่มีความซับซ้อนและมีขนาดใหญ่ ในหัวข้อ การค้นหาความรู้จากฐานข้อมูลขนาดใหญ่ หรือรู้จักกันดีในนามของ Knowledge Discovery in Large Databases (KDD) นับตั้งจุดนี้เองงานวิจัยในด้านนี้จึงได้รับความสนใจเป็นต้นมา

2.2.3 ประเภทและวิธีการทำเหมืองข้อมูล

ในการทำเหมืองข้อมูลมีการจัดประเภทไว้สองประเภทหลัก ๆ คือ การบรรยายข้อมูล (Description) คือค้นหารูปแบบที่น่าสนใจภายในจากฐานข้อมูลและบรรยายออกมาอยู่ในรูปแบบใดรูปแบบหนึ่ง ตัวอย่างของรูปแบบที่น่าสนใจ เช่นรูปแบบพฤติกรรมของลูกค้าที่มีพฤติกรรมในการซื้อสินค้าคล้ายๆกัน การทำนายข้อมูล (Prediction) คือการทำนายข้อมูลที่เข้ามาใหม่โดยอาศัยฐานข้อมูลเก่าที่มีอยู่ ส่วนใหญ่ในการทำนายข้อมูลเราจะต้องทำการสร้างโมเดลที่เหมาะสมกับข้อมูลนั้น โดยสามารถสร้างได้จากฐานข้อมูลที่มีอยู่



รูปที่ 2.2 แผนภาพประเภทและงานในการทำเหมืองข้อมูลแบบต่าง ๆ อย่างย่อ

รูปที่ 2.2 แสดงประเภทและตัวอย่างงานในด้านการทำเหมืองข้อมูลของแต่ละประเภท โดยนำเสนอเฉพาะตัวที่ได้รับความนิยมและจะกล่าวถึงในวิทยานิพนธ์นี้คือ

- การจำแนกประเภท (Classification) จุดประสงค์ของการจำแนกประเภทคือ การสร้างโมเดลที่สามารถบ่งบอกประเภทของข้อมูลที่เข้ามาใหม่ได้อย่างถูกต้องมากที่สุด โดยโมเดลนั้นจะถูกสร้างจากข้อมูลประเภทเดียวกันในอดีตที่เก็บไว้ซึ่งเรียกว่าข้อมูลที่ใช้ในการเรียนรู้ (Training dataset) เช่น สร้างโมเดลทำนายชนิดของนกที่พบเห็น จากข้อมูลลักษณะของนกที่เก็บไว้
- การจัดกลุ่มข้อมูล (Clustering) จุดประสงค์เพื่อพยายามรวมกลุ่มข้อมูลตัวอย่างที่มีคุณลักษณะบางอย่างคล้ายกันและในขณะเดียวกันก็สามารถแยกกลุ่มข้อมูลที่มีคุณลักษณะต่างกันออกจากกัน โดยอาศัยวิธีการวัดความคล้ายแบบใดแบบหนึ่ง

โดยปกติแล้วการทำงานของทั้งสองประเภทนี้จะแยกจากกันอย่างเด็ดขาด แต่ในวิทยานิพนธ์นี้นำเสนอโครงสร้างของระบบที่พยายามใช้การจัดกลุ่มข้อมูลมาช่วยเพิ่มประสิทธิภาพในการจำแนกประเภทข้อมูล

2.2.4 โครงสร้างทั่วไปของการทำเหมืองข้อมูล

หลังจากที่เราได้เลือกใช้วิธีการทำเหมืองข้อมูลแบบใดแล้วสิ่งที่ต้องพิจารณาต่อไปคือกระบวนการสร้างและนำไปใช้งานซึ่งทั้งการจำแนกประเภทและการจัดกลุ่มข้อมูลจะมีกระบวนการหลัก ๆ คล้ายกันดังนี้คือ

- เลือกวิธีการจำแนกหรือการจัดกลุ่มข้อมูล
- เลือกวิธีการวัดประสิทธิภาพของโมเดลที่ได้หรือผลลัพธ์ในการจัดกลุ่ม
- หลังจากนั้นจะทำการเรียนรู้จากข้อมูลที่มีอยู่โดยทำซ้ำกระบวนการดังต่อไปนี้
  - ดึงข้อมูลเข้ามาทีละอันหรือทั้งหมด
  - ทำการปรับโมเดลที่ใช้ในการเรียนรู้หรือจัดกลุ่มข้อมูลตามเงื่อนไข
  - ทำการวัดประสิทธิภาพของโมเดลที่ได้หรือวัดประสิทธิภาพของกลุ่มที่ได้
  - หยุดเมื่อถึงเงื่อนไขบางอย่าง เช่นจำนวนรอบที่กำหนด หรือค่าความผิดพลาดที่ได้ต่ำกว่าค่าที่กำหนด

เมื่อเสร็จสิ้นกระบวนการเราก็สามารถนำโมเดลในการจำแนกประเภทหรือกลุ่มที่ได้ในการจัดข้อมูลไปใช้ในการแก้ปัญหาต่าง ๆ ได้

2.3 การจำแนกประเภท

การจำแนกประเภทคือการทำนายชนิดหรือประเภทของข้อมูล โดยอาศัยโมเดลที่ผ่านการเรียนรู้จากข้อมูลที่มีอยู่ในฐานข้อมูล การสร้างโมเดลเพื่อใช้งานจริงนั้นจะประกอบไปด้วยสองขั้นตอนหลัก คือขั้นตอนการเรียนรู้ของโมเดล (Training phase) คือการสอนโมเดลเพื่อดึงเอาความรู้หรือรูปแบบที่เป็นประโยชน์ออกมาจากข้อมูลที่มีอยู่ในฐานข้อมูล ซึ่งตัวโมเดลที่ได้อาจจะ

อยู่ในรูปแบบโมเดลทางคณิตศาสตร์ หรือกฎ การทดสอบโมเดล (Testing phase) คือการทดสอบโมเดลที่ได้กับข้อมูลใหม่ที่ไม่ได้อยู่ในฐานข้อมูลที่ใช้ในการเรียนรู้

โดยทั่วไปแล้วข้อมูลจะถูกแบ่งออกเป็นสองส่วนคือ ส่วนแรกข้อมูลส่วนที่ใช้ในการเรียนรู้ของโมเดลซึ่งจะอยู่ในรูปแบบเวคเตอร์ของคุณลักษณะและคลาสหรือชนิดของข้อมูล โมเดลจะนำข้อมูลส่วนนี้ไปทำการปรับค่าต่าง ๆ ของโมเดลเพื่อให้เหมาะสมกับข้อมูล ส่วนที่สองข้อมูลที่ใช้ในการทดสอบโมเดลจะเหมือนกับข้อมูลที่ใช้ในการเรียนรู้ เพียงแต่ข้อมูลส่วนนี้จะใช้ทดสอบประสิทธิภาพการทำนายของโมเดล ในการทดสอบโมเดลระบบจะได้รับเพียงแค่ส่วนของคุณลักษณะหลังจากนั้นระบบจะทำการทำนายคลาสออกมา เราสามารถนำเอาค่าที่ระบบทำนายเปรียบเทียบกับคลาสที่แท้จริงได้เพื่อหาค่าประสิทธิภาพดังสมการ 2.1 ข้อมูลที่ใช้ในการทดสอบนี้จะบ่งบอกว่าโมเดลที่ได้มีความเป็นทั่วไป (Generalization) มากน้อยเพียงใด ซึ่งคุณสมบัตินี้เองที่บ่งบอกว่าโมเดลที่ได้นั้นเหมาะสมหรือไม่เมื่อนำไปใช้งานจริงกับข้อมูลที่ไม่เคยผ่านการเรียนรู้มาก่อน

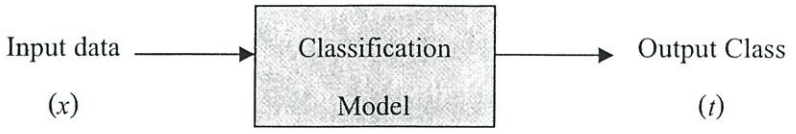
ข้อมูลส่วนที่ใช้ในการเรียนรู้จะประกอบไปด้วยเซตของข้อมูลดังนี้

$$D = (x_1, t_1), \ldots, (x_m, t_n), \ldots, (x_M, t_N)$$

ข้อมูลที่เข้ามาหนึ่งครั้งจะประกอบไปด้วย (x_i, t_j) โดยที่ $x_i \in X$ คืออินพุตเวคเตอร์ของระบบ และ $t_j \in T$ คือเอาท์พุทของข้อมูล ซึ่งเอาท์พุท T นั้นจะอยู่ในรูปดิสครีต T= $c_1, c_2, .. c_i$ โดยที่ c_i คือคลาสที่เป็นไปได้ของข้อมูล เมื่อเราพิจารณาฟังก์ชั่น $\phi : X \rightarrow T$ ซึ่งทำการแม็พอินพุทสเปช X ไปยังเอาท์พุตสเปช T การจำแนกข้อมูลก็คือการเรียนรู้เพื่อที่จะสร้างฟังก์ชั่นเป้าหมาย $f(x;\xi)$ เพื่อประมาณค่าฟังก์ชั่น ϕ

โดยปกติข้อมูลส่วนที่ใช้ในการเรียนรู้และข้อมูลส่วนที่ใช้ทดสอบจะเป็นตัวแทนของขอบเขตปัญหา ซึ่งข้อมูลที่ใช้ทดสอบจะต้องถูกแยกออกจากข้อมูลที่ใช้ในการเรียนรู้ไม่เช่นนั้นแล้วเมื่อเราวัดประสิทธิภาพในการเรียนรู้โดยใช้ชุดข้อมูลทดสอบจะไม่สะท้อนประสิทธิภาพของการเรียนรู้ที่แท้จริงออกมาเนื่องจากข้อมูลที่ใช้ทดสอบได้ผ่านการเรียนรู้มาแล้ว ในส่วนของการปรับโมเดลจะเกิดขึ้นเฉพาะขั้นตอนของการเรียนรู้เท่านั้น ขั้นตอนของการทดสอบเป็นเพียงการวัดประสิทธิภาพของโมเดลซึ่งจะไม่มีการปรับพารามิเตอร์ใด ๆ ในโมเดลทั้งสิ้น

การจำแนกประเภทจะเริ่มจากการสร้างโมเดลจากชุดข้อมูลที่ใช้ในการเรียนรู้หลังจากเรียนรู้จนกระทั่งถึงเงื่อนไขใด ๆ แล้วก็จะนำโมเดลตัวนั้นมาใช้ในการจำแนกข้อมูลที่เข้ามาใหม่ได้



รูปที่ 2.3 กระบวนการแม็พอินพุต x ไปยังเอาท์พุต y

รูปที่ 2.3 แสดงกระบวนในการทดสอบโมเดลที่ได้หลังจากการเรียนรู้แล้ว เราอาจจะพิจารณาโมเดลคล้ายกับกล่องดำ (Black box) ที่เราไม่รู้ว่าภายในเป็นอย่างไร โดยที่กล่องดำนี้จะทำการแม็พอินพุต x ไปยังเอาท์พุต y เมื่อโมเดลผ่านการเรียนรู้แล้วเราก็สามารถนำโมเดลนี้ไปใช้งานได้ เมื่อข้อมูลมีพฤติกรรมหรือลักษณะที่เปลี่ยนไป หลังจากที่ทำการเก็บข้อมูลใหม่แล้วเราก็สามารถนำโมเดลมาเรียนรู้และนำไปใช้งานใหม่ได้ แต่ในบางกรณีโมเดลสามารถเรียนรู้หรือปรับตัวได้เองเมื่อผู้ใช้ตอบสนอง (feedback) กลับมายังโมเดล

โมเดลที่ได้หลังจากการเรียนรู้เราไม่อาจจะสรุปได้ว่าโมเดลนั้นเป็นโมเดลที่ดีที่สุดและเหมาะสมกับข้อมูลทุกชุด เนื่องจากทฤษฎีบทที่ชื่อว่า "No Free Lunch Theorem" ของ Wolpert และ Macready [5] ได้กล่าวไว้โดยสรุปว่า ไม่มีโมเดลใดโมเดลหนึ่งที่จะมีประสิทธิภาพดีกว่าโมเดลอื่นๆ ทุกตัวในทุก ๆ ปัญหา ในการที่จะตรวจสอบโมเดลที่ได้ว่าดีหรือไม่นั้น โดยทั่วไป [4, 6, 7] ก็จะมีเงื่อนไขในการตรวจสอบอยู่หลายแบบดังดังต่อไปนี้

- ความถูกต้องของโมเดลในการทำนายข้อมูลที่ใช้ทดสอบ (Predictive accuracy) จะเป็นตัวทดสอบความเป็นทั่วไปของโมเดลที่ได้หลังจากการเรียนรู้ กล่าวคือโมเดลที่ได้มักจะเป็นโมเดลที่ปรับค่าพารามิเตอร์ต่าง ๆ ตามข้อมูลที่ใช้ในการเรียนรู้ ถ้าโมเดลใดไม่มีความเป็นทั่วไปค่าความถูกต้องที่ได้เมื่อทดสอบกับชุดข้อมูลทดสอบจะต่ำกว่าโมเดลที่มีความเป็นทั่วไป โดยทั่วไปค่าความถูกต้องของโมเดลสามารถหาได้จากสมการดังต่อไปนี้

$$Accuracy(M) = \frac{n_{true}}{N} \qquad (2.1)$$

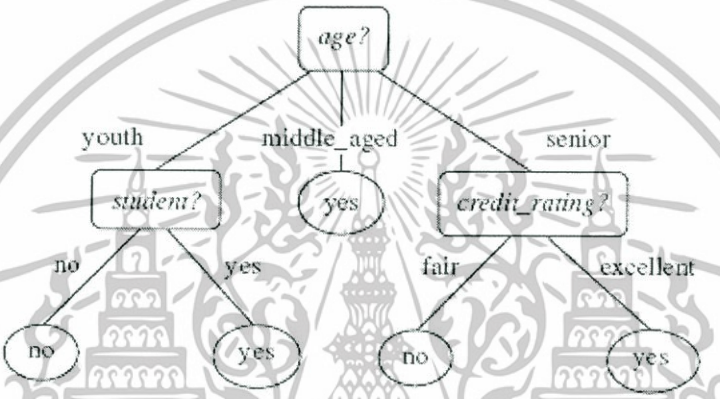
โดยที่ n_{true} คือจำนวนข้อมูลชุดทดสอบที่โมเดลนี้ทำนายถูกต้อง N คือจำนวนชุดข้อมูลทดสอบทั้งหมด

- ขนาดของโมเดลและเวลาที่ใช้ในการเรียนรู้ (Compactness and learning speed) แน่นอนว่าเราต้องการได้โมเดลที่มีขนาดเล็กซึ่งทำให้เราประหยัดหน่วยความจำและใช้เวลาในการเรียนรู้น้อยในขณะที่ค่าความถูกต้องจะต้องไม่น้อยกว่าอย่างมีนัยสำคัญเมื่อเทียบกับโมเดลที่มีขนาดใหญ่ ในบางกรณีอาจจะมีค่าความถูกต้องสูงกว่าได้
- การแทนความรู้ของโมเดลที่ได้หลังจากการเรียนรู้ (Representation) โมเดลที่ดีนั้นไม่จำเป็นที่จะต้องมีค่าความถูกต้องสูงเพียงอย่างเดียว แต่ควรจะเป็นโมเดลที่มนุษย์สามารถทำความเข้าใจความรู้ที่ดึงออกมาได้

2.3.1 การแทนความรู้ของโมเดล

การแทนความรู้ของโมเดลที่ได้หลังจากการเรียนรู้นั้น อาจจะถือได้ว่าเป็นวิธีในการเลือกโมเดลหรืออัลกอริธึมที่ใช้ในการเรียนรู้ โดยทั่วไปการแทนความรู้ที่ได้จะเป็นไปได้หลายรูปแบบ เช่น

- รูปแบบต้นไม้ที่ใช้ในการตัดสินใจ (Decision tree) ความรู้ที่ได้จะอยู่ในรูปแบบของต้นไม้ (tree) ดังรูปที่ 2.4 โหนดภายในแต่ละโหนดจะแทนคุณลักษณะที่ใช้ในการทดสอบ แขนของต้นไม้จะแทนเอาท์พุตของการทดสอบคุณลักษณะ โหนดปลาย (leaf node) จะแทนคลาสของข้อมูล



รูปที่ 2.4 ต้นไม้ที่ใช้ในการตัดสินใจ[2]

- รูปแบบโครงข่ายนิวรอน(Neural Network) ความรู้ที่ถูกแทนจะอยู่ในรูปแบบของโครงข่ายเชื่อมโยงระหว่างนิวรอน ส่วนใหญ่เราจะแทนโครงข่ายนิวรอนที่ได้หลังจากการเรียนรู้แล้วเป็นกล่องดำ (Black box) เพื่อเป็นการบ่งบอกว่าความรู้ที่ได้นั้นยากต่อการทำความเข้าใจ ถึงแม้ว่าในปัจจุบันจะมีการแปลงโครงข่ายนิวรอนให้อยู่ในรูปแบบของกฎ [8, 9] แต่กฎที่ได้ก็ยังไม่สามารถสื่อความหมายออกมาได้อย่างชัดเจน
- รูปแบบของกฎที่ใช้ในการตัดสินใจ (Decision rules) ถือเป็นวิธีการหนึ่งที่ได้รับความนิยมอย่างกว้างขวาง รูปแบบของความรู้ที่ได้จะอยู่ในลักษณะของเซตของกฎ IF-THEN

  IF *เงื่อนไข* THEN *แอคชั่น*

  ตัวอย่างเช่น

  R1: IF age = youth AND student = yes THEN buys_computer = yes

  ส่วนของ IF เราจะเรียกว่า*เงื่อนไข* ส่วนของ THEN จะเรียกว่า*แอคชั่น* ส่วนของเงื่อนไขก็จะประกอบไปด้วยคุณลักษณะของข้อมูลที่ต้องการตรวจสอบ โดยแต่ละคุณลักษณะจะเชื่อมด้วยลอจิก AND ส่วนของเงื่อนไขทั้งจะให้ค่าเป็นจริงหรืออาจจะกล่าวว่ากฎนั้นถูก

---

[2] I. H. Witten, and E. Frank, "Data mining: practical machine learning tools and techniques, 2 nd Edition" Morgan Kaufmann, 2005.

นำไปใช้ก็ต่อเมื่อทุก ๆ คุณลักษณะที่ตรวจสอบมีค่าเป็นจริงเท่านั้น ส่วนของแอคชั่นคือเมื่อกฎนั้นถูกนำไปใช้แล้วจะกระทำอะไร โดยส่วนมากในการจำแนกประเภทข้อมูลส่วนของแอคชั่นก็จะเป็นการตัดสินคลาสของข้อมูล

ในบทที่ 5 เราจะกล่าวถึงโมเดลหรืออัลกอริธึมของการจำแนกประเภท 2 แบบ แบบแรกการเรียนรู้แบบวิวัฒนาการ (Evolution Learning) และ โครงข่ายนิวรอน (Artificial Neural Network) ซึ่งเป็นอัลกอริธึมที่ใช้ในวิทยานิพนธ์ฉบับนี้

2.4 การจัดกลุ่มข้อมูล

การจัดกลุ่มข้อมูล [4] คือการแบ่งข้อมูลออกเป็นกลุ่ม ๆ โดยข้อมูลที่กลุ่มเดียวกันจะมีคุณลักษณะบางอย่างคล้ายกัน และข้อมูลที่อยู่กลุ่มต่างกันมีคุณลักษณะบางอย่างต่างกัน ข้อมูลที่ใช้ในการจัดกลุ่มนั้นจะมีลักษณะต่างจากข้อมูลที่ใช้ในการจำแนกประเภท กล่าวคือข้อมูลที่ใช้สร้างโมเดลในจำแนกประเภทนั้นจะประกอบไปด้วย เวคเตอร์ของคุณลักษณะและคลาส แต่ข้อมูลที่ใช้ในการจัดกลุ่มจะมีเพียงแค่เวคเตอร์ของคุณลักษณะเท่านั้น เนื่องจากจุดมุ่งหมายของการจัดกลุ่มข้อมูลคือการพยายามกำหนดคลาสหรือกลุ่มให้กับข้อมูลอาศัย โดยตั้งอยู่บนสมมุติฐานที่ว่าข้อมูลที่อยู่ในกลุ่มเดียวกันจะมีคุณลักษณะบางอย่างคล้ายกัน โดยเราไม่จำเป็นที่จะต้องคำนึงถึงค่าความถูกต้องในการทำนายข้อมูลที่ไม่เคยเห็นมาก่อนซึ่งจุดนี้เองทำให้การจัดกลุ่มต่างจากการจำแนกประเภท ข้อแตกต่างอีกประการคือ การจัดกลุ่มข้อมูลมุ่งเน้นในการหาความสัมพันธ์ของข้อมูล แต่การจำแนกประเภทมุ่งเน้นการประเมินว่าการจัดแบ่งข้อมูลที่ได้จะต้องสอดคล้องกับกลุ่มกำหนดไว้ก่อนหน้า [10]

ข้อมูลที่ใช้ในการจัดกลุ่มสามารถแสดงในรูปแบบ $X=\{x_1, x_2, ..,x_n\}$ โดยที่ N คือจำนวนข้อมูลทั้งหมด ข้อมูลแต่ละตัวจะแสดงได้อยู่ในรูป $x_j=\{x_{j1},x_{j2}, ...,x_{jd}\}^T \in \Re^d$ โดยที่ d คือจำนวนมิติคุณลักษณะของอินพุต ในการบ่งบอกลักษณะความคล้ายกันของข้อมูลภายในกลุ่มนั้นเราจะอาศัยฟังก์ชั่นวัดระยะห่างหรือฟังก์ชั่นวัดความคล้ายระหว่างข้อมูล โดยทั่วไปจะมีการวัดอยู่สองประเภทคือ การวัดระยะห่าง (Distance measures) และการวัดความคล้าย (Similarity measures)

- การวัดระยะห่างแบบ Minkowski เมื่อมีข้อมูลสองตัว $x_i = (x_{i1}, x_{i2},..., x_{ip})$ และ $x_j = (x_{j1}, x_{j2},..., x_{jp})$ ระยะห่างระหว่างข้อมูลทั้งสองตัวสามารถคำนวณได้จาก

$$d(x_i, x_j) = (|x_{i1} - x_{j1}|^g + |x_{i2} - x_{j2}|^g + ... + |x_{ip} - x_{jp}|^g)^{1/g} \quad (2.2)$$

ในกรณีที่ $g = 2$ จะเป็นการวัดระยะแบบยูคลิด (Euclidian distance) ซึ่งเราจะคุ้นเคยกันดี ในกรณีที่ $g = 1$ จะเป็นการวัดระยะที่มีชื่อว่าแมนฮัตตัน (Manhattan metric) การเลือกใช้การวัดระยะห่างนั้นอาจจะมีผลกระทบต่อการวิเคราะห์ผลลัพธ์ที่ได้ออกมา ดังนั้นเราอาจจะมีการกำหนด

น้ำหนักของแต่ละคุณลักษณะ (feature) ของข้อมูลก็ได้ โดยการวัดระยะก็จะอยู่ในรูปของการวัดระยะแบบมีน้ำหนักดังนี้

$$d(x_i, x_j) = (w_1 \mid x_{i1} - x_{j1} \mid^g + w_2 \mid x_{i2} - x_{j2} \mid^g + \ldots + w_p \mid x_{ip} - x_{jp} \mid^g)^{1/g} \quad (2.3)$$

โดยที่ $w_i = [0, \infty)$

- การวัดความคล้ายแบบ Cosine จะใช้การวัดมุมระหว่างเวคเตอร์ของข้อมูล 2 ตัวแสดงความคล้ายกันของข้อมูลทั้งสองได้ ซึ่งสามารถหาได้จากผลคูณภายในของเวคเตอร์ทั้งสองดังสมการ

$$s(x_i, x_j) = \frac{x_i^T . x_j}{\| x_i \| . \| x_j \|} \quad (2.4)$$

หลังจากที่จัดกลุ่มได้แล้วก็จะต้องมีการประเมินประสิทธิภาพในการจัดกลุ่มของอัลกอริธึ่มนั้นว่าจัดกลุ่มออกมาได้ดีหรือไม่อย่างไร แต่ก็เป็นเรื่องค่อนข้างยากที่จะบอกได้ว่ากลุ่มที่จัดออกมาได้ดีหรือไม่ ในบางกรณีอาจจะต้องใช้คนเป็นผู้พิจารณาด้วยสายตาเองว่ากลุ่มที่จัดออกมาเหมาะสมหรือไม่ อย่างไรก็ตามมีหลายวิธีการพัฒนาวิธีมาเพื่อวัดประสิทธิภาพของการจัดกลุ่ม โดยจะแบ่งออกเป็นการวัดประสิทธิจากภายใน (Internal evaluation) และการวัดประสิทธิภาพจากภายนอก (external evaluation)

- การวัดที่อ้างอิงจากภายใน(Internal quality) การวัดประเภทนี้ไม่จำเป็นต้องอาศัยข้อมูลใดๆ จากภายนอก โดยการวัดจะอาศัยการเปรียบเทียบฟังก์ชั่นที่ในการวัดความคล้าย เช่น การวัดค่า Mean squared error ซึ่งเป็นที่นิยมเป็นอย่างมาก การวัดความผิดพลาดสามารถวัดได้จากการพิจารณาว่าข้อมูลนั้นอยู่ห่างจากจุดศูนย์กลางกลุ่มเท่าไหร่ ซึ่งสามารถแสดงสมการได้ดังนี้

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} \| x - c_i \|^2 \quad (2.5)$$

โดยที่ k คือจำนวนกลุ่มทั้งหมด C_i คือกลุ่มแต่ละกลุ่ม x คือสมาชิกของกลุ่มนั้น

||x - ci|| คือระยะห่างระหว่างข้อมูลกับจุดศูนย์กลางของกลุ่ม

- การวัดที่อ้างอิงจากภายนอก(External quality) การวัดประเภทนี้จำเป็นต้องอาศัยข้อมูลจากภายนอกเข้ามาช่วยตัดสิน เช่น ข้อมูลกลุ่มของเอกสารที่ถูกกำหนดไว้ล่วงหน้าโดยผู้เชี่ยวชาญ เช่น การวัดค่าพลังงาน(Entropy) เป็นการวัดโดยอาศัยค่าความน่าจะเป็นของข้อมูลที่อยู่ในกลุ่ม นั้นคือ ค่าเอนโทรพีจะมีค่าเป็นศูนย์เมื่อสมาชิกทุกตัวของกลุ่มจัดขึ้น

ใหม่นั้นเป็นกลุ่มเดียวกันกับกลุ่มที่จัดโดยผู้เชี่ยวชาญ ค่าเอนโทรพีจะมีค่าสูงสุดเมื่อสมาชิกทุกตัวในกลุ่มใหม่เป็นคนละกลุ่มกับกลุ่มที่จัดโดยผู้เชี่ยวชาญ

$$E_j = -\sum_i p_{ij} \log(p_{ij}) \tag{2.6}$$

โดยที่ค่าเอนโทรพีรวมของทุกกลุ่มสามารถหาได้จาก

$$E_{cs} = \sum_{j=1}^{m} \frac{n_j \times E_j}{n} \tag{2.7}$$

โดยที่ p คือความน่าจะเป็นที่สมาชิกของกลุ่ม i ขึ้นอยู่กับกลุ่ม j

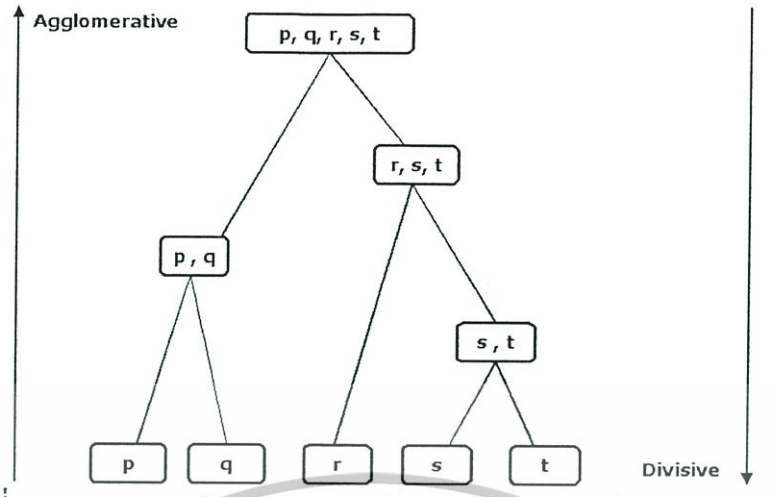
n_j คือ จำนวนข้อมูลทั้งหมดในกลุ่ม j

n คือ จำนวนข้อมูลทั้งหมด

2.4.1 ประเภทการจัดกลุ่มข้อมูล

ในการจัดกลุ่มข้อมูลเราสามารถจัดประเภทของวิธีการจัดกลุ่มข้อมูลหลัก ๆ ได้ดังนี้

- แบบโครงสร้างแบบลำดับชั้น (Hierarchical method) โดยที่ชั้นบนสุดจะมองเห็นเป็นกลุ่มเดียวกันหมด และชั้นล่างสุดข้อมูลแต่ละอันถือว่าเป็นหนึ่งกลุ่ม ในชั้นตรงกลางเกิดจากการรวมกันของสองกลุ่มในชั้นต่ำกว่า หรือจะมองว่าเกิดจากการแตกออกเป็นสองกลุ่มของชั้นสูงกว่าก็ได้ แผนภาพที่นิยมใช้แสดงการจัดกลุ่มแบบโครงสร้างแบบลำดับชั้นเรียกว่าแผนภาพเดนโดแกรม (Dendogram) ซึ่งมีลักษณะคล้ายกับโครงสร้างแบบต้นไม้
- แบบอาศัยโมเดลทางสถิติ (Model-based) การแบ่งแบบนี้จะอาศัยฟังก์ชั่นทางสถิติมาช่วยในการกำหนดค่าความเป็นไปได้ว่าข้อมูลมีโอกาสเป็นสมาชิกกลุ่มหนึ่ง ๆ เท่าไหร่ ส่วนใหญ่จะใช้เกาส์เซียนฟังก์ชั่น (Gaussian function) [11]



รูปที่ 2.5 แผนภาพเดนโดแกรมของการจัดกลุ่มแบบโครงสร้างแบบลำดับชั้น

- แบบแบ่งเป็นกลุ่มแยกจากกัน (Partitioning) การจัดกลุ่มข้อมูลจะอาศัยการแบ่งข้อมูลที่มีอยู่ออกเป็น k กลุ่ม ปกติแล้ว $k \leq N$ ซึ่งมีเงื่อนไขดังนี้คือ แต่ละกลุ่มต้องมีข้อมูลอย่างน้อยหนึ่งข้อมูล และแต่ละข้อมูลสามารถเป็นสมาชิกกลุ่มได้เพียงกลุ่มเดียว ในบางกรณีข้อมูลอาจจะเป็นสมาชิกของกลุ่มได้มากกว่าหนึ่งกลุ่ม เช่น การจัดกลุ่มแบบฟัซซี่ [12]

บทต่อไปจะกล่าวถึง การทบทวนบทความต่าง ๆ แรงจูงใจ รวมถึงแนวทางการวิจัยของวิทยานิพนธ์ฉบับนี้

บทที่ 3

งานวิจัยที่เกี่ยวข้องและแนวคิดเบื้องต้น

3.1 บทนำ

ในบทที่ 2 ได้พูดถึงภาพรวมของกระบวนการค้นหาความรู้ในฐานข้อมูล การทำเหมืองข้อมูลซึ่งถือเป็นหนึ่งในกระบวนการค้นหาความรู้ในฐานข้อมูลที่สำคัญ รวมถึงเกริ่นนำประเภทในการทำเหมืองข้อมูลสองประเภทหลักคือ กระบวนการจำแนกประเภทและกระบวนการจัดกลุ่มซึ่งถือเป็นข้อมูล การจำแนกประเภทถือได้ว่าเป็นกระบวนการที่เกิดขึ้นบ่อยมากในชีวิตประจำวันของเรา เช่น การพยายามจำแนกวัตถุออกเป็นชนิดต่าง ๆ โดยอาศัยประสบการณ์ที่เราได้เห็นวัตถุที่มีลักษณะใกล้เคียงกับวัตถุสิ่งนั้น กระบวนการตัดสินใจในการลงทุนซื้อหุ้นโดยอาศัยข้อมูลของตลาดหุ้นที่ผ่านมา หรือกระบวนการตรวจจับวัตถุต้องห้ามในสนามบิน การจัดกลุ่มข้อมูลก็เป็นอีกเป็นกระบวนการพื้นฐานที่สำคัญในการวิเคราะห์ข้อมูล เช่น ในการวิเคราะห์ทางเศรษฐศาสตร์เราอาจจะต้องการทราบกลุ่มประเทศที่มีลักษณะทางเศรษฐกิจคล้ายกันเพื่อการขยายฐานในการลงทุน แม้กระทั้งการจัดกลุ่มพฤติกรรมผู้บริโภคที่มีลักษณะคล้ายกัน หรือ การจัดกลุ่มเอกสารต่าง ๆ เพื่อให้ง่ายต่อการค้นหาเป็นต้น

ปัจจุบันแนวโน้มปริมาณข้อมูลที่เพิ่มขึ้นอย่างมากมายมหาศาล ทำให้การจำแนกประเภทใช้เวลาในการเรียนรู้ข้อมูลที่ใช้ในการฝึกสอนนานมาก อีกทั้งโมเดลของระบบจำแนกประเภทที่ได้อาจจะมีขนาดใหญ่และซับซ้อนไม่เหมาะสมที่จะนำไปใช้งานจริง วิทยานิพนธ์ฉบับนี้นำเสนอการใช้งานร่วมกันของจักรกลเรียนรู้ที่ใช้ในการจัดกลุ่มข้อมูลกับจักรกลเรียนรู้ที่ใช้ในการจำแนกประเภท เพื่อเพิ่มประสิทธิภาพในการจำแนกประเภทและลดเวลาในการเรียนรู้ โดยตัวจำแนกประเภท (อัลกอริธึมหรือโมเดลที่ใช้ในการจำแนกประเภทข้อมูล) ที่มุ่งเน้นในการวิจัยนี้คือ ระบบจำแนกประเภทแบบเรียนรู้ได้ (Learning Classifier System – LCS)

ในบทนี้จะกล่าวถึงการสำรวจบทความ ปัญหาและแนวทางที่วิทยานิพนธ์ฉบับนี้มุ่งที่จะศึกษา วิจัย โดยในหัวข้อ 3.2 จะกล่าวถึงภาพรวมขอบระบบจำแนกประเภทแบบเรียนรู้ได้ (Learning Classifier System – LCS) และปัญหาที่เกิดขึ้นเมื่อนำระบบจำแนกประเภทแบบเรียนรู้ได้ไปใช้งานกับฐานข้อมูลที่มีขนาดใหญ่ หัวข้อที่ 3.3 กล่าวถึงประเภทของการแตกปัญหา รวมถึงวิธีการต่าง ๆ ที่นิยมใช้ในการแตกปัญหา หัวข้อที่ 3.4 กล่าวถึงแนวทางการพัฒนาระบบที่วิทยานิพนธ์ฉบับนี้นำเสนอเพื่อพัฒนาประสิทธิภาพโดยรวมของระบบจำแนกประเภทแบบเรียนรู้ได้

3.2 ทบทวนบทความ

ระบบจำแนกประเภทแบบเรียนรู้ได้ (Learning Classifier System – LCS) เป็นจักรกลเรียนรู้ที่อยู่ในรูปแบบของกฎซึ่งมีการทำปรับปรุงเป็นเวอร์ชั่นต่าง ๆ ออกมามากมาย LCS ถูกนำเสนอครั้งแรกในปี 1973 โดย Holland [13] ซึ่งมีส่วนประกอบหลักอยู่สองส่วนและเป็นพื้นฐานของเวอร์ชั่นอื่น ๆ ด้วย ส่วนแรกจะเป็นในส่วนของกระบวนการเรียนรู้ โดย LCS จะอาศัยกระบวนการเรียนรู้แบบรีอินฟอสเมนต์ (Reinforcement learning) เพื่อใช้ในการปรับปรุงค่าพารามิเตอร์ต่าง ๆ ของกฎ โดยจะอาศัยค่าตอบแทน (Reward) ที่ได้รับจากสภาพแวดล้อมนั้น ๆ ส่วนที่สองจะอาศัยกระบวนการเจเนติกในการสร้างกฎใหม่ขึ้นมา โดยหวังว่ากฎใหม่ที่ได้มานั้นจะเป็นกฎที่มีประสิทธิ ภาพในการจำแนกประเภทข้อมูลดีขึ้น ข้อดีของ LCS สามารถสรุปได้ดังนี้คือ ประการแรกคือ เนื่องจาก LCS เป็นจักรกลเรียนรู้ที่อยู่ในรูปแบบของกฎ ดังนั้นมนุษย์จึงสามารถแปลความหมายและทำความเข้าใจความรู้ได้ที่ได้จากระบบนี้ ประการที่สอง LCS เป็นระบบแบบออนไลน์ กล่าวคือLCS มีการปรับค่าพารามิเตอร์ต่าง ๆ แบบทันทีหลังจากมีข้อมูลผ่านเข้ามา ดังนั้นระบบจึงสามารถปรับเปลี่ยนได้เมื่อสภาพแวดล้อมมีการปรับเปลี่ยนไปได้ LCS จึงเหมาะสมสำหรับการทำเหมืองข้อมูลแบบที่มีข้อมูลเข้ามาแบบต่อเนื่อง (Stream data mining) ซึ่งกำลังเป็นหัวข้อวิจัยที่ได้รับความสนใจเป็นอย่างมากในปัจจุบัน ประการที่สาม ระบบที่ได้จะสามารถครอบคลุมขอบเขตของปัญหา เนื่องจาก LCS อาศัยกระบวนการเจเนติกเป็นพื้นฐานในการสร้างกฎใหม่ จึงทำให้ระบบสามารถที่จะสำรวจขอบเขตของปัญหาได้อย่างมีประสิทธิภาพ

หลังจากนั้นในปี 1995 [14, 15] Wilson ได้ปรับปรุงและพัฒนาระบบ LCS โดยใช้ชื่อว่า XCS (eXtended Classifier System) ส่วนหลักที่ได้รับการปรับปรุงและพัฒนาคือ การปรับค่าความเหมาะสมให้ขึ้นอยู่กับค่าประมาณของความผิดพลาดในการทำนายค่าตอบแทน ระบบ XCS ได้รับความนิยมในการใช้งานด้านต่าง ๆ เป็นอย่างมากทั้งในด้านรีอินฟอสเมนต์เช่น การหาเส้นทางไปยังเป้าหมายในระบบหุ่นยนต์อัตโนมัติ [16] การประยุกต์ใช้งานในระบบควบคุมต่าง ๆ [17] และโดยเฉพาะอย่างยิ่งงานด้านการทำเหมืองข้อมูล

ในยุคแรก Saxon และ Barry [18] ได้ทำการทดสอบ XCS กับข้อมูลชื่อ Monk ซึ่งเป็นข้อมูลสังเคราะห์ได้ที่รับความนิยมในการทดสอบการทำเหมืองข้อมูล ผลการทดสอบแสดงให้เห็นว่า XCS ให้ประสิทธิภาพในการทำนายชุดข้อมูลทดสอบได้เทียบเท่ากับอัลกอริธึมจักรกลเรียนรู้อื่นเช่น นิวรอนเน็ตเวิร์ก, CN2, AQ-15 และ ID3 สามอัลกอริธึมสุดท้ายเป็นอัลกอริธึมที่ใช้สร้างกฎเช่นเดียวกับ XCS ถัดจากนั้นได้มีการนำเสนอการประยุกต์ใช้ XCS ในการทำเหมืองข้อมูลกับข้อมูลจริงเปรียบเทียบกับจักรกลเรียนรู้อื่น ๆ อาทิเช่น ปี 2001 Bernado และ Garrell-Guiu [19] ได้นำเสนอผลการเปรียบเทียบ XCS กับ 0-R, IB1, IBk, NBa, C4.5, PART และ SMO ในการทดลองกับข้อมูลจริง 15 ชนิด โดยส่วนใหญ่เป็นข้อมูลที่ได้มาจากฐานข้อมูล UCI KDD - [20] ซึ่งเป็นแหล่งรวบรวมข้อมูลที่เก็บได้จากสถานการณ์จริงและได้รับความนิยมในการทดสอบ

จักรกลเรียนรู้ด้านการจำแนกประเภท จากผลการทดลองแสดงให้เห็นว่า XCS นั้นให้ประสิทธิภาพที่ดีกว่า IB1 และ NBa อย่างมีนัยสำคัญและให้ประสิทธิภาพที่ไม่แตกต่างจากจักรกลเรียนรู้อื่น ๆ อย่างมีนัยสำคัญ ในปีเดียวกัน Dixon และทีมงาน [21] ได้ทดสอบ XCS และจักรกลเรียนรู้อื่นโดยใช้ข้อมูล 12 ชุดซึ่งส่วนใหญ่นำมาจากฐานข้อมูล UCI KDD และพบว่า XCS นั้นสามารถนำมาเป็นเครื่องมือในการทำเหมืองข้อมูลได้อย่างมีประสิทธิภาพ ถัดจากนั้นในปี 2004 Butz [22] ได้เสนอวิทยานิพนธ์ในระดับปริญญาเอกศึกษาเปรียบเทียบ XCS กับจักรกลเรียนรู้อื่น เช่น C4.5, Naïve Bayes Classifier, PART, instance based, Nearest neighbor และ SVM โดยใช้ข้อมูลในการทดสอบถึง 42 ชนิด โดยข้อมูลส่วนใหญ่นำมาจากฐานข้อมูล UCI KDD ผลการทดสอบแสดงให้เห็นว่า XCS ให้ประสิทธิภาพที่ดีกว่าบางอัลกอริธึมอย่างมีนัยสำคัญ

ในปี 2003 Bernadóและทีมงาน [23] ได้นำเสนอระบบจำแนกประเภทแบบเรียนรู้ได้อีกเวอร์ชั่นเรียกว่า UCS (sUpervised Classifier System) ซึ่งอาจจะถือได้ว่า UCS นั้นเป็นเวอร์ชั่นพิเศษของ XCS ที่ใช้ในงานสำหรับการจำแนกประเภทข้อมูลโดยเฉพาะ Bernadó และทีมงานแสดงให้เห็นว่า UCS นั้นสามารถเรียนรู้ได้เร็วกว่า XCS เนื่องจากใช้จำนวนกฎในการเรียนรู้ที่น้อยกว่า โดยเฉพาะอย่างยิ่งเมื่อใช้งานกับข้อมูลที่มีหลายคลาส

ปัจจุบันข้อมูลที่เก็บในองค์กรมีแนวโน้มเพิ่มขึ้นอย่างมหาศาล ทำให้มีความจำเป็นที่จะต้องปรับปรุงอัลกอริธึมในการทำเหมืองข้อมูลเพื่อให้สามารถรองรับข้อมูลที่เพิ่มขึ้นได้ เช่นเดียวกันเมื่อประยุกต์ใช้งาน LCSs (ในที่นี้ LCSs แทนถึงระบบจำแนกประเภททุกเวอร์ชั่น - LCS, XCS และ UCS) กับข้อมูลที่มีขนาดใหญ่ ธรรมชาติของ LCSs จะสร้างกฎขึ้นมาเป็นจำนวนมากและจะทำให้ระบบทำงานได้ช้าลง

ในปี 2003 Bagnall และ Cawley [24] ได้เริ่มทดสอบ XCS เพื่อศึกษาความเป็นไปได้ในการประยุกต์ใช้งานกับฐานข้อมูลที่มีขนาดใหญ่ โดยฐานข้อมูลที่นำมาทดสอบมีชื่อว่า Forest Cover Type ซึ่งจัดเป็นฐานข้อมูลที่มีขนาดใหญ่ มีจำนวนคุณลักษณะทั้งหมด 54 คุณลักษณะ ผสมกันทั้งคุณลักษณะแบบตัวเลขจำนวนจริงและแบบนอมินอล (Nominal) มีจำนวนเรคอร์ดข้อมูลทั้งหมด 581,012 เรคอร์ด มีเอาท์พุตคลาสทั้งหมด 7 คลาส ฐานข้อมูลนี้สามารถดาว์นโหลดได้ที่ UCI KDD งานวิจัยนี้ได้ปรับปรุงเงื่อนไขกระบวนการแลกเปลี่ยนโครโมโซมและกระบวนการผ่าเหล่าเพื่อให้เหมาะสมกับลักษณะข้อมูล อีกทั้งยังได้ปรับปรุงกระบวนการการลบกฎและกระบวนการเรียกใช้เจเนติก ผลการการทดสอบเปรียบเทียบกับอัลกอริธึมต้นไม้ (Tree) นิวรอนเน็ตเวิร์ค และ SVM รวม 8 ประเภทปรากฏว่า C5 มีประสิทธิภาพสูงที่สุด XCS มีประสิทธิภาพที่ดีกว่าอัลกอริธึม 3 ประเภทอย่างมีนัยสำคัญ ในปี 2005 Hai H. Dam และทีมงาน ได้นำเสนอการประยุกต์ใช้งาน XCS กับฐานข้อมูลขนาดใหญ่อีกชนิดคือฐานข้อมูลระบบตรวจจับผู้บุกรุก (Intrusion Detection System) ซึ่งสามารถดาว์นโหลดได้ที่ฐานข้อมูล UCI KDD เช่นเดียวกัน ฐานข้อมูลนี้มีคุณลักษณะทั้งหมดมากกว่า 40 คุณลักษณะมีจำนวนเรคอร์ดข้อมูลประมาณ 5 ล้านเรคอร์ด มีเอาท์พุตคลาสทั้งหมด 5 คลาส งานวิจัยชิ้นนี้ได้นำแสดงให้เห็นว่าการใช้งาน XCS

แบบดั้งเดิมนั้นให้ค่าประสิทธิภาพในการจำแนกประเภทข้อมูลไม่สูงมากนัก อีกทั้งบางเอาท์พุตคลาสไม่สามารถจำแนกประเภทได้เลย Hai H. Dam และทีมงานได้นำเสนอวิธีการปรับแต่ง XCS โดยได้ปรับแต่งในกระบวนการผ่าเหล่า กระบวนการลบกฎ และนำเสนอวิธีการตัดสินเอาท์พุตของ XCS จากการโหวตของกฎที่ใกล้เคียงกับอินพุตที่เข้ามาโดยอาศัยการวัดระยะห่างแบบยูคลิดช่วย ผลการทดลองแสดงให้เห็นว่า XCS รุ่นที่ปรับปรุงให้ค่าประสิทธิภาพในการจำแนกประเภทดีกว่า XCS รุ่นดั่งเดิม พร้อมทั้งเสนอแนวทางในการพัฒนา XCS เพื่อให้สามารถใช้งานได้จริง โดยควรมุ่งเน้นศึกษาวิจัยเพื่อให้ระบบ XCS สามารถเรียนรู้ได้เร็วขึ้น หนึ่งในวิธีการพัฒนาความเร็วในการเรียนรู้คือสร้างเป็นระบบแบบกระจาย (Distributed System) โดยให้ XCS แต่ละตัวจัดการกับเอาท์พุตคลาสที่แตกต่างกัน งานวิจัยของ Bagnall และ Hai H. Dam ถือเป็นการจุดเริ่มต้นเพื่อแสดงให้เห็นถึงแนวทางในการประยุกต์ใช้งานระบบ LCSs กับข้อมูลที่มีขนาดใหญ่

อย่างไรก็ตามในการนำระบบจำแนกประเภทแบบเรียนรู้ได้ LCSs ประเภทต่าง ๆ มาใช้งานกับข้อมูลขนาดใหญ่ LCSs จะให้จำนวนกฎที่มาก โดยเฉพาะอย่างยิ่งข้อมูลส่วนใหญ่เป็นข้อมูลแบบไม่เป็นเชิงเส้น (non-linear) มีคุณลักษณะ (feature) และเอาท์พุตคลาสจำนวนมาก ข้อมูลลักษณะนี้ยิ่งทำให้ระบบสร้างกฎจำนวนมาก เมื่อมีจำนวนกฎมากทำให้ใช้เวลาในการประมวลผลสำหรับข้อมูลที่กำลังเข้ามานานขึ้น เพราะระบบจะต้องทำการตรวจสอบทุกกฎที่มีอยู่ในระบบเพื่อค้นหากฎที่เข้ากันได้กับข้อมูลที่เข้ามา ดังนั้นจึงมีงานวิจัยจำนวนหนึ่งนำเสนออัลกอริธึมแบบต่าง ๆ ขึ้นเพื่อจัดการกับข้อมูลที่มีขนาดใหญ่ ซึ่งจากการสำรวจงานวิจัยต่าง ๆ สามารถสรุปเนื้อหาอย่างย่อได้ดังนี้

Hai H. Dam และทีมงาน [25, 26] ได้นำเสนอการประยุกต์ใช้งาน LCSs กับการทำเหมืองข้อมูลแบบกระจายซึ่งเรียกว่า DLCS ระบบจะแบ่งการทำงานเป็น XCS ลูกข่ายหลาย ๆ ระบบ และเครื่องแม่ข่ายที่เกิดจากการรวม XCS จากเครื่องลูกข่าย โดย XCS ลูกข่ายจะทำการเรียนรู้ข้อมูลไปพร้อม ๆ กันและจะทำการส่งเฉพาะกฎบางกฎไปยังเครื่องแม่ข่าย การหาเอาท์พุตจะหาที่เครื่องแม่ข่ายซึ่งหาได้จากการโหวตเสียงส่วนมาก (Majority vote) หรือวิธี Knowledge Probing การทดลองจะเน้นไปในการเปรียบเทียบปริมาณข้อมูลที่ส่งระหว่างเครื่องลูกข่ายกับเครื่องแม่ข่ายกับระบบแบบรวมศูนย์ (Centralize) ซึ่งต้องทำการส่งข้อมูลทุกตัวจากเครื่องลูกข่ายไปยังเครื่องแม่ข่าย และเครื่องแม่ข่ายจะต้องสร้าง XCS ขึ้นมาเรียนรู้ข้อมูลที่ได้รับจากเครื่องลูกข่ายทั้งหมดซึ่งจะใช้เวลาในการเรียนรู้นานกว่า ผลการทดลองแสดงให้เห็นว่า DLCS ลดปริมาณการส่งข้อมูลในเน็ตเวิร์กเปรียบเทียบกับระบบแบบรวมศูนย์และลดเวลาในการเรียนรู้ลงได้

Hai H. Dam และทีมงาน [6] นำเสนอการเปลี่ยนการแทนรูปแบบของกฎในส่วนของการกระทำ (Action) ของ UCS ให้อยู่ในรูปแบบของนิวรอนเน็ตเวิร์คโดยให้ชื่อว่า NLCS (Neural-Based Learning Classifier System) ผลการทดลองแสดงให้เห็นว่าสามารถที่จะลดจำนวนกฎในการเรียนรู้ลงได้อย่างมาก อีกทั้ง NLCS ยังให้ประสิทธิภาพในการจำแนกประเภทข้อมูลที่ดีกว่าหรือเทียบเท่ากับระบบ UCS เดิมในการทดลองกับข้อมูล 14 ชนิดจากฐานข้อมูล UCI KDD

Bull, L. และทีมงาน [27, 28] นำเสนอการระบบในการใช้งาน LCSs หลายตัวร่วมกัน (Ensemble) ซึ่งสามารถนำไปประยุกต์ใช้ได้กับ LCSs รุ่นต่าง ๆ ได้ ในงานวิจัยนี้ทดลองกับ YCS [29] ซึ่งเป็นรุ่นหนึ่งของ XCS ที่ตัดบางฟังก์ชั่นออกเพื่อให้ง่ายต่อการเข้าใจระบบเบื้องต้น และ MCS (Minimal Classifier System) ซึ่งเป็นอีกรุ่นหนึ่งของ LCSs เช่นกัน โดยนำเสนออัลกอริธึมการแลกเปลี่ยนกฎระหว่าง LCSs แต่ละตัว จากผลการทดลองแสดงเมื่อทดสอบกับปัญหามัลติเพล็กเซอร์ (Multiplexer – MUX) แสดงให้เห็นว่าอัลกอริธึมการแลกเปลี่ยนกฎสามารถเพิ่มความเร็วในการเรียนรู้ข้อมูลเมื่อเทียบกับใช้ LCSs เพียงระบบเดียว อีกทั้งยังสามารถทำแบบขนานไปพร้อมกันได้ด้วย

การแตกปัญหาใหญ่เป็นปัญหาย่อย ๆ หลายปัญหา (Problem decomposition) ก็ถือเป็นอีกแนวทางหนึ่งที่ใช้ในการจัดการกับข้อมูลที่มีขนาดใหญ่ แต่ยังไม่มีการประยุกต์ใช้กับระบบ LCSs แนวคิดหลักคือทำการแตกปัญหาเป็นปัญหาย่อยแบบอัตโนมัติ ซึ่งแต่ละปัญหาย่อยจะถูกจัดการโดยจักรกลเรียนรู้ที่แตกต่างกัน ส่วนใหญ่แล้วจะเป็นจักรกลเรียนรู้ประเภทเดียวกัน การแตกปัญหาเป็นปัญหาย่อยนี้คาดหวังว่าสามารถที่จะลดความซับซ้อนของกระบวนการเรียนรู้ที่เกิดจากการพยายามสร้างจักรกลเรียนรู้ขนาดใหญ่เรียนปัญหานั้นทั้งหมด เมื่อเราแตกเป็นปัญหาย่อยแล้วเราจะสามารถใช้กลุ่มของจักรกลเรียนรู้ซึ่งแต่ละตัวที่มีความซับซ้อนน้อยกว่าเรียนรู้ปัญหาย่อยแต่ละปัญหาได้

3.3 การแตกปัญหา

ในการศึกษาวิจัยของ Maimon, O. และ Rokach, L. [30] แสดงให้เห็นถึงข้อดีหลายประการในการแตกปัญหาเป็นปัญหาย่อย ประการแรกสามารถเพิ่มประสิทธิภาพในการจำแนกประเภทข้อมูล ประการที่สองการแตกปัญหาเป็นปัญหาย่อยสามารถลดความซับซ้อนของปัญหาได้ ประการที่สามเพิ่มขีดความสามารถและความเป็นไปได้ในการจัดการกับปัญหาขนาดใหญ่ ประการที่สี่ผลลัพธ์ของจักรกลเรียนรู้ที่ได้สามารถทำความเข้าใจได้ง่ายกว่า นอกจากนี้แล้วยังมีข้อดีอื่น ๆ อีก เช่น สามารถลดเวลาที่ใช้ในการเรียนรู้เนื่องจากจักรกลเรียนรู้จัดการกับปัญหาที่มีขนาดเล็กกว่าและมีความซับซ้อนน้อยกว่า เปิดโอกาสให้ระบบสามารถใช้จักรกลเรียนรู้ต่างประเภทกันสำหรับปัญหาย่อยแต่ละปัญหาเพื่อเพิ่มประสิทธิภาพให้สูงที่สุด

Maimon, O. และ Rokach, L. ได้แบ่งประเภทของการแตกปัญหาออกเป็นหลัก ๆ สองประเภทคือ

- การแตกปัญหาโดยการแปลงไปยังโดเมนอื่น (Intermediate concept decomposition) ปัญหาที่ต้องการแตกจะถูกแปลงไปยังอีกโดเมนโดยอาศัยฟังก์ชั่นและถูกแตกเป็นปัญหาย่อยที่โดเมนนั้น การแปลงไปยังอีกโดเมนสามารถทำได้สองแบบทั้งการแปลงคุณลักษณะหรือการแปลงเอาท์พุตคลาสของข้อมูลเริ่มต้น โดยที่คุณลักษณะหรือเอาท์พุตคลาสของข้อมูลอีกโดเมนอาจจะมีจำนวนไม่เท่ากับข้อมูลเริ่มต้น

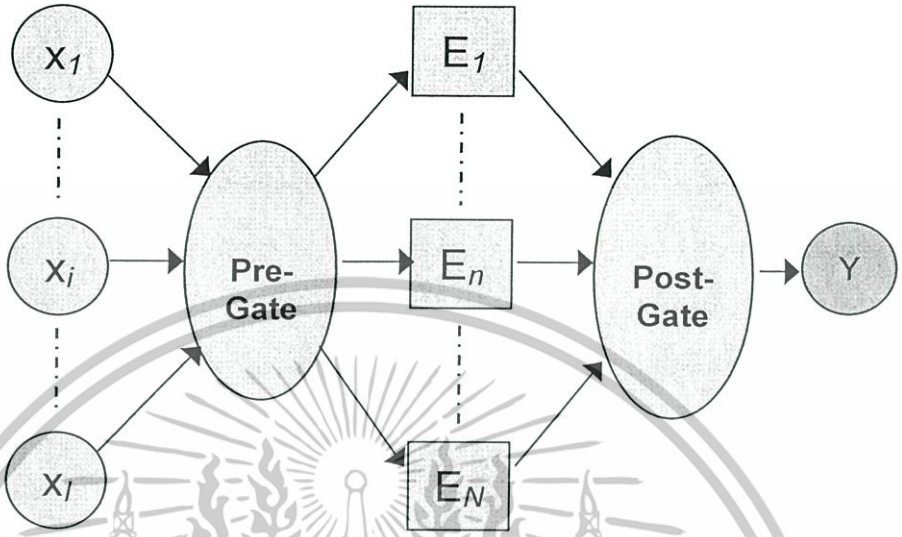
ตัวอย่างเช่น ในงานวิจัยของ Buntine [31] นำเสนอทำการจัดกลุ่มเอกสารออกเป็นกลุ่มต่าง ๆ ตามเรื่องที่กำหนด โดยหัวเรื่องของเอกสารทุกตัวจะถูกแปลงไปยังกลุ่มของหัวเรื่อง จากนั้นจึงทำการจัดกลุ่ม ในงานวิจัยของ Anand [32] ได้นำเสนอการแตกปัญหาสำหรับปัญหา K เอาท์พุตคลาสออกเป็นสองเอาท์พุตคลาส เอาท์พุตคลาสแรกเป็นเอาท์พุตคลาสที่ถูกเลือกส่วนอีกเอาท์พุตคลาสเป็นเอาท์พุตคลาสอื่น ๆ ซึ่งจะทำการเปลี่ยนเอาท์พุตคลาสแรกไปเรื่อย ๆ จนกระทั่งครบ โดยที่ปัญหาแต่ละปัญหาจะถูกเรียนรู้โดยอาศัยนิวรอนเน็ตเวิร์กแยกกัน สุดท้ายจะทำการรวมนิวรอนเน็ตเวิร์กแต่ละตัวอีกที

- การแตกปัญหาโดยตรงจากโดเมนเริ่มต้น (Original concept decomposition) การแตกปัญหาในลักษณะนี้ปัญหาจะถูกแตกโดยตรงไม่ต้องแปลงไปยังโดเมนอื่น ข้อมูลที่ใช้ในการเรียนรู้จะถูกแบ่งออกเป็นชุดข้อมูลย่อย ๆ แยกจากกัน จากนั้นชุดข้อมูลแต่ละชุดจะถูกนำไปเรียนรู้โดยตัวจำแนกประเภทแยกจากกัน หลังจากกระบวนการเรียนรู้เสร็จสิ้นก็จะถูกนำมารวมกันด้วยวิธีการใดวิธีการหนึ่ง

วิทยานิพนธ์ฉบับนี้มุ่งเน้นเฉพาะการแตกปัญหาโดยตรงจากโดเมนเริ่มต้นซึ่งยังสามารถแบ่งออกเป็นอีกสองประเภทย่อยคือการแบ่งโดยอาศัยการสุ่มเลือกของข้อมูล (Sample-based) และการแบ่งตามพื้นที่ข้อมูล (Space-based) ตัวอย่างการแบ่งประเภทแรกส่วนใหญ่เป็นโมเดลของกลุ่มตัวจำแนกประเภท (Ensemble) อาทิเช่น Bagging [33] วิธีการคือชุดข้อมูลที่ใช้ในการเรียนรู้จะถูกสุ่มตัวอย่างแบบใส่คืน (Sampling with replacement) เพื่อสร้างชุดข้อมูลออกเป็นหลาย ๆ ชุด แต่ละชุดข้อมูลนั้นก็จะถูกเรียนรู้โดยสมาชิกของกลุ่มตัวจำแนกประเภทแต่ละตัวแตกต่างกัน เอาท์พุตของระบบจะได้จากการโหวตเสียงส่วนมากหรือโหวตแบบถ่วงน้ำหนักจากตัวจำแนกประเภทที่เป็นสมาชิกของกลุ่มทั้งหมด Boosting [34] เป็นอีกโมเดลของกลุ่มตัวจำแนกประเภทที่ได้รับความนิยม กระบวนการจะอาศัยตัวจำแนกประเภทมากกว่าหนึ่งตัว โดยที่ข้อมูลตัวที่ถูกทำนายผิดจากตัวจำแนกประเภทก่อนหน้าจะถูกให้น้ำหนักมากกว่าข้อมูลที่ถูกทำนายถูก การให้น้ำหนักจะส่งผลถึงการสุ่มเลือกข้อมูลที่ใช้ในการเรียนรู้ของตัวจำแนกประเภทปัจจุบัน กล่าวคือตัวจำแนกประเภทตัวปัจจุบันพยายามเรียนรู้เฉพาะข้อมูลที่ถูกจำแนกผิดพลาดจากตัวจำแนกประเภทก่อนหน้า

ประเภทที่สอง การแบ่งประเภทข้อมูลโดยแบ่งตามพื้นที่ข้อมูลคือการแบ่งข้อมูลออกเป็นพื้นที่ต่าง ๆ ข้อมูลแต่ละตัวก็จะขึ้นอยู่กับบางพื้นที่ ตัวอย่างเช่น Mixture of expert (ME) [35] จะทำการแตกพื้นที่ของอินพุตโดยอาศัยการถ่วงน้ำหนักของผู้เชี่ยวชาญ (ในงานวิจัยนี้หมายถึงนิวรอนเน็ตเวิร์ก) แต่ละระบบแตกต่างกัน เอาท์พุตของระบบเกิดจากการรวมคำตอบของผู้เชี่ยวชาญทุกตัวและมีผู้เชี่ยวชาญอีกระบบทำหน้าที่เป็นพิจารณาการถ่วงน้ำหนักเรียกว่าเกท

(Gate) นอกจากนั้นทีมงานวิจัยเดียวกันยังพัฒนา ME ต่อเป็น Hierarchical mixture of experts (HME) [36] โดยระบบสามารถที่จะการแบ่งวนซ้ำตามที่ต้องการได้



รูปที่ 3.1 ระบบทั่วไปในการใช้งานตัวจำแนกประเภทหลายตัว X_i คืออินพุตใด ๆ E_N คือตัวจำแนกประเภทใด ๆ Y คือเอาท์พุตของระบบ

กระบวนการแตกปัญหานั้นสามารถทำได้โดยตรงหรือทางอ้อมและมักจะใช้ตัวจำแนกประเภทร่วมกันหลายตัว ซึ่งโดยปกติทั่วไปแล้วจำเป็นที่จะต้องอาศัยเกทด้วย H. A. Abbass [37] แบ่งเกทในการใชตัวจำแนกประเภทร่วมกันหลายตัวออกเป็นสองประเภทดังรูปที่ 3.1

พรีเกท (Pre-gate) ทำหน้าที่เป็นตัวเราท์เตอร์(Router) หรือตัวกระจาย (Distributor) ส่งต่อข้อมูลไปยังสมาชิกของกลุ่มตัวจำแนกประเภทด้วยเงื่อนไขที่กำหนด งานวิจัยที่เกี่ยวกับตัวพรีเกทโดยมากจะเกี่ยวข้องกับกระบวนการในการตัดสินใจว่าตัวจำแนกประเภทตัวใดควรเรียนรู้ข้อมูล [37],[38] ซึ่งสามารถสรุปได้ดังนี้

1. ตัวพรีเกททำหน้าที่ในการสุ่ม (Sub-Sampling) เป็นวิธีที่นิยมใช้กันมาก ชุดข้อมูลที่ใช้ในการเรียนรู้ โดยทำการแบ่งเป็นสับเซตย่อยหลาย ๆ อัน โดยส่วนใหญ่สับเซตย่อยที่สุ่มขึ้นมาอาจจะซ้อนทับกันหรือไม่ซ้อนทับกันก็ได้ วิธีนี้ที่นิยมคือ Bagging (Bootstrap Aggregating) [33] ซึ่งมีอัลกอริธึ่มในการทำงานดังรูปที่ 2 หลักการคือ สร้างชุดข้อมูลที่ใช้ในการเรียนรู้ขึ้นมาใหม่โดยมีขนาด m โดยการ sampling with replacement จากข้อมูลที่ใช้ในการเรียนรู้หลักที่มีขนาด z (z >> m) ซึ่งข้อมูลบางตัวจะปรากฏหลาย ๆ ชุดข้อมูลแต่ บางตัวปรากฏเพียงชุดเดียว หรือ บางตัวอาจจะไม่ปรากฏเลย อีกวิธีคือ AdaBoost (Adaptive Boosting) [39] จะทำการสร้าง weak learner ขึ้นมาจากข้อมูลที่ใช้ในการเรียนรู้ที่แตกต่างกัน ผลสุดท้ายได้จากการนำเอา weak learner หลาย ๆ ตัวมารวมกันเป็น strong classifier ขึ้นมา หลักการคือ ตัวอัล

กอริธึ่ม AdaBoost จะพยายามหาเซตค่าน้ำหนัก S' ให้กับเซตข้อมูลที่ใช้ในการเรียนรู้ โดยในแต่ละรอบเซตค่าน้ำหนักของอินพุต $S_1, S_2, \ldots S_T$ จะถูกสร้างขึ้นมาตามลำดับ ในตอนเริ่มต้นค่าน้ำหนักนี้จะมีค่าเท่ากันหมดสำหรับทุก ๆ ข้อมูล และค่าน้ำหนักของข้อมูลที่ถูกจำแนกผิดพลาดนั้นจะมีค่าเพิ่มขึ้น ส่วนค่าน้ำหนักของข้อมูลที่จำแนกได้ถูกต้องจะมีค่าลดลง

2. ตัวพรีเกททำหน้าที่กระจายข้อมูลที่ใช้ในการเรียนรู้ไปยังตัวจำแนกประเภทที่เป็นสมาชิกทั้งหมด แต่ตัวจำแนกประเภทแต่ละตัวจะได้คุณลักษณะของข้อมูลนั้นแตกต่างกันออกไป กล่าวค่อคุณลักษณะของข้อมูลจะถูกกระจายไปยังสมาชิกแต่ละตัวเพื่อให้เรียนรู้ในโดเมนที่แตกต่างกันไป (Manipulation of input features) ตัวอย่างเช่น ในงานวิจัยของ Cherkauer[40] สร้างระบบที่ใช้นิวรอนเน็ตเวิร์กทั้งหมด 32 ตัว แต่ละตัวจะเรียนรู้บนคุณลักษณะของข้อมูลที่แตกต่างกันไป
3. ตัวพรีเกททำหน้าที่แบ่งข้อมูลที่ใช้ในการเรียนรู้ตามคลาส โดยจะทำการแบ่งข้อมูลออกเป็นสับเซตย่อย โดยที่แต่ละสับเซตจะมีจำนวนคลาสน้อยกว่าเมื่อเปรียบเทียบกับข้อมูลทั้งหมด (Manipulation of output targets)
4. ตัวพรีเกททำหน้าที่แบ่งข้อมูลตามโดยอาศัยลักษณะแข่งกันเรียนรู้ (Competitive learning) โดยกลุ่มข้อมูลจะถูกจัดจัดกลุ่มตามคุณลักษณะ และตัวจำแนกประเภทแต่ละตัวจะทำการแข่งกันเพื่อที่จะเรียนรู้ข้อมูลที่จะเข้ามา โดยมากจะอาศัยการแบ่งข้อมูลหรือจัดกลุ่มข้อมูลเพื่อให้สมาชิกแต่ละตัวในกลุ่มเรียนรู้ข้อมูลแตกต่างกัน

โพสเกท (Post-gate) ทำหน้าที่เป็นตัวตัดสินคำตอบสุดท้ายที่ได้จากคำตอบของสมาชิกแต่ละตัวในกลุ่มตัวจำแนกประเภท โพลเกทส่วนใหญ่ก็สามารถแบ่งได้เป็น 2 แบบหลัก ๆ คือ

1. ถ้าสมาชิกแต่ละตัวรับผิดชอบแต่ล่ะสถานการณ์แตกต่างกันอย่างชัดเจนคำตอบสุดท้ายที่ได้ก็จะถูกตัดสินโดยสมาชิกตัวนั้น ๆ ส่วนใหญ่จะใช้คู่กับตัวพรีเกทแบบที่ 4
2. ถ้าสมาชิกทุกตัวมีส่วนร่วมในการตัดสินใจสำหรับอินพุตที่เข้ามาก็จะเกิดเหตุการณ์ขัดแย้งกันได้ ตัวโพสเกทก็จะทำหน้าที่รับผิดชอบตัดสินคำตอบสุดท้ายซึ่งส่วนใหญ่ก็จะเป็นการโหวตเสียงส่วนมาก โหวตแบบถ่วงน้ำหนักหรือโหวตแบบอื่น ๆ ในกรณีที่มีความซับซ้อนมากขึ้น [41, 42] ส่วนใหญ่จะใช้คู่กับตัวพรีเกทแบบที่ 1 2 และ 3

เมื่อไม่นานมานี้ Rokach และทีมงาน [43] นำเสนออัลกอริธึม K-classifier โดยใช้อัลกอริธึมการจัดกลุ่ม K-means เพื่อแบ่งข้อมูลออกเป็นกลุ่มที่แยกจากกันก่อน จากนั้นข้อมูลแต่ละกลุ่มจะถูกเรียนรู้โดยตัวจำแนกประเภทแต่ละตัวแยกจากกัน การแตกปัญหาในลักษณะนี้อาจจะมองอยู่ในรูปของการใช้งานตัวพรีเกทโดยที่ข้อมูลที่เข้ามาจะถูกส่งต่อไปยังตัวจำแนกประเภทที่เหมาะ สมโดยอาศัยการจัดกลุ่มของข้อมูล ผลการทดลองกับต้นไม้ตัดสินใจ (Decision tree)

นิวรอนเน็ตเวิร์ก และ Naive Bayes แสดงให้เห็นประสิทธิภาพในการจำแนกประเภทที่เพิ่มขึ้นในบางข้อมูลที่ใช้ทดสอบ แต่ในการทดลองไม่ได้วิเคราะห์เปรียบเทียบขนาดของตัวจำแนกประเภท (ในที่นี้หมายถึงจำนวนกฎที่เกิดขึ้นหรือจำนวนโหนดของนิวรอนเน็ตเวิร์ก) และเวลาที่ใช้ในการเรียนรู้ระหว่างการใช้ตัวจำแนกประเภทตัวเดียวกับการใช้ตัวจำแนกประเภทหลายตัว ข้อสังเกตอีกข้อคือกระบวนการแบ่งกลุ่มข้อมูลโดย K-mean นั้นแยกออกจากกระบวนการเรียนรู้ของตัวจำแนกประเภท ข้อมูลจะถูกจัดกลุ่มแบบออฟไลน์ทั้งหมดก่อนส่งต่อไปยังตัวจำแนกประเภท

วิทยานิพนธ์ฉบับนี้นำเสนอการเชื่อมต่อกันระหว่างกระบวนการแตกปัญหาและกระบวนการจำแนกประเภท โดยมีแนวคิดคือ ใช้ตัวพรีเกทในแบบที่ 4 ร่วมกับตัวโพสเกทแบบที่ 1 เพื่อใช้ในการจัดกลุ่มข้อมูลจัดกลุ่มข้อมูลออกเป็นกลุ่มต่าง ๆ ในกลุ่มแต่ละกลุ่มจะมีอัลกอริธึมที่ใช้ในการจำแนกประเภทข้อมูลเรียนรู้ข้อมูลของกลุ่มนั้น ๆ อยู่ ดังนั้นจึงทำให้ตัวจำแนกประเภทแต่ละตัวเรียนรู้ข้อมูลที่แตกต่างกัน โดยทั้งสองกระบวนการ (กระบวนการจัดกลุ่มและกระบวนการเรียนรู้ของตัวจำแนกประเภท) จะเกิดขึ้นต่อเนื่องกันในแต่ละอินพุตที่เข้ามา กล่าวคือเมื่อทำการจัดกลุ่มข้อมูลที่เข้ามาเสร็จสิ้นก็จะส่งต่อให้ตัวจำแนกประเภทเรียนรู้ข้อมูลนั้นทันที

การสร้างระบบลักษณะนี้ตัวจำแนกประเภทที่ใช้จำเป็นที่จะต้องเป็นตัวจำแนกประเภทเรียนรู้แบบออนไลน์ (Online machine learning) หมายความว่าโมเดลจะมีการปรับตัวให้เข้ากับอินพุตที่เข้าแบบทันที ตัวอย่างตัวจำแนกประเภทแบบออนไลน์เช่น LCSs หรือ นิวรอนเน็ตเวิร์ก ตัวจำแนกประเภทแบบออฟไลน์ (Offline machine learning) ซึ่งจำเป็นที่จะต้องประมวลผลข้อมูลทั้งหมดก่อนเพื่อที่จะหาเงื่อนไขตั้งต้นนั้นไม่สามารถนำมาใช้ได้ เช่น ตัวจำแนกประเภทต้นไม้ตัดสินใจ ID3 หรือ C4.5 จำเป็นที่จะต้องหาคุณลักษณะที่จะมาเป็นโหนดรากโดยวิเคราะห์จากข้อมูลทั้งหมดนั้นไม่สามารถนำมาใช้งานได้

การเชื่อมต่อระหว่างกระบวนการแตกปัญหาและกระบวนการจำแนกประเภทมีข้อดีหลายประการ ประการแรกถ้าเราแยกทั้งสองกระบวนการออกจากกันปัญหาจะถูกแตกก่อนที่จะมีการเรียนรู้ ลักษณะเช่นนี้จะทำให้ยากต่อการทำเหมืองข้อมูลแบบต่อเนื่อง (Stream data mining) ซึ่งลักษณะข้อมูลจะเข้ามาอย่างต่อเนื่องแบบเรียลไทม์ (Real time) และมีปริมาณมาก ประการที่สองข้อมูลเกิดการเปลี่ยนแปลงขึ้น (Concept Drift) เช่น พฤติกรรมของผู้บริโภคเกิดการเปลี่ยนแปลงหรือข้อมูลของสภาพแวดล้อมมีการเปลี่ยนแปลง ระบบที่แยกกระบวนการแตกปัญหาออกจากกระบวนการเรียนรู้จะต้องทำการแตกปัญหาใหม่ทั้งหมดและเริ่มกระบวนการเรียนรู้ใหม่

3.4 แนวทางการวิจัย

ในวิทยานิพนธ์ฉบับนี้เสนอการประยุกต์ใช้แนวคิดในการแตกปัญหาใหญ่เป็นปัญหาย่อยหลายปัญหา กับระบบจำแนกประเภทแบบเรียนรู้ได้ UCS โดยแทนที่จะใช้ระบบ UCS ขนาดใหญ่ที่มีจำนวนกฎมากเพื่อจัดการปัญหาทั้งหมด วิทยานิพนธ์ฉบับนี้นำเสนอระบบในการแตกปัญหา

โดยใช้เซลฟ์ออร์แกนไนซ์ซิ่งแม็พ (Self-Organizing Map –SOM) และโกรอิ่งนิวรอนแก๊ส (Growing Neural Gas – GNG) เป็นตัวพรีเกท (Pre-gate) เพื่อทำการแตกปัญหาเป็นปัญหาย่อย หลายปัญหาที่สามารถจำแนกประเภทได้โดยอาศัย UCS ที่ผูกติดอยู่กับโหนดแต่ละโหนดของทั้งเซลฟ์ออร์แกนไนซ์ซิ่งแม็พหรือโกรอิ่งนิวรอนแก๊ส ซึ่ง UCS แต่ละตัวจะมีขนาดเล็กใช้จำนวนกฎน้อยกว่าการใช้ UCS เพียงระบบเดียวจัดการกับปัญหาทั้งหมด

โดยผู้เขียนสันนิษฐานว่าการนำเซลฟ์ออร์แกนไนซ์ซิ่งแม็พหรือโกรอิ่งนิวรอนแก๊สมาใช้เป็นตัวพรีเกทสามารถเพิ่มประสิทธิภาพในการจำแนกประเภทข้อมูลได้ เนื่องจาก UCS แต่ละระบบจะจำแนกประเภทข้อมูลที่มีความซับซ้อนน้อยลง และยังสามารถลดเวลาในการเรียนรู้ได้ เนื่องจาก UCS แต่ละระบบใช้กฎน้อยกว่าการใช้ UCS เพียงระบบเดียว นอกจากนี้ผู้เขียนยังได้ทดลองเปลี่ยนตัวจำแนกประเภท UCS เป็นนิวรอนเน็ตเวิร์กเพื่อทดสอบประสิทธิภาพของระบบเมื่อนำไปใช้กับตัวจำแนกประเภทอื่นที่ไม่ใช่ LCSs

ในบทที่ 4 จะกล่าวถึงพื้นฐานอัลกอริธึมของจักรกลเรียนรู้แบบต่างทั้งตัวจำแนกประเภทและการจัดกลุ่มข้อมูลที่ใช้ในวิทยานิพนธ์ฉบับนี้

บทที่ 4

อัลกอริธึมการจำแนกประเภท และการจัดกลุ่ม

4.1 บทนำ

ในบทนี้จะกล่าวถึงอัลกอริธึมที่ใช้ในวิทยานิพนธ์ฉบับนี้ทั้งการจำแนกประเภทและการจัดกลุ่ม โดยในหัวข้อ 4.2 กล่าวถึงอัลกอริธึมที่ใช้การสร้างกฎที่ชื่อว่า ระบบจำแนกประเภทแบบเรียนรู้ได้ (Learning Classifier Systems – LCSs) ซึ่งเป็นหนึ่งอัลกอริธึมที่ใช้ในการจำแนกประเภท และเป็นระบบจำแนกประเภทหลักของวิทยานิพนธ์ฉบับนี้ หัวข้อที่ 4.3 จะกล่าวถึงนิวรอนเน็ตเวิร์กซึ่งเป็นระบบจำแนกประเภทที่นิยมใช้กันอย่างกว้างขวาง หัวข้อที่ 4.4 เซลฟ์ออร์แกนไนซ์ซิ่งแม็พ (Self-Organizing Map –SOM) และหัวข้อ 4.5 กล่าวถึงโกรอิ่งนิวรอนแก๊ส (Growing Neural Gas – GNG) ซึ่งเป็นสองอัลกอริธึมที่ใช้ในการจัดกลุ่มที่สำคัญ

4.2 ระบบจำแนกประเภทแบบเรียนรู้ได้

ในช่วงหลายสิบปีที่ผ่านมามีการนำเสนออัลกอริธึมสำหรับการจำแนกประเภทโดยอาศัยอัลกอริธึมในด้านจักรกลเรียนรู้มากมาย [4] ทั้งด้าน สถิติ (Statistic) โครงข่ายนิวรอน(Neural Network) และในด้านการเรียนรู้แบบวิวัฒนาการ(Evolution Learning) เจเนติกอัลกอริธึม (Genetic Algorithm - GA) [13] ก็เป็นอีกอัลกอริธึมหนึ่งของจักรกลเรียนรู้ที่กำลังได้รับความนิยมในงานวิจัยทางด้านการจำแนกประเภท เจเนติกอัลกอริธึมมีความสามารถที่จะสำรวจ (Explore) ขอบเขตของปัญหาขนาดใหญ่โดยอาศัยกระบวนการวิวัฒนาการทางธรรมชาติ ความสามารถที่น่าสนใจอีกอันหนึ่งคือความสามารถในการแทนความรู้ที่ได้มา ซึ่งเจเนติกสามารถแทนความรู้ได้หลายรูปแบบ แต่แบบที่นิยมคือ กฎ เนื่องจากเป็นรูปแบบที่มนุษย์สามารถทำความเข้าใจได้ง่าย

4.2.1 เจเนติกอัลกอริธึม

เจเนติกอัลกอริธึมเป็นอัลกอริธึมที่อาศัยกระบวนการวิวัฒนาการจากสิ่งมีชีวิตเป็นต้นแบบ เพื่อใช้ในการค้นหาคำตอบที่คาดว่าเป็นคำตอบดีที่สุด คำตอบของปัญหาจะถูกเข้ารหัส (Encoding) อยู่ในรูปโครโมโซม (Chromosomes) ซึ่งโครโมโซมอาจจะมีมากกว่าหนึ่งอันได้ เราเรียกกลุ่มของโครโมโซมหลาย ๆ ตัวว่าประชากร (population) โดยอาศัยทฤษฎีการอยู่รอดโดยอาศัยค่าความเหมาะสม (Fitness) เป็นหัวใจสำคัญในการวิวัฒนาการของโครโมโซม จุดประสงค์เพื่อที่จะรักษาโครโมโซมที่ดีให้อยู่รอดในกระบวนการวิวัฒนาการ ในการประยุกต์ใช้งานเราจำเป็นที่จะต้องมีการเข้ารหัสปัญหาที่ดี และมีฟังก์ชั่นที่ใช้วัดค่าความเหมาะสมของโครโมโซมที่

เหมาะสมเพื่อที่จะให้คำตอบของปัญหาที่ดีนั้นถูกเลือกเพื่อเข้าสู่กระบวนการสร้างคำตอบของปัญหาใหม่แทนที่คำตอบเดิมที่มีอยู่

เจเนติกต่างจากวิธีการค้นหาแบบอื่น ๆ คือ ประการแรกเจเนติกใช้การเข้ารหัสของคำตอบแทนที่จะใช้คำตอบในการค้นหาโดยตรง ประการที่สองเจเนติกค้นหาจากประชากร (คำตอบหลาย ๆ ตัว) แทนที่จะค่อย ๆ หาทีละคำตอบ ประการที่สามเจเนติกอาศัยฟังก์ชั่นมุ่งหมาย (Objective function) ประการสุดท้าย เจเนติกอาศัยกฎของความน่าจะเป็นแทนที่กระบวนการของดีเทอมินิสติก (Deterministic) เหมือนการค้นหาบางลักษณะ

เจเนติกในยุคแรกที่ Holland เป็นผู้นำเสนอและที่จะกล่าวถึงในหัวข้อนี้เป็นเจเนติกแบบง่าย (Simple Genetic) ที่เป็นรากฐานสำหรับแตกยอดงานวิจัยในการปรับปรุงเจเนติกไปอย่างมากมายในยุคปัจจุบัน ซึ่งมีกระบวนการทำงานดังรูปที่ 4.1

```
Simple Genetic Algorithm()
{
    1. Initial population;
    while termination criterion not reached
    {
        2. Evaluate individuals in the population;
        3. Select parents for recombination based on their fitness;
        4. Apply genetic operators to genetic offspring;
        5. Form a new population by offspring with partial/whole current population;
    }
}
```

รูปที่ 4.1 กระบวนการทำงานพื้นฐานของเจเนติกอัลกอริธึม

โดยพื้นฐานก่อนเริ่มกระบวนการเจเนติกจะต้องมีการสร้างประชากรเริ่มต้นด้วยการเข้ารหัสตัวแปรต่าง ๆ ที่จะหาคำตอบ โดยการเข้ารหัสนั้นจะขึ้นอยู่กับลักษณะของปัญหา แต่ที่สำคัญคือบิตสตริงหรือโครโมโซมหนึ่งตัวจะต้องแปลงไปเป็นคำตอบได้เพียงอันเดียวที่ไม่ซ้ำกับคำตอบอื่น

หลังจากนั้นจะเข้าสู่กระบวนการวัดค่าความเหมาะสมสำหรับแต่ละโครโมโซมเพื่อใช้เป็นค่าวัดในกระบวนการคัดสรรโครโมโซม ฟังก์ชั่นวัดค่าความเหมาะสมนั้นจะขึ้นอยู่กับจุดมุ่งหมายของเราว่าเราต้องการอยากได้อะไร เช่นถ้าการหาค่าสูงสุดของสมการ เราก็อาจจะใช้สมการนั้นมาเป็นฟังก์ชั่นวัดค่าความเหมาะสมได้ หรือ ถ้าเป็นการจำแนกประเภทข้อมูลเราก็อาจจะวัดจากความถูกต้องในการจำแนกข้อมูลของโครโมโซมนั้น

กระบวนการคัดสรรโครโมโซมนั้นมุ่งเน้นไปที่ภาพรวมของทั้งประชากรมากกว่าพิจารณาเป็นรายโครโมโซมเพื่อให้ได้ชุดของคำตอบที่ถูกต้องมากที่สุด โดยจะอาศัยหลักการคัดสรรตามธรรมชาติที่กล่าวว่า ผู้ที่แข็งแกร่งกว่าย่อมมีโอกาสอยู่รอดและดำรงเผ่าพันธุ์ได้มากกว่าผู้ที่อ่อนแอ เช่นเดียวกันกับกระบวนการเจเนติก โครโมโซมที่มีค่าความเหมาะสมมากกว่าก็จะมี

โอกาสได้รับการคัดสรรให้อยู่รอด และมีโอกาสที่จะสร้างโครโมโซมลูกเพื่อสืบทอดไปยังรุ่นต่อไปได้มากกว่าโครโมโซมที่มีค่าความเหมาะสมต่ำกว่า วิธีการที่นิยมคือการคัดสรรแบบรูเรทวิลล์ (Roulette Wheel Selection)

โครโมโซมแต่ละคู่ที่ถูกคัดสรรขึ้นมาจะถูกนำมาใช้เป็นพ่อและแม่พันธุ์ (Parent) ในการสร้างโครโมโซมลูกขึ้นมาใหม่ ซึ่งขั้นตอนเริ่มจากกระบวนการที่เรียกว่าการแลกเปลี่ยนยีนส์ระหว่างพ่อแม่ซึ่งหวังว่าจะเป็นส่งถ่ายโครงสร้างที่ดีของพ่อและแม่ไปยังลูก หลังจากนั้นจะผ่านกระบวนการที่เรียกว่าการผ่าเหล่า (Mutation) โดยเป็นการเปลี่ยนแปลงโครโมโซมบางส่วนเพื่อสร้างความแตกต่างให้กับลูก

ขั้นตอนสุดท้ายเป็นการสร้างประชากรชุดใหม่ขึ้นมา ในกรณีที่ง่ายที่สุดคือการสร้างประชากรชุดใหม่ขึ้นมาแทนที่ชุดเก่าทั้งหมด แต่ในกรณีของ Steady-state เราจะเก็บบางส่วนของประชากรรุ่นก่อนหน้าไว้และแทนที่ด้วยประชากรรุ่นใหม่บางส่วน ซึ่งการแทนที่นั้นก็จะนำไปแทนโครโมโซมใดนั้นอาศัยหลักความน่าจะเป็นโดยขึ้นอยู่กับค่าความเหมาะสม

ข้อดีของเจเนติกอีกประการคือ เจเนติกต้องการความรู้เบื้องต้น (Prior-knowledge) น้อยมาก เมื่อเทียบกับวิธีการอื่น ๆ อย่างไรก็ตามเจเนติกเองนั้นก็มีข้อด้อยอยู่คือ กระบวนการวิวัฒนาการนั้นค่อนข้างช้า ยิ่งปัญหามีความซับซ้อนจำนวนประชากรมากยิ่งต้องใช้เวลานาน ข้อด้อยอีกประการคือเจเนติกเป็นกระบวนการที่อาศัยความน่าจะเป็นเป็นหลักทั้งกระบวนการคัดสรร การผ่าเหล่า และการแลกเปลี่ยนยีนส์ ดังนั้นจึงจำเป็นที่จะต้องอาศัยการทดสอบหลาย ๆ ครั้งโดยกำหนดค่าเริ่มต้นให้แตกต่างกันไป (seeds)

4.2.2 ระบบจำแนกประเภทแบบเรียนรู้ได้

การประยุกต์เจเนติกใช้ในการวิวัฒนาการกฎสำหรับการจำแนกประเภทนั้นถูกนำเสนอในช่วงปี 70 [13, 44] เรียกว่าระบบจำแนกประเภทแบบเรียนรู้ได้ (Learning Classifier Systems – LCS) ซึ่งเราสามารถแบ่งออกเป็น 2 ประเภทย่อยได้แก่ระบบแบบพีซเบิร์ก (Pittsburgh) [45, 46] และระบบแบบมิชิแกน (Michigan) [13, 47] ข้อแตกต่างที่สำคัญของทั้งสองระบบคือการเข้ารหัสของโครโมโซมแต่ละตัวในประชากร ระบบแบบพีซเบิร์กนั้นโครโมโซมแต่ล่ะอันคือเซตของกฎที่แทนวิธีการแก้ปัญหาทั้งหมด แต่ในระบบมิชิแกนโครโมโซมแต่ล่ะอันจะแทนแค่กฎเพียงกฎเดียวซึ่งเป็นเพียงส่วนหนึ่งของวิธีการแก้ปัญหา เมื่อนำระบบแบบมิชิแกนไปใช้จะต้องนำทุกโครโมโซมหรือทุกกฎไปใช้งาน แต่ในระบบแบบพีซเบิร์กจะนำเพียงโครโมโซมที่ดีที่สุดเพียงอันเดียวไปแก้ปัญหา ตัวอย่างของระบบแบบแบบพีซเบิร์กที่นิยมใช้ในปัจจุบันคือ GALE [48] และ GAssist [49] สำหรับระบบแบบมิชิแกนที่นิยมใช้คือ XCS [22] และ UCS [23]

งานวิจัยหลายฉบับได้ทำการทดลองเปรียบเทียบระบบทั้งสองกับข้อมูลทดสอบต่าง ๆ ที่นิยมใช้ในการจำแนกประเภท เช่น Bernodo-Mansilla et al. [19] ทำการเปรียบเทียบระบบ XCS กับ GALE จากการศึกษาและทดลองกับข้อมูลทดสอบ 16 ชนิด แสดงให้เห็นว่าค่าความถูกต้องที่

ได้จากระบบทั้งสองไม่มีความแตกต่างอย่างมีนัยสำคัญ แต่ระบบ GALE จะใช้เวลาในการเรียนรู้ข้อมูลนานกว่าระบบ XCS ในงานวิจัยของ Bacardit และ Butz [49] ทำการศึกษาเปรียบเทียบระบบ XCS และ GAssist ในการทำเหมืองข้อมูล พบว่าระบบทั้งสองเหมาะสมสำหรับการทำเหมืองข้อมูลและระบบทั้งสองให้ค่าความถูกต้องไม่แตกต่างกันอย่างมีนัยสำคัญ ในงานวิจัยของ Butz [22] ได้ให้ข้อสังเกตว่าระบบจำแนกประเภทแบบมิชิแกนนั้นเหมาะสมกับการเรียนรู้แบบออนไลน์ (online) และเรียลไทม์ (real time) เนื่องจากมีการปรับค่าพารามิเตอร์ต่าง ๆ แบบทันทีหลังจากมีข้อมูลผ่านเข้ามา ในขณะที่ระบบจำแนกประเภทแบบพิซเบิร์กปกติจะนิยมใช้งานกับระบบออฟไลน์ (off-line)

ในงานวิทยานิพนธ์ฉบับนี้เป็นการนำเสนอระบบเพื่อปรับปรุงระบบจำแนกประเภทแบบเรียนรู้ได้แบบมิชิแกน ซึ่งจะนำเสนอในหัวข้อถัดไป โดยจะละเว้นการนำเสนอรายละเอียดของระบบแบบพีซเบิร์กไว้ หากผู้อ่านสนใจสามารถศึกษารายละเอียดได้ที่ [48] และ[49]

4.2.3 ระบบจำแนกประเภทแบบเรียนรู้ได้แบบมิชิแกน

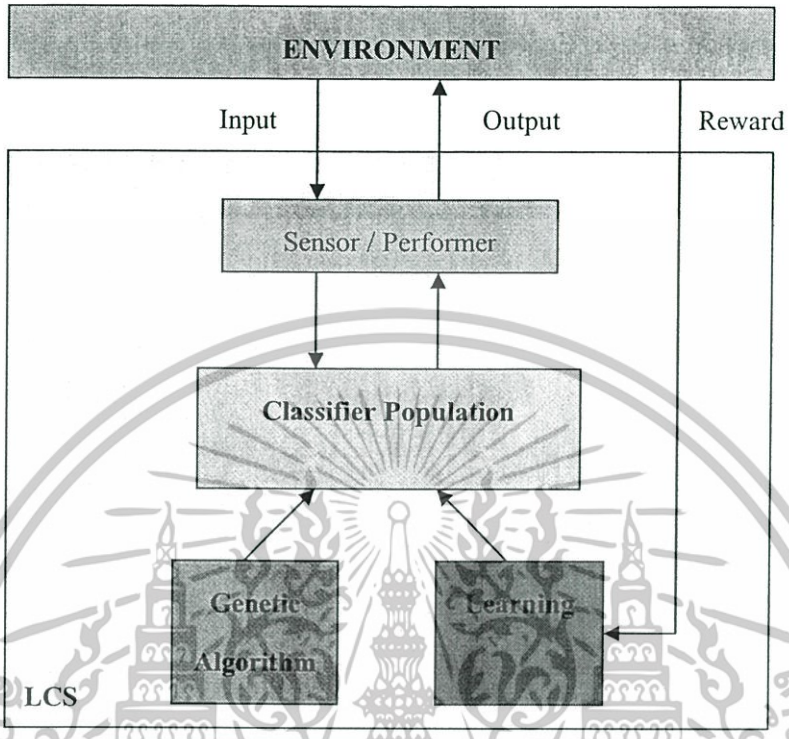
ระบบจำแนกประเภทแบบเรียนรู้ได้แบบมิชิแกนประกอบไปด้วยตัวจำแนกประเภทหรือกฎหลายตัว (Classifiers) ซึ่งตัวจำแนกประเภทเหล่านี้รวมกันเรียกว่าประชากร (Population) ในการนำระบบแบบมิชิแกนไปใช้งานจำเป็นจะต้องใช้ทั้งหมดของประชากรใช้เพียงส่วนใดส่วนหนึ่งของประชากรไม่ได้ ตัวจำแนกประเภทตัวหนึ่งจะประกอบไปด้วยสองส่วนสำคัญ ส่วนแรกคือส่วนของกฎและส่วนที่สองคือพารามิเตอร์ต่างๆประจำกฎ เช่น พารามิเตอร์ฟิตเนสใช้บ่งบอกว่าตัวจำแนกประเภทตัวนั้นถูกต้องมากน้อยเพียงใด (ในบางกรณีเราอาจเรียกตัวจำแนกประเภทว่ากฎเพื่อเน้นเฉพาะส่วนที่เป็นกฎเท่านั้น)

ตัวกฎจะประกอบไปด้วยส่วนกระตุ้นและส่วนตอบสนอง (เงื่อนไข-การกระทำ หรือ condition-action) กฎเหล่านี้จะอยู่ในรูปแบบของสัญลักษณ์ที่มนุษย์สามารถแปลความหมายได้ โดยปกติแล้วส่วนของเงื่อนไขจะถูกเข้ารหัสโดยใช้สัญลักษณ์ {0,1,#} ในส่วนตอบสนองจะถูกเข้ารหัสอยู่ในรูป {0,1} สัญลักษณ์ # หมายถึงสัญลักษณ์ที่แทนได้ทั้ง ‘0’ และ ‘1’ (Generalization) หรือบางทีเราอาจจะเรียกว่า don’t care ซึ่งจะเข้ากันได้กับสถานะทุก ๆ อย่างที่เข้ามาจากสภาวะแวดล้อม โครงสร้างของกฎจะอยู่ในรูปแบบดังนี้

<เงื่อนไข> : <การกระทำ>

สมมุติในระบบรับอินพุตจากสภาพแวดล้อม 4 อินพุต แต่ละอินพุตแทนด้วยค่า ‘0’ หรือ ‘1’ กฎ #011:1 จะเข้ากันได้กับทั้งอินพุต 0011 และ 1011 โดยที่จะกระทำแอคชั่น 1 เมื่อกฎนี้ถูกกระตุ้น

การทำงานของระบบจำแนกประเภทแบบเรียนรู้ได้แบบมิชิแกนอาศัยสองหลักใหญ่คือ กระบวนการวิวัฒนาการและกระบวนการเรียนรู้ ดังแสดงให้เห็นในรูป 4.2



รูปที่ 4.2 โครงสร้างภาพรวมของระบบจำแนกประเภทแบบเรียนรู้ได้

กระบวนการวิวัฒนาการจะทำหน้าที่ค้นหา วิวัฒนาการและสร้างกฎใหม่ให้กับประชากร โดยกระบวนการวิวัฒนาการจะเริ่มจากการคัดสรรกฎสองกฎ (พ่อและแม่) ซึ่งจะถูกเลือกจากประชากรทั้งหมด (population) หรือเฉพาะประชากรที่มีความเกี่ยวข้องกัน (niche – action set) หลังจากคัดสรรแล้วก็จะทำกระบวนการแลกเปลี่ยนโครโมโซมและสุ่มทำกระบวนการผ่าเหล่า เพื่อให้เกิดกฎลูกสองกฎ โดยกฎทั้งสองนั้นจะทำการสืบทอดค่าเฉลี่ยฟิตเนสจากพ่อและแม่ หลังจากนั้นก็จะถูกใส่กลับคืนไปในประชากรและทำการลบกฎออกจากประชากรสองกฎโดยอาศัยค่าฟิตเนสและการเลือกแบบรูเรทวิลล์

ส่วนกระบวนการเรียนรู้จะทำหน้าที่รับผิดชอบกระบวนการปรับปรุงค่าพารามิเตอร์ต่าง ๆ ขึ้นอยู่กับค่าตอบแทนที่ได้รับจากสภาพแวดล้อม ตัวจำแนกประเภทแต่ละตัวจะถูกปรับปรุงค่าพารามิเตอร์ต่าง ๆ ไปเรื่อย ๆ โดยค่าพารามิเตอร์เหล่านี้จะแสดงให้เห็นว่ากฎนั้นดีหรือไม่อย่างไร ค่าที่สำคัญคือค่าฟิตเนสซึ่งจะถูกนำมาพิจารณาว่าตัวจำแนกประเภทนั้นดีหรือไม่อย่างไร ตัวจำแนกประเภทที่มีค่าฟิตเนสต่ำก็จะมีโอกาสโดนลบออกไปจากประชากรผ่านกระบวนการวิวัฒนาการได้มาก

ดังนั้นกระบวนการเรียนรู้จึงเป็นส่วนนำทางกระบวนการวิวัฒนาการเพื่อให้การวิวัฒนาการนั้นมุ่งไปสู่การค้นหากฎที่คาดหวังว่าจะดีขึ้นเหมาะสมขึ้น กระบวนการสอง

กระบวนการนี้จึงเป็นหัวใจสำคัญระบบจำแนกประเภทแบบเรียนรู้ได้ โดยปกติกระบวนการวิวัฒนาการจะเหมือนกันในแต่ล่ะเวอร์ชั่นของระบบจำแนกประเภทแบบเรียนรู้ได้ แต่สิ่งที่ต่างกันคือกระบวนการเรียนรู้ ในหัวข้อถัดไปจะอธิบายถึงกระบวนการเรียนรู้สองชนิดที่นิยมใช้และชนิดที่สองเป็นกระบวนการเรียนรู้ที่ใช้ในวิทยานิพนธ์ฉบับนี้

4.2.3.1 กระบวนการเรียนรู้

กระบวนการเรียนรู้ที่นิยมในระบบจำแนกประเภทแบบเรียนรู้ได้มีอยู่สองแบบ แบบแรกเรียกว่าแบบรีอินฟอสเมนต์ (Reinforcement learning) แบบที่สองเรียกว่าแบบมีผู้สอน (Supervised learning)

XCS (eXtended Classifier System) [14, 15] เป็นระบบจำแนกประเภทแบบเรียนรู้ได้ที่ใช้กระบวนการเรียนรู้แบบแรกที่เป็นที่นิยม XCS สามารถใช้งานได้ทั้งการเรียนรู้แบบรีอินฟอสเมนต์ทั้งแบบขั้นตอนเดียว (single-step) และหลายขั้นตอน (multiple-step)

UCS (sUpervised Classifier System) [23] เป็นระบบจำแนกประเภทแบบเรียนรู้ได้ที่ใช้กระบวนการเรียนรู้แบบที่สอง ซึ่งพัฒนามาจาก XCS ถือได้ว่าเป็นเวอร์ชั่นพิเศษของ XCS ที่เน้นทำงานในด้านการจำแนกประเภท การคำนวณค่าความถูกต้องของตัวจำแนกประเภทจะต่างกับ XCS

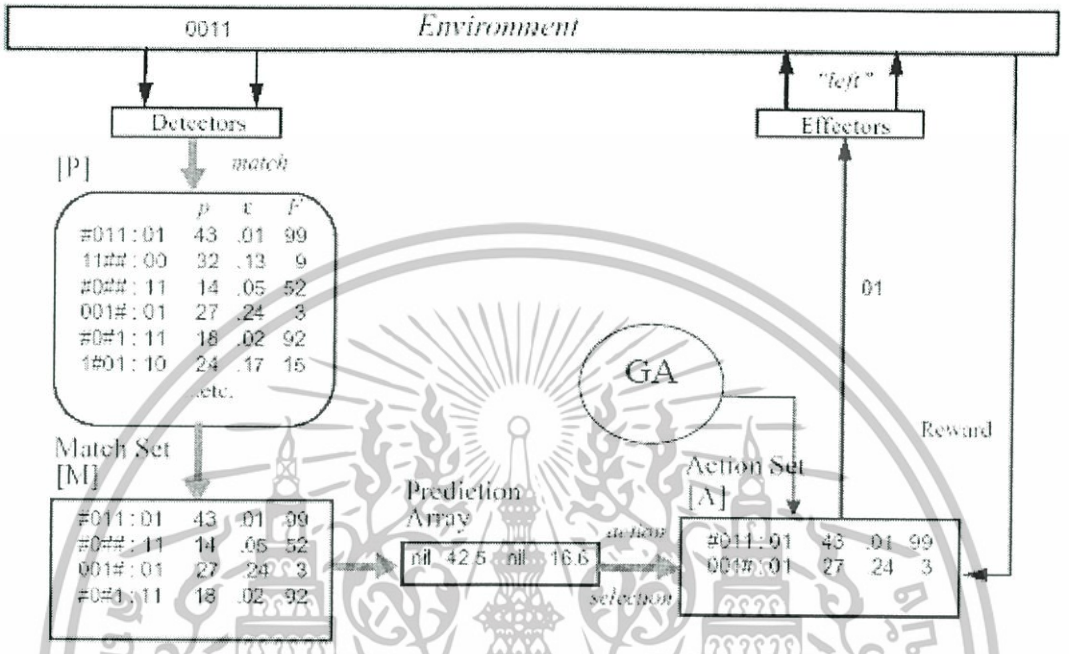
วิทยานิพนธ์ฉบับนี้มุ่งเน้นการพัฒนาทางด้านการจำแนกประเภทดังนั้นจึงใช้งาน UCS เป็นหลัก แต่อย่างไรก็ตามเนื่องจาก UCS เป็นเวอร์ชั่นพิเศษของ XCS ดังนั้น UCS จึงคงลักษณะส่วนใหญ่คล้ายกับ XCS ทั้งวิธีการคำนวณค่าความเหมาะสม (fitness) การใช้ GA เฉพาะกฎที่มีความเกี่ยวข้องกัน (niche-GA) และอื่น ๆ ดังนั้นจึงเป็นการเหมาะสมที่เราจะทำความเข้าใจกับระบบ XCS ก่อนเพื่อเพิ่มความเข้าใจในระบบ UCS

4.2.3.2 กระบวนการเรียนรู้แบบรีอินฟอสเมนต์ - XCS

พารามิเตอร์ในระบบ XCS จะมีสามพารามิเตอร์ที่สำคัญคือ ค่า p (Prediction) ซึ่งเป็นค่าประมาณของค่าตอบแทน (Reward) ที่จะได้รับจากสภาพแวดล้อม (ดูรูป 5.2 ประกอบ) ถ้ากฎนั้นถูกนำไปใช้งาน ค่า ε (Prediction error) ซึ่งเป็นค่าประมาณของความผิดพลาดในการทำนายค่าตอบแทน p และค่า F (Fitness) ซึ่งเป็นค่าวัดความถูกต้องของกฎนั้น โดยค่าฟิตเนสสามารถคำนวณโดยอาศัยค่า ε

ตัวจำแนกประเภทตัวหนึ่ง ๆ ใน XCS จะเป็นแบบแมคโคร (macro) หมายความว่า กฎที่อยู่ในประชากรจะเป็นกฎที่แตกต่างกันทั้งในส่วนของเงื่อนไขและการกระทำ เมื่อใดที่มีการสร้างตัวจำแนกประเภทตัวใหม่เกิดขึ้น ระบบจะทำการสแกนตรวจสอบตัวจำแนกประเภทในประชากรทั้งหมดถ้าพบว่ามีส่วนของกฎซ้ำกับกฎที่มีอยู่แล้วระบบจะไม่เพิ่มลงไปในประชากร แต่จะทำการบวกค่าเพิ่มหนึ่งให้กับพารามิเตอร์ที่มีชื่อว่า numerosity ของตัวจำแนกประเภทที่มีอยู่ก่อนแล้ว

แต่ถ้าไม่พบว่าซ้ำกับกฎที่มีอยู่แล้วก็จะทำการเพิ่มไปในประชากรพร้อมกับกำหนดค่า numerosity มีค่าเป็นหนึ่ง ค่าพารามิเตอร์ experience เป็นค่าบอกว่าตัวจำแนกประเภทนี้ปรากฏอยู่ในแมตช์เซต (Match set – [M]) มากน้อยเพียงใด ซึ่งจะเป็นตัวบอกความเป็นทั่วไป (Generality) ของกฎนี้



รูปที่ 4.3 โครงสร้างภาพรวมของระบบจำแนกประเภทแบบเรียนรู้ได้

รูปที่ 4.3 แสดงกระบวนการเรียนรู้ทั้งหมดของระบบ XCS เมื่อมีอินพุตเข้ามาในระบบ ระบบจะนำอินพุตมาสแกนประชากรทั้งหมดเพื่อหากฎที่เข้ากันได้ (Match) กับอินพุต (พิจารณาเฉพาะส่วนเงื่อนไข) สร้างเป็นแมตช์เซต [M] ดังนั้นใน [M] ตอนนี้จะมีส่วนของการกระทำ (action) อยู่หลากหลาย เราจำเป็นที่จะต้องเลือกการกระทำใดอันหนึ่งเพื่อกระทำต่อสภาพแวดล้อม วิธีการเลือกการกระทำนั้นระบบจะสร้างอะเรย์ [PA] ซึ่งมีขนาดเท่ากับการกระทำที่เป็นไปได้ทั้งหมดใน [M] หลังจากนั้นแต่ละการกระทำจะเฉลี่ยค่าฟิตเนสจากตัวจำแนกประเภทที่มีการกระทำเหมือนกันใน [M] การกระทำตัวไหนมีค่า PA มากที่สุดก็จะถือว่าถูกเลือกเพื่อส่งการกระทำนั้นไปสู่สภาพแวดล้อม หลังจากนั้นเซตของการกระทำ [A] จะถูกสร้างขึ้นจากตัวจำแนกประเภทใน [M] ที่มีส่วนของการกระทำเหมือนกับที่เลือกไว้

การปรับปรุงพารามิเตอร์ต่าง ๆ ในตัวจำแนกประเภทแต่ละตัวนั้นจะอาศัยแนวคิดในการปรับปรุงค่าคือการปรับค่าความถูกต้องในการทำนายค่าตอบแทนที่จะได้จากสภาพแวดล้อมหลังมีการใช้ตัวจำแนกประเภทนั้นและทำเฉพาะในขั้นตอน สำรวจ (explore) เท่านั้น โดยกระบวนการจะเริ่มหลังจากรับค่าตอบแทน (Reward) มาจากสภาพแวดล้อม XCS จะทำการปรับปรุงค่าประมาณค่าตอบแทน p ก่อน หลังจากนั้นเป็นค่า ε (Prediction error) และค่าฟิตเนส F เฉพาะตัวจำแนกประเภทที่อยู่ในแอคชั่นเซต [A] โดยจะมีสมการการปรับค่าดังนี้

$$p \leftarrow p + \beta(R - p) \tag{4.1}$$

โดยที่ $\beta(0 < \beta \leq 1)$ คืออัตราการเรียนรู้ (Learning rate) และ R เป็นค่าตอบแทนที่ได้รับจากสภาพแวดล้อม ถัดจากนั้นจะทำการปรับปรุงค่า ε ซึ่งเป็นค่าความผิดพลาดในการทำนายค่า p ดังนี้

$$\varepsilon \leftarrow \varepsilon + \beta(|R - p| - \varepsilon) \tag{4.2}$$

ในการปรับปรุงค่าฟิตเนส F เริ่มจากหาค่าความถูกต้องของกฎเรียกว่าค่า K หลังจากได้ค่า K จะนำมาหาค่าความถูกต้องสัมพันธ์ (Relative accuracy) K' ซึ่งทั้งสองสามารถคำนวณได้จากสมการดังนี้

$$K = \begin{cases} 1, & if(\varepsilon < \varepsilon_0) \\ \alpha\left(\frac{\varepsilon}{\varepsilon_0}\right)^{-\nu}, & otherwise \end{cases} \tag{4.3}$$

$$K' = \frac{K}{\sum_{x \in [A]} K_x} \tag{4.4}$$

พารามิเตอร์ ε_0 ($\varepsilon_0 > 0$) จะคอยควบคุมระดับการยอมรับในการประมาณค่าความผิดพลาดของกฎ ถ้าค่าความผิดพลาด ε ของกฎมีค่าน้อยกว่า ε_0 จะถือว่ายอมรับได้และกำหนดให้มีค่าความถูกต้อง K ของกฎนั้นเป็น 1 เลย ในส่วนของพารามิเตอร์ α ($0 < \alpha < 1$) และพารามิเตอร์ ν ($\nu > 0$) ทั้งสองเป็นค่าคงที่ที่คอยควบคุมค่า K (ในกรณีที่ค่าความผิดพลาด ε เกิน ε_0) ถ้าค่าความผิดพลาด ε ของกฎนั้นมีค่ามากจะส่งผลให้ค่าความถูกต้อง K ยิ่งน้อยลง

หลังจากนั้นค่าความถูกต้อง K ของกฎที่อยู่ในแอคชั่นเซต [A] จะถูกเปลี่ยนเป็นค่าความถูกต้องสัมพันธ์ (relative accuracy) K' ตามสมการที่ 4.5 หลังจากได้ค่า K' แล้วก็จะทำการปรับปรุงค่าฟิตเนส F ดังนี้

$$F \leftarrow F + \beta(\kappa' - F) \tag{4.5}$$

กล่าวโดยสรุปคือ ระบบ XCS ค่าฟิตเนสของกฎจะถูกปรับตามค่าความถูกต้องที่สัมพันธ์กับกฎอื่น ๆ ในแอคชั่นเซต [A]

หลังจากกระบวนการปรับปรุงค่าเสร็จสิ้น XCS จะทำการวิวัฒนาการกฎอาศัย GA โดยจะวิวัฒนาการเฉพาะกฎที่มีความเกี่ยวข้องกัน นั้นคือจะทำ GA เฉพาะในแอคชั่นเซต [A] เท่านั้นซึ่ง Wilson [15] เป็นคนนำเสนอแนวโดยกล่าวว่าการทำ GA ใน [A] พ่อและแม่จะไม่ต่างกันมาก

เกินไปและมีโอกาสให้โครโมโซมลูกที่ดีกว่า กระบวนการคัดสรรจะเลือกตัวจำแนกประเภท 2 ตัว จาก [A] ซึ่งใช้เป็นพ่อและแม่ วิธีการเลือกก็จะอาศัยความน่าจะเป็นตามสัดส่วนของค่าฟิตเนส หลังจากนั้นจะทำการสร้างตัวจำแนกประเภทลูก 2 ตัวโดยอาศัยกระบวนการแลกเปลี่ยนโครโมโซมและกระบวนการผ่าเหล่าตามข้างต้น โดยตัวจำแนกประเภททั้งสองนั้นจะทำการสืบทอดค่าเฉลี่ยฟิตเนส จากพ่อและแม่ เมื่อเสร็จสิ้นแล้วจะยังไม่ทำการลบพ่อและแม่นั้นทันทีแต่จะถูกใส่กลับคืนไปในประชากรพร้อมกับลูก และจะตรวจสอบจำนวนประชากรที่มีทั้งหมดถ้ายังไม่เกินจำนวนที่กำหนดไว้ก็จะยังไม่ลบตัวจำแนกประเภทใด ๆ แต่ถ้าจำนวนประชากรเกินระบบจะทำการลบตัวจำแนกประเภทแบบสุ่มตามค่าฟิตเนสจากประชากรทั้งหมด

Wilson [15] และ Kovac [50] ได้วิเคราะห์และกล่าวว่า เทคนิคกระบวนการวิวัฒนาการกฎจากค่าฟิตเนสที่อาศัยค่าความถูกต้องในการทำนายค่า p และการวิวัฒนาการเฉพาะกฎที่มีความเกี่ยวข้องกัน (niche GA) หรือเฉพาะในแอคชั่นเซต [A] นั้น สามารถที่จะวิวัฒนาการตัวจำแนกประเภทที่มีความถูกต้อง (Accuracy) และมีความเป็นทั่วไปสูงสุด (Maximally general) ตัวอย่างของกฎที่มีความเป็นทั่วไปสูงสุดเช่น สำหรับอินพุต 000000, 000001, 000010, 000011, 000100, 000110, 000111 คือ 000###

4.2.3.3 กระบวนการเรียนรู้แบบมีผู้สอน - UCS

ระบบจำแนกประเภทแบบเรียนรู้ได้ UCS เป็นระบบจำแนกประเภทที่มุ่งเน้นวิธีการเรียนรู้แบบมีผู้สอน (Supervised learning) หมายความว่าอินพุตที่เข้ามายังระบบจะมีส่วนของคลาสหรือการกระทำที่ถูกต้องสำหรับอินพุตนั้นเข้ามาด้วย เมื่อระบบรับอินพุตพร้อมคลาสเข้ามาระบบจะทำการสร้างแมตช์เซต [M] เหมือนกับระบบ XCS หลังจากนั้นจะสร้างคอล์เลคเซต (Correct set – [C]) ซึ่งจะมีคุณสมบัติคล้ายกับ [A] คือระบบจะทำการเลือกเฉพาะกฎใน [M] ที่มีส่วนของการกระทำตรงกับคลาสของอินพุตที่เข้ามา ถ้าไม่สามารถหากฎที่แมตช์กับอินพุตและคลาส หรือ [C] เป็นเซตว่าง ระบบจะทำการสร้างกฎ (Covering) ที่แมตช์กับอินพุตและมีส่วนของการกระทำที่ตรงกับคลาสของอินพุต

ในการปรับค่าพารามิเตอร์สำหรับ UCS จะปรับพารามิเตอร์ของตัวจำแนกประเภทที่อยู่ในแมตช์เซต [M] ทุกตัวซึ่งต่างจาก XCS ที่ปรับเฉพาะตัวจำแนกประเภทที่อยู่ในแอคชั่นเซต [A] มีการปรับค่าฟิตเนสแบบของ UCS จะไม่ปรับค่าความถูกต้องที่สัมพันธ์กับกฎอื่นอีกเหมือนใน XCS ค่าฟิตเนสของตัวจำแนกประเภทใน UCS จะปรับโดยคิดจากค่าความถูกต้องของตัวมันเอง ซึ่งค่าความถูกต้องสามารถคำนวณได้จากค่าอัตราส่วนของจำนวนครั้งที่กฎนั้นมีส่วนการกระทำที่ตรงกับอินพุตหารด้วยจำนวนครั้งที่กฎนั้นเข้าไปยังแมตช์เซต [M] ดังนี้

$$acc = \frac{N_{correct}}{N_{matches}} \tag{4.6}$$

ค่าฟิตเนสสามารถปรับได้ดังนี้

$$F = (acc)^{\nu} \tag{4.7}$$

โดยที่ ν คือค่าคงที่ปกติจะมีค่าเท่ากับ 5 ค่าฟิตเนสที่ได้จะไม่มีการคิดเทียบกับค่าฟิตเนสของตัวจำแนกประเภทตัวอื่น ๆ

ในส่วนของการวิวัฒนาการของตัวจำแนกประเภทของ UCS จะคล้ายกับ XCS คือจะทำ GA เฉพาะในกฎที่มีความเกี่ยวข้องกันเพื่อให้ระบบสามารถที่จะวิวัฒนาการตัวจำแนกประเภทที่มีความถูกต้องและมีความเป็นทั่วไปสูงสุดเช่นเดียวกับ XCS การวิวัฒนาการจะเกิดในคอล์เลคชั่นเซต [C] โดยจะกระทำ GA ทุกครั้งเมื่อค่าเฉลี่ยของเวลาครั้งล่าสุดที่เกิด GA กับตัวจำแนกประเภทนั้นมีค่ามากกว่าค่าที่กำหนดโดยผู้ใช้ กฎสองตัวใน [C] จะถูกเลือกเป็นพ่อและแม่โดยจะอาศัยค่าความน่าจะเป็นตามค่าฟิตเนส หลังจากนั้นก็จะสร้างกฎลูกขึ้นมาสองตัวผ่านกระบวนการแลกเปลี่ยนโครโมโซมและกระบวนการผ่าเหล่า หลังจากนั้นจะถูกใส่เข้าไปในประชากรโดยก่อนใส่จะตรวจสอบว่ากฎนั้นเป็นกฎที่ซ้ำซ้อน (subsumed) กับพ่อและแม่หรือไม่ ถ้าซ้ำซ้อนก็จะเพิ่มค่าหนึ่งให้กับค่า numerosity ของพ่อหรือแม่ แต่ถ้าไม่ซ้ำก็จะใส่เข้าไปในประชากรแล้วทำการตรวจสอบว่าประชากรเกินหรือไม่ ถ้าเกินก็จะทำการสุ่มลบตามค่าฟิตเนสจากประชากรทั้งหมด

สังเกตว่ากระบวนการวิวัฒนาการและกระบวนการลบตัวจำแนกประเภท UCS จะเหมือนกับ XCS แต่จะทำงานบนคอล์เลคชั่นเซต [C] แทนแอคชั่นเซต [A] เท่านั้น Bernado-Mansilla และ Garrell-Guiu [23] ได้ทำการทดลองกับข้อมูลจริง (Real data) แสดงให้เห็นว่า UCS นั้นให้ค่าความถูกต้องเทียบเท่ากับ XCS และในการทดลองกับข้อมูลเทียม (Artificial data) UCS ลู่เข้าได้เร็วกว่าและยังมีจำนวนประชากรน้อยกว่า XCS

4.2.3.4 ความแตกต่างระหว่าง XCS และ UCS

ดั้งเดิม XCS มีจุดประสงค์เพื่อให้ระบบทำการพัฒนากฎให้ครอบคลุม (Complete) และถูกต้อง (Accurate) กับขอบเขตของปัญหา (Problem space) ทั้งหมดโดยอาศัยความเป็นทั่วไป (Generalization) ของกฎ การหากฎที่ถูกต้องนั้นทำได้เนื่องจากกระบวนการวิวัฒนาการของ XCS อาศัยค่าฟิตเนสของกฎที่มาจากความถูกต้องในการทำนายค่า p เป็นพื้นฐาน ดังนั้นตัวจำแนกประเภทในประชากรจะมีทั้งกฎที่บ่งบอกว่าการกระทำนี้ถูกและกฎที่บ่งบอกว่าการกระทำนี้ไม่ถูก กฎที่บ่งบอกว่าการกระทำนี้ถูกคือกฎที่เมื่อถูกนำไปใช้จะให้ค่าตอบแทนสูงที่สุด (ปกติมีค่าเท่ากับ 1000) และกฎที่บ่งบอกว่าการกระทำนี้ไม่ถูกคือกฎที่เมื่อนำไปใช้แล้วจะให้ค่าตอบแทนต่ำที่สุด (ปกติมีค่าเท่ากับ 0) นั้นคือระบบ XCS จะสามารถบอกได้ว่าถ้าอินพุตเข้ามาแล้วการกระทำแบบนี้ถูก และการกระทำอีกแบบผิดได้อย่างถูกต้อง

ระบบ UCS จะเน้นเฉพาะกฎที่ให้ค่าการกระทำดีที่สุดเท่านั้น ค่าฟิตเนสของกฎจะขึ้นอยู่กับค่าความถูกต้องในการกระทำของกฎนั้นจริง ๆ ดังนั้นกฎที่อยู่รอดจะเป็นกฎที่บ่งบอกว่าการ

กระทำนี้ถูกและให้ค่าตอบแทนที่สูงเท่านั้น ส่วนกฎที่บ่งบอกว่าการกระทำนี้ไม่ถูกต้องจะมีค่าฟิตเนสต่ำ และมีโอกาสในการโดนลบออกไปจากประชากรได้สูงกว่า

โดยปกติการจำแนกประเภทจัดเป็นปัญหาแบบขั้นตอนเดียวที่มีการกำหนดค่าตอบแทนเพียงสองค่าคือค่าสูงสุดในกรณีตอบถูกและค่าต่ำสุดในกรณีตอบผิดเท่านั้น ดังนั้นลักษณะการวิวัฒนาการกฎของ UCS ซึ่งจะวิวัฒนาการเฉพาะกฎที่บ่งบอกว่าการกระทำนี้ถูกต้อง น่าจะเหมาะสมกว่า การวิวัฒนาการกฎของ XCS ซึ่งจะวิวัฒนาการทั้งกฎที่บ่งบอกว่าการกระทำนี้ถูกและกฎที่บ่งบอกว่าการกระทำนี้ไม่ถูกซึ่งกฎเหล่านี้จะไม่ได้ใช้ในงานประเภทการจำแนกประเภท Bernado-Mansilla และ Garrell-Guiu [23] ได้กล่าวถึงความแตกต่างระหว่างระบบ UCS และ XCS ไว้ดังนี้

- ในแง่ของจำนวนประชากร ถ้าเป็นงานทางด้านการทำเหมืองข้อมูลนั้นจำนวนคลาสของข้อมูลมีจำนวนมาก ระบบ XCS ย่อมสร้างกฎมากกว่าระบบ UCS เนื่องจาก XCS พยายามสร้างกฎที่ถูกและไม่ถูกให้ครอบคลุมทุกคลาส ดังนั้นจะเห็นได้ว่า UCS จะสร้างกฎน้อยกว่า XCS อย่างชัดเจน
- ในแง่ของการสำรวจ ระบบ XCS จำเป็นที่จะต้องสำรวจให้ครบทุกคลาส แต่ระบบ UCS มุ่งเน้นเฉพาะคลาสที่ถูกต้องเท่านั้น ดังนั้นการสำรวจของระบบ XCS จึงใช้เวลานานกว่าระบบ UCS
- ในแง่ของความซับซ้อน ระบบ XCS ให้จำนวนประชากรที่มากกว่าและต้องการจำนวนรอบในการเรียนรู้นานกว่า ดังนั้นจึงทำให้ระบบ XCS มีความซับซ้อนมากกว่าระบบ UCS ในงานด้านการจำแนกประเภท

ในหัวข้อ 5.2 นี้เราได้กล่าวถึงโครงสร้างและวิธีการเรียนรู้ระบบจำแนกประเภทแบบเรียนรู้ได้ทั้งแบบ XCS และ UCS ซึ่งในวิทยานิพนธ์นี้เราจะใช้ระบบจำแนกประเภทแบบเรียนรู้ได้แบบ UCS เป็นตัวเรียนรู้หลัก ปัญหาที่เด่นชัดของระบบ UCS คือจำนวนกฎที่ใช้ในการเรียนรู้ยังคงมีมากซึ่งจะส่งผลให้ใช้เวลาในการเรียนรู้นาน ซึ่งจะเป็นประเด็นที่จะนำเสนอในวิทยานิพนธ์นี้ในบทต่อไป ในหัวข้อถัดไปจะกล่าวถึงนิวรอนเน็ตเวิร์กซึ่งใช้เป็นตัวเรียนรู้อีกตัวที่อ้างอิงถึงในการทดลอง

4.3 นิวรอนเน็ตเวิร์ก

นิวรอนเน็ตเวิร์ก(Artificial Neural Network – ANN) เป็นอีกหนึ่งอัลกอริธึมในการจำแนกประเภทที่นิยมใช้กัน จากที่กล่าวมาในบทก่อนหน้าลักษณ์ความรู้ที่ได้ของนิวรอนเน็ตเวิร์กจะต่างจากระบบจำแนกประเภทแบบเรียนรู้ได้ที่มีลักษณะความรู้ที่ได้อยู่ในรูปแบบของกฎ แต่ลักษณะความรู้ที่ได้ของนิวรอนเน็ตเวิร์กจะอยู่ในรูปแบบของกล่องดำ (Black box) ซึ่งมนษย์ไม่

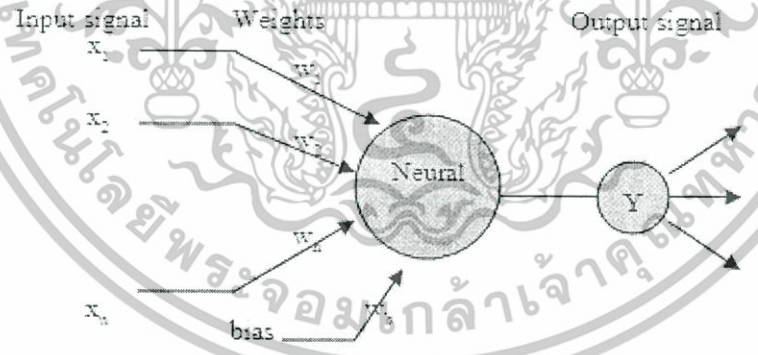
สามารถที่จะอ่านเข้าใจได้ แต่ถึงอย่างไรก็ตามความรู้ที่ได้ของนิวรอนเน็ตเวิร์กนั้นจะใช้เนื้อที่เก็บน้อยมาก

แนวคิดของนิวรอนเน็ตเวิร์กนั้นมาจากลักษณะการทำงานของสมอง ซึ่งถูกนำเสนอในช่วงต้นปีค.ศ. 1940 โดย McCulloch W.S และ Pitts W. [51] และถูกใช้งานกันอย่างแพร่หลายในช่วงปี 1980 Haykin [52] ได้แสดงให้เห็นถึงคุณสมบัติและคุณลักษณะหลายอย่างของนิวรอนเน็ตเวิร์กที่สามารถเรียนรู้ปัญหาซับซ้อน และสามารถปรับตัวเมื่อสภาพแวดล้อมปัญหามีการปรับตัว Zhang [53] ได้ทำงานวิจัยสำรวจที่เกี่ยวข้องกับเน็ตเวิร์กพบว่าถูกนำไปใช้งานมากมายทั้งทางด้าน การจดจำลายมือ การจดจำเสียง การตรวจสอบผลิตภัณฑ์ การทำนายข้อมูล การค้นหาข้อมูลที่ผิดพลาด งานด้านการแพทย์ และอื่น ๆ อีกมากมาย

ในวิทยานิพนธ์ฉบับนี้เราได้สร้างระบบเกี่ยวกับการแตกปัญหาใหญ่ให้เป็นปัญหาย่อย โดยมุ่งเน้นไปเพื่อแก้ปัญหาการเรียนรู้ของระบบจำแนกประเภทแบบเรียนรู้ได้ UCS เป็นหลัก แต่เพื่อทดสอบความยืดหยุ่นของระบบเราจึงทำการทดลองกับตัวจำแนกประเภทนิวรอนเน็ตเวิร์กด้วย ดังนั้นในหัวข้อนี้จะอธิบายถึงโครงสร้างและกระบวนการเรียนรู้อย่างคร่าว ๆ ของนิวรอนเน็ตเวิร์ก รายละเอียดสามารถอ่านได้ที่ [52]

4.3.1 โครงสร้างนิวรอนเน็ตเวิร์ก

นิวรอนเน็ตเวิร์กประกอบไปด้วยนิวรอนหลาย ๆ นิวรอนหรือที่เรียกว่าเพอเซฟตรอน (Perceptron)



รูปที่ 4.4 โครงสร้างเพอเซฟตรอน

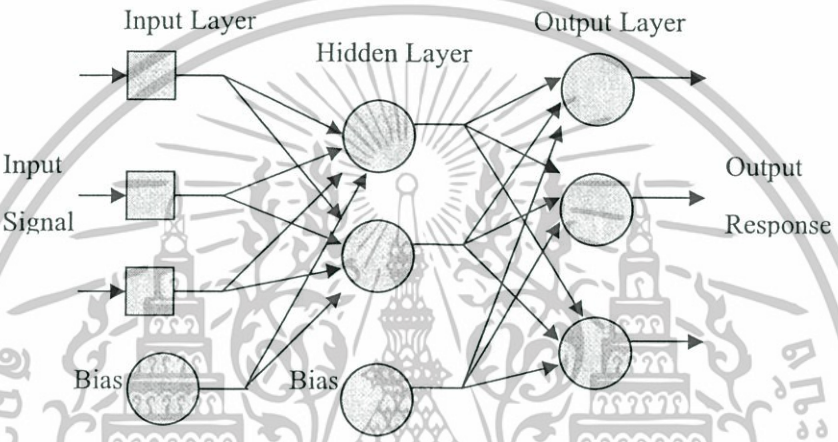
อินพุตที่เข้ามายังเพอเซฟตรอนจะมีทั้งอินพุตจากสิ่งแวดล้อมหรือเอาท์พุตของเพอเซฟตรอนตัวอื่น รูปที่ 4.4 แสดงโครงสร้างเพอเซฟตรอน เมื่อเพอเซฟตรอนได้รับอินพุตเข้ามาก็จะสร้างเอาท์พุตโดยอาศัยน้ำหนัก (Weight) และ แอคติเวชั่นฟังก์ชั่น (Activation function) ฟังก์ชั่นที่นิยมคือฟังก์ชั่นซิกมอยฟังก์ชั่น (Sigmoid function)

$$Y^{Sigmoid} = \frac{1}{1+e^{-X}} \tag{4.8}$$

และ

$$X = \sum_{i=1}^{N} x_i w_i \qquad (4.9)$$

โดยที่ x_i คืออินพุตลำดับที่ i w_i คือเวกเตอร์น้ำหนักที่ i N คือ จำนวนอินพุตทั้งหมด และ Y คือเอาท์พุตของนิวรอน ค่าไบแอส (Bias) คือปกติจะมีค่าเป็นหนึ่งเสมอ นิวรอนเน็ตเวิร์กโดยทั่วไปจะประกอบไปด้วยเพอเซพตรอนหลายตัวจัดเรียงตัวเป็นเลเยอร์อยู่ระหว่างอินพุตเลเยอร์และเอาท์พุตเลเยอร์เรียกว่า เลเยอร์ซ่อน (Hidden layer) ดังแสดงในรูป 4.5

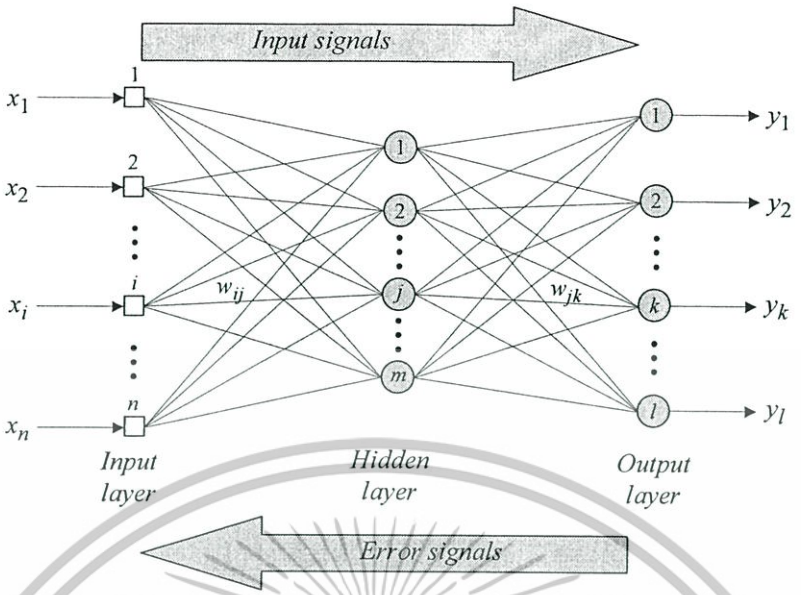


รูปที่ 4.5 โครงสร้างนิวรอนเน็ตเวิร์กแบบมีเลเยอร์ซ่อนหนึ่งเลเยอร์

จำนวนโหนดของอินพุตเลเยอร์จะมีเท่ากับจำนวนคุณลักษณะของข้อมูลและจำนวนโหนดของเอาท์พุตเลเยอร์จะมีจำนวนเท่ากับคลาสที่เป็นไปได้ทั้งหมดของข้อมูล จากรูปอินพุตมีสามคุณลักษณะและเอาท์พุตมีสามคลาส ชั้นซ่อนมีหนึ่งชั้นซ่อนและมีโหนดจำนวนสองโหนด ในการตัดสินว่าเป็นคลาสอะไรนั้นจะอาศัยความรู้ที่อยู่ในรูปเซตของน้ำหนักต่าง ๆ เป็นตัวตัดสินใจ เมื่ออินพุตเข้ามาก็จะส่งผ่าน (Feed-Forward) มายังชั้นซ่อนและไปยังชั้นเอาท์พุต การหาเอาท์พุตคลาสก็จะตัดสินจากเอาท์พุตที่มากที่สุดของนิวรอนในชั้นเอาท์พุต

4.3.2 กระบวนการเรียนรู้

กระบวนการเรียนรู้ในนิวรอนเน็ตเวิร์ก (กระบวนการปรับค่าน้ำหนัก) มีหลายวิธีแต่ที่นิยมคือการแพร่ย้อนกลับ [54] (Back-propagation) ดังรูปที่ 4.6



รูปที่ 4.6 กระบวนการเรียนรู้แบบแพร่ย้อนกลับ กรณีมีเลเยอร์ซ่อนเพียงชั้นเดียว

อินพุต $x_1, x_2, \ldots, x_n$ จะแพร่จากชั้นอินพุตไปยังชั้นเอาท์พุตและค่าความผิดพลาด (error) $e_1, e_2, \ldots, e_n$ จะแพร่จากชั้นอินพุตไปยังชั้นเอาท์พุต w_{ij} แทนค่าน้ำหนักประจำเส้นเชื่อมระหว่างอินพุตที่ i กับโหนดที่ j และ w_{jk} แทนค่าน้ำหนักประจำเส้นเชื่อมระหว่างโหนดที่ j กับเอาท์พุตที่ k

กระบวนการแพร่ไปข้างหน้าจากชั้นอินพุตไปยังชั้นเอาท์พุตจะอาศัยสมการ 4.8 และ 4.9 ส่วนกระบวนการแพร่ย้อนกลับจะเริ่มจากชั้นเอาท์พุตไปยังชั้นซ่อน สามารถหาได้จากสมการ

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p) \tag{4.10}$$

และ

$$e_k(p) = y_{d,k}(p) - y_k(p) \tag{4.11}$$

โดยที่ $y_k(p)$ คือเอาท์พุตที่ได้จากโหนด k ณ. รอบที่ p และ $y_{d,k}(p)$ คือเอาท์พุตที่ถูกต้องของโหนด k รอบที่ p

เมื่อได้ค่าความผิดพลาดจากชั้นเอาท์พุตจะทำการแพร่ไปยังชั้นซ่อนที่เหลือดังสมการ

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} (\delta_k(p) \times w_{jk}(p)) \tag{4.12}$$

โดยที่ l จำนวนโหนดในชั้นเอาท์พุต

$$y_j(p) = \frac{1}{1 + e^{-X_j(p)}} \tag{4.13}$$

และ

$$X_j(p) = \sum_{i=1}^{n} (x_i(p) \times w_{ij}(p)) - \theta_j \qquad (4.14)$$

โดยที่ n คือจำนวนนิวรอนในชั้นอินพุต จากสมการที่ 4.12 เราจึงสามารถที่จะแพร่ค่าความผิดพลาดมายังชั้นซ่อนได้

หลังจากนั้นเราจะทำการปรับค่าน้ำหนักที่เชื่อมระหว่างโหนดในชั้นต่าง ๆ โดยเริ่มจากค่าน้ำหนักที่เชื่อมระหว่างชั้นเอาท์พุตกับชั้นซ่อน สำหรับนิวรอนที่ k ในชั้นเอาท์พุตกับนิวรอนชั้นซ่อนที่ j ใด ๆ

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \qquad (4.15)$$

โดยที่ $\Delta w_{jk}(p)$ คือค่าน้ำหนักที่ปรับใหม่ซึ่งปรับได้จากอินพุตที่เข้ามายังโหนดนั้น ในกรณีนี้อินพุตสำหรับชั้นเอาท์พุตคือ ค่าเอาท์พุตของโหนดในชั้นซ่อน

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p) \qquad (4.16)$$

โดยที่ $\delta_k(p)$ คือค่าความผิดพลาดของนิวรอน k ในชั้นเอาท์พุต ณ รอบที่ p ดังสมการที่ (4.10) ในการปรับค่าน้ำหนักที่เชื่อมระหว่างชั้นซ่อนกับชั้นอินพุตสามารถคำนวณค่า $\Delta w_{ij}(p)$ สำหรับนิวรอนที่ j ในชั้นซ่อนกับอินพุต i ใด ๆ

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p) \qquad (4.17)$$

โดยที่ $\delta_j(p)$ คือค่าความผิดพลาดของนิวรอน j ในชั้นซ่อน ณ รอบที่ p ดังสมการที่ 4.17 หลังจากจบกระบวนการปรับค่าน้ำหนักแล้วก็ถือว่าเสร็จสิ้นกระบวนการเรียนรู้สำหรับอินพุตที่เข้ามา ณ รอบที่ p โดยปกติระบบจะทำการเรียนรู้ไปจนกระทั่งถึงเงื่อนไขหนึ่งซึ่งอาจจะเป็นจำนวนรอบที่กำหนดหรืออาจจะเรียนรู้ไปจนกระทั่งสามารถทำนายข้อมูลที่ใช้ในการทดสอบได้ถูกต้องถึงระดับที่กำหนด

4.4 เซลฟ์ออร์แกนไนท์ซิ่งแม็พ

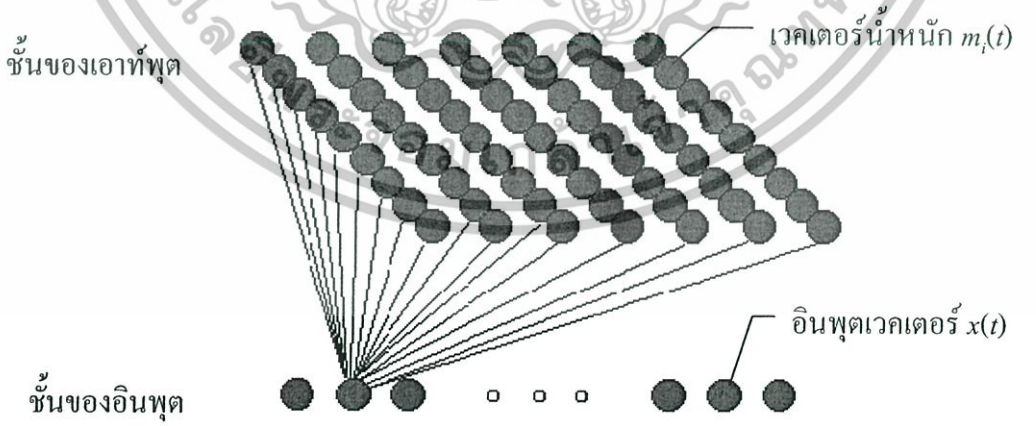
เซลฟ์ออร์แกนไนท์ซิ่งแม็พ (Self-Organizing Map) หรือ SOM เป็นนิวรอนเน็ตเวิร์กแบบไม่มีผู้สอนประเภทหนึ่ง [55] ซึ่งแตกต่างจากนิวรอนเน็ตเวิร์ก ANN ซึ่งเป็นนิวรอนเน็ตเวิร์กแบบมีผู้สอน อัลกอริทึมของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พถูกนำเสนอโดยศาสตราจารย์โคโฮเนน (Kohonen) ในปี ค.ศ. 1982 [56] ซึ่งรู้จักกันในชื่อ แผนภาพคุณลักษณะของโคโฮเนน (Kohonen feature map)

เซลฟ์ออร์แกนไนซ์ซิ่งแม็พเป็นอัลกอริทึมที่ใช้ในการจัดกลุ่มข้อมูล โดยสามารถจัดกลุ่มข้อมูลที่มีมิติสูงให้อยู่ในรูปของแผนภาพ 2 มิติซึ่งประกอบไปด้วยโหนดของนิวรอน ข้อมูลจะถูกจัดลงในโหนดต่าง ๆ ของแผนภาพ หลังจากเสร็จสิ้นกระบวนการเรียนรู้แผนภาพจะถูกจัดเรียงตัวโดยข้อมูลที่มีความคล้ายคลึงกันจะอยู่ในกลุ่มโหนดใกล้เคียงกัน ดังนั้นแผนภาพที่ได้จะแสดงคุณลักษณะของข้อมูล ได้เป็นอย่างดี เช่น การกระจายของข้อมูล ความสัมพันธ์ระหว่างข้อมูลในกลุ่มเป็นต้น

ในวิทยานิพนธ์ฉบับนี้เราได้นำเซลฟ์ออร์แกนไนซ์ซิ่งแม็พมาใช้เป็นส่วนประกอบของระบบในการแตกปัญหาใหญ่ให้เป็นปัญหาย่อย ซึ่งเซลฟ์ออร์แกนไนซ์ซิ่งแม็พจะถูกจัดวางเป็นชั้นอยู่ก่อนชั้นของระบบจำแนกประเภทแบบเรียนรู้ได้ UCS เมื่ออินพุตเข้ามาก็จะถูกเซลฟ์ออร์แกนไนซ์ซิ่งแม็พจัดกลุ่มข้อมูลและส่งต่อไปยัง UCS ที่เหมาะสมเพื่อใช้ในการเรียนรู้แยกจากกลุ่มข้อมูลอื่น หัวข้อนี้จะกล่าวถึงโครงสร้างและลักษณะการเรียนรู้ของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ

4.4.1 กระบวนการเรียนรู้

โมเดลของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พประกอบด้วยเซลล์ 2 ชั้น [55] ดังรูปที่ 4.7 ชั้นแรกคือชั้นของอินพุต(Input layer)ประกอบไปด้วยเซตของอินพุตเวคเตอร์ $x(t)$ ที่มีขนาด n มิติ(1 x n มิติ) ซึ่งเป็นอินพุตที่ใช้ในการเรียนรู้ของแผนภาพ โดยที่ t คืออินเด็กซ์ของอินพุตหรือแทน ณ เวลาใด ๆ ก็ได้ ชั้นที่สองคือชั้นของแผนภาพโคโฮเนน (Kohonen layer) หรือชั้นของเอาท์พุตประกอบไปด้วยโหนดของนิวรอนที่เรียงตัวอยู่ในรูปแบบของแผนภาพ 2 มิติ ในแต่ละโหนด i จะเป็นค่าเวคเตอร์น้ำหนักแทนด้วย $m_i(t)$ นั่นคือ $m_i(t) \in \Re^n$ โดยที่ $\Re^n$ คือโดเมนของขนาดของ n และขนาดของเวคเตอร์น้ำหนักจะต้องมีขนาดเท่ากับอินพุตเวคเตอร์ $x(t)$



รูปที่ 4.7 แสดงโมเดลพื้นฐานของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พแบบสี่เหลี่ยม [3]

[3] ที่มา www.icadl.org/intranet/papers/A_Scalable-98.htm

กระบวนการเรียนรู้ของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พเกิดขึ้นจากการปรับตัวของเวคเตอร์น้ำหนักที่มีต่ออินพุตเวคเตอร์ โดยเริ่มแรกจะทำกำหนดน้ำหนักเริ่มต้นขนาดเล็กให้กับโหนดทุกโหนด จากนั้นจะเริ่มต้นกระบวนการเรียนรู้ดังนี้

1. เลือกอินพุตเวคเตอร์แบบสุ่มเลือกจากอินพุตโดเมน
2. เปรียบเทียบอินพุตเวคเตอร์ $x(t)$ กับโหนด $m_i(t)$ ทุกโหนดเพื่อหาโหนดชนะจากโหนดทั้งหมด
3. ปรับเวคเตอร์น้ำหนักของโหนดชนะ เพื่อให้โหนดชนะเข้าใกล้อินพุตมากขึ้น
4. ปรับเวคเตอร์น้ำหนักของโหนดใกล้เคียง เพื่อให้อินพุตเวคเตอร์ถัดไปที่มีค่าใกล้เคียงมีโหนดชนะใหม่อยู่ใกล้กัน

กระบวนการเหล่านี้จะถูกทำซ้ำไปเรื่อย ๆ จนกว่าจะสอดคล้องตามเงื่อนไขหรือจนกว่าจะครบจำนวนรอบของการเรียนรู้ จากกระบวนการเรียนรู้ข้างต้นมีการคำนวณที่สำคัญอยู่ 2 ส่วนคือ ส่วนแรกคือการคำนวณเพื่อหาโหนดชนะ(ขั้นตอนที่ 2) ในการคำนวณหาโหนดชนะอินพุตเวคเตอร์ $x(t)$ ถูกนำไปเปรียบเทียบกับโหนด $m_i(t)$ ทุกโหนดเพื่อหาโหนดชนะจากโหนดทั้งหมด ฟังก์ชั่นที่ใช้ในการเปรียบเทียบโดยทั่วไปแล้วจะใช้ฟังก์ชั่นวัดระยะทางแบบยูคลิด (Euclidean distance)

การหาโหนดที่ชนะ c สามารถหาได้จากโหนดที่มีระยะห่างระหว่างอินพุตเวคเตอร์กับเวคเตอร์น้ำหนักของโหนดนั้นน้อยที่สุดดังสมการที่ 1.1

$$c : m_c(t) = \min_i \| x(t) - m_i(t) \| \tag{4.18}$$

ส่วนที่สองคือการปรับเวคเตอร์น้ำหนัก หลังจากที่ได้โหนดชนะแล้วจะต้องทำการปรับน้ำหนักเพื่อให้เข้าใกล้อินพุตมากขึ้น นอกจากการเรียนรู้ที่เกิดขึ้นที่โหนดชนะแล้ว โหนดใกล้เคียง จะเกิดการเรียนรู้ด้วย ค่าเวคเตอร์น้ำหนักของโหนดใกล้เคียงจะปรับค่าให้เข้าใกล้กับอินพุตเวคเตอร์เดียวกัน เพื่อเพิ่มโอกาสให้อินพุตใหม่ที่ใกล้เคียงกับอินพุตเดิมสามารถที่จะมีโหนดชนะใหม่ใกล้กับโหนดชนะเดิมได้ สมการในการปรับค่าน้ำหนักสามารถแสดงได้ดังสมการที่ 4.19

$$m_i(t+1) = m_i(t) + \alpha(t) \times h_{ci}(t) \times [x(t) - m_i(t)] \tag{4.19}$$

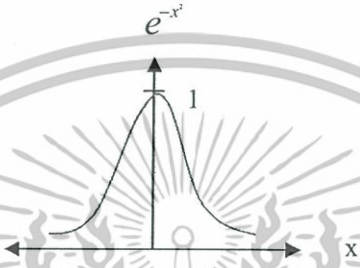
เมื่อt คือรอบปัจจุบันของการเรียนรู้ $x(t)$ คืออินพุตเวคเตอร์ปัจจุบัน$m_i(t)$ คือเวคเตอร์น้ำหนัก $\alpha(t)$ คืออัตราการเรียนรู้ โดยที่อัตราการเรียนรู้ $\alpha(t)$ จะขึ้นอยู่กับจำนวนรอบซึ่งแสดงเป็นสมการเชิงเส้นได้ดังสมการที่ 4.20

$$\alpha(t) = \alpha(0) \times \frac{T - t}{T} \tag{4.20}$$

เมื่อ T คือจำนวนรอบทั้งหมด t คือจำนวนรอบปัจจุบัน $h_{ci}(t)$ คือฟังก์ชั่นที่ใช้ในการกำหนดน้ำหนักในการปรับค่าโหนดใกล้เคียงโดยทั่วไปแล้ว $h_{ci}(t)$ จะใช้ฟังก์ชั่นเกาส์เซียน (Gaussian) ซึ่งสามารถเขียนได้ดังสมการที่ 4.21

$$h_{ci}(t) = \exp(-\frac{\| r_c - r_i \|^2}{2\sigma^2(t)}) \qquad (4.21)$$

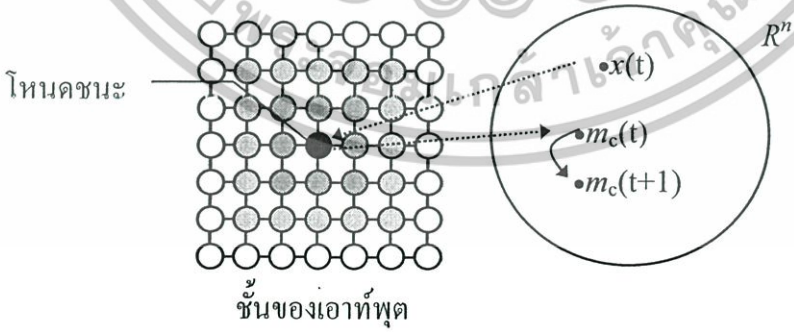
เมื่อ $\|r_c-r_i\|$ คือระยะห่างของตำแหน่งของโหนด i กับโหนดชนะ c และ $\sigma(t)$ คือรัศมีของบริเวณโหนดใกล้เคียง



รูปที่ 4.8 แสดงกราฟของฟังก์ชั่นเกาส์เซียน ($y = e^{-x}$)

ในรูปที่ 4.8 ลักษณะของฟังก์ชั่นเกาส์เซียนคือ เมื่อค่า x คือค่าระยะห่างมีค่ามาก ค่าที่ส่งกลับมาจากฟังก์ชั่นจะลดลงไปเรื่อย ๆ จนเข้าใกล้ศูนย์ ซึ่งสอดคล้องกับการปรับน้ำหนักของโหนดชนะและโหนดใกล้เคียง โหนดชนะจะมีค่า x เป็นศูนย์ซึ่งจะให้ค่าเกาส์เซียนฟังก์ชั่นออกมาเป็นหนึ่งซึ่งมากที่สุด โหนดที่ใกล้กับโหนดชนะจะมีการปรับค่าเวคเตอร์น้ำหนักมากกว่าโหนดที่อยู่ไกล โดยจะมีการกำหนดรัศมีของโหนดใกล้เคียงโดยปกติรัศมีของโหนดใกล้เคียงจะค่อย ๆ ลดลงตามจำนวนรอบในการเรียนรู้ t ดังสมการที่ 4.22

$$\sigma(t+1) = 1 + (\sigma(t) - 1) \times \frac{T - t}{T} \qquad (4.22)$$



รูปที่ 4.9 แสดงโครงสร้างของ SOM ขนาด 7x7

ในรูปที่ 4.8 แสดงเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 7x7 แบบสี่เหลี่ยมโหนดสีเข้มที่สุดคือโหนดชนะสำหรับอินพุตเวคเตอร์ $x(t)$ จากนั้นค่าเวคเตอร์น้ำหนักของโหนด $m_c(t)$ จะถูกปรับค่าให้เข้าใกล้กับอินพุตเวคเตอร์มากขึ้น หลังจากนั้นจะทำการปรับโหนดใกล้เคียงของโหนดชนะ

โดยความเข้มสีของโหนดจะแสดงถึงปริมาณการปรับค่าของเวคเตอร์น้ำหนัก โหนดที่มีสีเข้มมากจะมีการปรับค่าเวคเตอร์น้ำหนักมากกว่าสีเข้มน้อย

4.5 โกล์วอิ่งนิวรอนแก๊ส

นอกจากใช้เซลฟ์ออร์แกนไนท์ซิ่งแม็พในการจัดกลุ่มข้อมูลก่อนส่งต่อไปยังระบบเรียนรู้จำแนกประเภท UCS แล้วในวิทยานิพนธ์ฉบับนี้ยังนำเสนอทดลองใช้โกลว์อิ่งนิวรอนแก๊ส (Growing Neuron Gas –GNG) มาใช้ในการจัดกลุ่มข้อมูลแทนเซลฟ์ออร์แกนไนท์ซิ่งแม็พด้วย

โกลว์อิ่งนิวรอนแก๊สจัดว่าเป็นนิวรอนเน็ตเวิร์กแบบไม่มีผู้สอนประเภทหนึ่งถูกพัฒนาโดย Fritzke [57] ข้อดีที่ทำให้โกลว์อิ่งนิวรอนแก๊สมีจุดเด่นคือ โกลว์อิ่งนิวรอนแก๊สต้องการความรู้เบื้องต้นเกี่ยวกับข้อมูลและการตั้งค่าระบบเบื้องต้นน้อยมาก โกลว์อิ่งนิวรอนแก๊สมีพารามิเตอร์เริ่มต้นของระบบมีเพียงจำนวนรอบทั้งหมดในการเรียนรู้ นอกจากนี้ยังไม่ต้องกำหนดจำนวนโหนดทั้งหมดในขณะเริ่มต้น รวมถึงยังไม่ต้องกำหนดลักษณะการเชื่อมต่อของโหนดด้วย ซึ่งต่างจาก เซลฟ์ออร์แกนไนท์ซิ่งแม็พเราจำเป็นที่จะต้องกำหนดจำนวนโหนดและลักษณะการเชื่อมต่อระหว่างโหนดตั้งแต่เริ่มต้นเรียนรู้ หัวข้อนี้จะอธิบายถึงโครงสร้างและลักษณะการเรียนรู้ของโกลว์อิ่งนิวรอนแก๊ส

4.5.1 กระบวนการเรียนรู้

โครงสร้างของโกลว์อิ่งนิวรอนแก๊สจะเริ่มจากกราฟที่มีโหนดสองโหนด ถ้าโหนดใด ๆ เชื่อมต่อกันเราจะถือว่าเป็นโหนดใกล้เคียงกัน โหนดแต่ละโหนดจะมีค่าประจำโหนดสามค่าคือ

ค่าเวคเตอร์ประจำโหนดเรียกว่า $\overline{w_k}$ ซึ่งแทนตำแหน่งในโดเมนของอินพุต$\Re^n$ โดยที่ $\Re^n$ คือโดเมนของขนาดของ n และขนาดของเวคเตอร์น้ำหนักจะต้องมีขนาดเท่ากับอินพุตเวคเตอร์ $\bar{x}$

ค่าความผิดพลาดสะสม $error_k$ ซึ่งจะใช้ในการตัดสินใจในการแทรกโหนดใหม่ลงไปในกราฟ ค่าความผิดพลาดสะสมนี้จะทำหน้าที่บ่งบอกว่าโหนดนั้นครอบคลุมบริเวณของอินพุตใหญ่เกินไปหรือไม่ ในกรณีที่โหนดครอบคลุมบริเวณของอินพุตกว้างจะทำให้ค่าความผิดพลาดสะสมมีค่าสูง

เซตของแขน (Edge) ที่เชื่อมกับโหนด k ใด ๆ แขนแต่ละอันจะมีค่าอายุ (age) ประจำแขนนั้น ๆ ค่าอายุจะช่วยในการจัดโครงสร้างของระบบ โดยปกติอายุของแขนจะเพิ่มค่าขึ้นไปเรื่อย ๆ ถ้าโหนดที่ใกล้อินพุตที่สุดลำดับแรกไม่เชื่อมต่อกับโหนดที่ใกล้อินพุตที่สุดลำดับที่สอง แต่ถ้าโหนดทั้งสองเชื่อมต่อกันค่าของอายุจะถูกรีเซตเป็นศูนย์ ดังนั้นเมื่อแขนมีอายุมากถึงค่าหนึ่งนั้นแสดงว่าแขนนั้นอาจจะไม่ใช่โครงสร้างที่เหมาะสมและจะถูกลบออกและเมื่อโหนดใดไม่มีแขนเชื่อมต่อ (dead node) โหนดนั้นก็จะถูกยุบด้วย

กระบวนการเรียนรู้เริ่มจากการสุ่มสร้างสองโหนด [58] โดยที่ทั้งสองโหนดจะเชื่อมต่อกัน อายุของแขนและค่าความผิดพลาดในแต่ละโหนดจะถูกกำหนดเป็นศูนย์ กระบวนการเรียนรู้สามารถแสดงได้ดังนี้

1. เลือกอินพุตเวคเตอร์ $\bar{x}$ แบบสุ่มเลือกจากอินพุตโดเมน
2. หาโหนดที่ใกล้อินพุตสองโหนดกำหนดเป็นโหนด s และ t ซึ่งมีเวคเตอร์น้ำหนักเป็น $\overline{w_s}$ และ $\overline{w_t}$ ตามลำดับ ในการหาระยะทางที่ใกล้ที่สุด ส่วนใหญ่ก็จะใช้การวัดระยะทางแบบยูคลิด
3. ปรับค่าความผิดพลาดสะสมของโหนด s ดังสมการ

$$error_s \leftarrow error_s + |\overline{w_s} - \bar{x}|^2 \tag{4.23}$$

4. หลังจากนั้นจะปรับค่าน้ำหนักของโหนด s และโหนดที่เชื่อมต่อกับโหนด s เพื่อให้เหมาะสมกับอินพุต $\bar{x}$ ที่เข้ามา

$$\overline{w_s} \leftarrow \overline{w_s} + e_w(\bar{x} - \overline{w_s}) \tag{4.24}$$

$$\overline{w_n} \leftarrow \overline{w_n} + e_n(\bar{x} - \overline{w_n}), \forall n \in Neighbour(s) \tag{4.25}$$

5. เพิ่มค่าอายุแขนทุกแขนของโหนด s
6. ถ้าโหนด s และ t เชื่อมต่อกันอยู่ให้กำหนดค่าแขนที่เชื่อมระหว่างทั้งสองโหนดเป็นศูนย์ ถ้ายังไม่เชื่อมต่อกันให้ทำการสร้างแขนเชื่อม
7. ตรวจสอบแขนทั้งหมดถ้ามีอายุมากกว่าค่าที่กำหนด a_{max} ให้ทำการลบแขนนั้นทิ้ง และตรวจสอบโหนดใด ๆ ที่ไม่มีการเชื่อมต่อให้ลบโหนดนั้นด้วย
8. ตรวจช่วงรอบที่ผ่านมาถ้ามีจำนวนเกินกว่าค่าที่กำหนด λ และจำนวนโหนดที่มีในปัจจุบันยังมีค่าไม่ถึงจำนวนโหนดสูงสุดที่กำหนด ให้ทำการแทรกโหนดใหม่ r ลงไปในกราฟ โดยมีวิธีการแทรกโหนดดังนี้
   - หาโหนด u ที่มีค่าความผิดพลาดสูงที่สุด เพื่อแทรกโหนดลงบริเวณโหนดที่ u รับผิดชอบ เนื่องจากโหนด u มีพื้นที่ครอบคลุมอินพุตมากจึงทำให้มีค่าความผิดพลาดสะสมสูง สมควรที่จะแทรกโหนดลงบริเวณนั้น
   - หาโหนด v ซึ่งเป็นโหนดใกล้เคียงโหนด u ที่มีความผิดพลาดสูงสุด เพื่อใช้เป็นจุดอ้างอิงอีกในการแทรกโหนดใหม่
   - แทรกโหนด r ระหว่างโหนด u และโหนด v และกำหนดค่าน้ำหนักของโหนด r เป็น

$$\overline{w_r} \leftarrow \frac{(\overline{w_u} + \overline{w_v})}{2} \tag{4.26}$$

- สร้างแขนเชื่อมระหว่างโหนด u กับโหนด r และ โหนด r กับโหนด v
- ปรับลดค่าความผิดพลาดของโหนด u กับโหนด v และกำหนดค่าความผิดพลาดเริ่มต้นของโหนด r

$$error_u \leftarrow \alpha \times error_u, error_v \leftarrow \alpha \times error_v \tag{4.27}$$

$$error_r \leftarrow error_u \tag{4.28}$$

9. ปรับลดค่าความผิดพลาดของโหนดทุกโหนด

$$error_j \leftarrow error_j - \beta \times error_j \tag{4.29}$$

กระบวนการเหล่านี้จะถูกทำซ้ำไปเรื่อย ๆ จนกว่าจะสอดคล้องตามเงื่อนไขหรือจนกว่าจะครบจำนวนรอบของการเรียนรู้

ในบทนี้ได้กล่าวถึงความรู้เบื้องต้นเกี่ยวกับโครงสร้างและการเรียนรู้ของระบบจำแนกประเภทแบบเรียนรู้ได้ XCS และ UCS โดยเป็นพื้นฐานในการวิเคราะห์ปัญหาที่เกิดขึ้นเกี่ยบกับระบบจำแนกประเภทเรียนรู้ได้แบบ UCS ซึ่งเป็นระบบจำแนกประเภทหลักของระบบที่ใช้ในวิทยานิพนธ์ฉบับนี้ ตัวจำแนกประเภทอีกตัวทดสอบกับระบบที่นำเสนอคือนิวรอนเน็ตเวิร์กซึ่งได้กล่าวถึงในบทนี้ด้วย นอกจากนี้เรายังกล่าวถึงการจัดกลุ่มข้อมูลสองแบบคือ เซลฟ์ออร์แกนไนซ์ซิ่งแม็พ - SOM และ โกลว์อิ่งนิวรอนแก๊ส - GNG โดยในระบบที่นำเสนอจะใช้การจัดกลุ่มทั้งสองในการจัดกลุ่มข้อมูลก่อนถูกส่งไปยังระบบจำแนกประเภทต่าง ๆ เป็นผลให้ระบบจำแนกประเภทรับผิดชอบขอบเขตของปัญหาแตกต่างกัน ซึ่งเป็นการแตกปัญหาใหญ่ให้เป็นปัญหาย่อยโดยปัญหาย่อยแต่ละตัวจะถูกจำแนกด้วยระบบจำแนกประเภทที่มีขนาดเล็กกว่าและซับซ้อนน้อยกว่าเมื่อใช้ระบบจำแนกประเภทตัวเดียวแก้ปัญหาใหญ่ทั้งหมด

ในบทถัดไปเราจะนำเสนอโครงข่ายงานในการแตกปัญหาที่รวมการจัดกลุ่มข้อมูลและการจำแนกประเภทข้อมูลเข้าด้วยกัน เพื่อลดเวลาในการเรียนรู้ของระบบจำแนกประเภทแบบเรียนรู้ได้ UCS อีกทั้งยังเพิ่มประสิทธิภาพในการจำแนกประเภทด้วย

บทที่ 5

ระบบสร้างกฎแบบปรับตัวได้

5.1 บทนำ

วิทยานิพนธ์ฉบับนี้นำเสนอการเชื่อมต่อระหว่างกระบวนการแตกปัญหาและกระบวนการเรียนรู้ของตัวจำแนกประเภท กระบวนการแตกปัญหาจะเปรียบเสมือนกับตัวพรีเกท ซึ่งข้อมูลจะผ่านกระบวนการแตกปัญหาก่อนเข้าไปสู่กระบวนการจำแนกประเภท ในที่นี้เราจะอาศัยกระบวนการจัดกลุ่มข้อมูล ดังนั้นข้อมูลที่เข้ามาจะถูกจัดกลุ่มก่อนส่งไปให้ตัวจำแนกประเภทเรียนรู้

ในส่วนของกระบวนการจำแนกประเภทเราจะใช้การจำแนกประเภทแบบออนไลน์ที่สามารถปรับตัวและเรียนรู้ได้เชื่อมต่อกับกลุ่มข้อมูลแต่ละกลุ่ม การเชื่อมต่อของทั้งสองกระบวนการสามารถมองได้ในรูปแบบของการแข่งกันเรียนรู้ของตัวจำแนกประเภท การที่ตัวจำแนกประเภทสามารถแข่งกันเรียนรู้ได้นั้นเนื่องจากอาศัยการผูกติดตัวเองไว้กับแต่ละโหนดของอัลกอริธึมที่ใช้ในการจัดกลุ่มข้อมูล

การจัดกลุ่มข้อมูลสามารถมองได้ว่าแต่ละกลุ่มพยายามดึงข้อมูลที่เข้ามาแต่ละครั้งไปเป็นสมาชิกของกลุ่มตัวเองโดยอาศัยการเปรียบเทียบคุณลักษณะของข้อมูลเป็นหลัก ซึ่งโดยธรรมชาติข้อมูลก็จะถูกกระจายไปยังกลุ่มต่าง ๆ เช่น เซลฟ์ออร์แกนไนซ์ซิ่งแม็พพยายามกระจายข้อมูลไปยังโหนดต่าง ๆ ดังนั้นจึงทำให้ตัวจำแนกประเภทแต่ละตัวเรียนรู้ข้อมูลในกลุ่มที่แตกต่างกัน โดยที่สมาชิกในกลุ่มเดียวกันก็จะมีคุณลักษณะคล้ายกัน

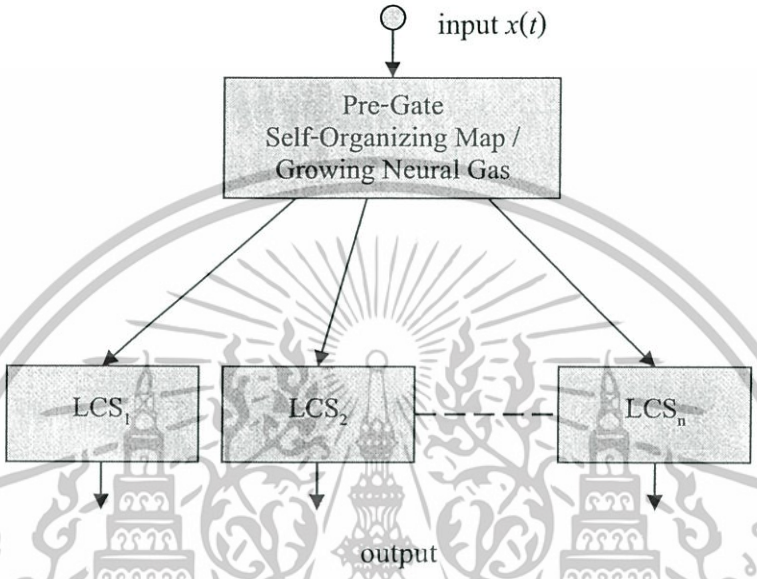
เมื่อสามารถแตกปัญหาให้เป็นปัญหาย่อยหลายปัญหาได้จักรกลเรียนรู้ที่ใช้ในการเรียนรู้ก็จะสามารถลดขนาดหรือใช้จำนวนกฎในการเรียนรู้น้อยลง ทำให้เพิ่มความเร็วในการเรียนรู้ ในการแตกปัญหานี้เราสันนิษฐานไว้ว่าเราอาจจะสามารถเพิ่มประสิทธิภาพในการจำแนกประเภทได้ด้วยในบางกรณี ตัวจำแนกประเภทหลักที่ใช้ทดสอบหลักคือ ตัวจำแนกประเภทแบบเรียนรู้ได้ UCS ตัวพรีเกทหลักที่ใช้ในการทดสอบคือ เซลฟ์ออร์แกนไนซ์ซิ่งแม็พและโกลว์อิ่งนิวรอนแก๊ส

ในบทนี้จะกล่าวถึงระบบของระบบสร้างกฎแบบปรับตัวได้รวมถึงผลการทดลองระบบที่นำเสนอกับข้อมูลสังเคราะห์ (Artificial datasets) และข้อมูลจริงจาก UCI KDD

5.2 ระบบสร้างกฎแบบปรับตัวได้

ระบบสร้างกฎแบบปรับตัวได้ที่นำเสนอจะอยู่ในรูปแบบของระบบดังแสดงในรูป 5.1 ในระบบแบบแรกตัวพรีเกทที่ใช้ในการแตกปัญหาจะเป็นเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ (Self-

Organizing Map – SOM) ในขณะที่ทำการแตกปัญหาไปนั้นตัวจำแนกแบบเรียนรู้ได้ UCS ก็จะทำการเรียนรู้ปัญหาย่อยแต่ละปัญหาไปด้วยพร้อม ๆ กัน เอาท์พุตสุดท้ายที่ได้ของระบบเกิดจากเอาท์พุตของ UCS ระบบใดระบบหนึ่งที่เหมาะสมกับอินพุตที่เข้ามาในขณะนั้น เราตั้งชื่อการเชื่อมต่อกันของทั้งสองระบบว่า "ระบบสร้างกฎแบบปรับตัวได้ – Self-Organizing UCS – SOUCS"



รูปที่ 5.1 ระบบของระบบสร้างกฎแบบปรับตัวได้

เซลฟ์ออร์แกนไนซ์ซิ่งแม็พในระบบสามารถทำงานได้สองลักษณะคือ ประการแรกคือทำหน้าที่ในการแตกปัญหาโดยการจัดกลุ่มข้อมูลจากคุณลักษณะของข้อมูล ประการที่สองเซลฟ์ออร์แกนไนซ์ซิ่งแม็พทำหน้าที่คล้ายกับเป็นตัวกำหนดเส้นทาง (Router) ให้กับข้อมูลที่เข้ามาเพื่อทำการกระจายไปยังตัวจำแนกประเภท UCS ที่อยู่ในระบบ ดังนั้นจะเห็นได้ว่าข้อมูลที่เข้ามาหนึ่งตัวนั้นจะมีตัวจำแนกประเภท UCS เพียงแค่หนึ่งตัวเท่านั้นที่รับผิดชอบในการเรียนรู้หรือการจำแนกประเภท

อย่างไรก็ตามเซลฟ์ออร์แกนไนซ์ซิ่งแม็พมีข้อจำกัดในการใช้งานอยู่บ้างคือ ต้องมีการกำหนดลักษณะการเชื่อมโยงของโครงข่ายนิวรอนไว้ก่อนซึ่งการกำหนดลักษณะการเชื่อมโยงก็จะส่งผลต่อการเรียนรู้แตกต่างกันไป และยังอาจจะทำให้เกิดโหนดตาย (Dead node) ขึ้นในแผนภาพได้ โหนดตายคือโหนดที่ไม่มีข้อมูลตกในโหนดนั้น ทำให้การเรียนรู้ของตัวจำแนกประเภทไม่เกิดขึ้น ดังนั้นเพื่อป้องกันไม่ให้เกิดปัญหาต่าง ๆ เหล่านี้ในงานวิจัยต่อมาเราจึงเปลี่ยนตัวพรีเกทจากเซลฟ์ออร์แกนไนซ์ซิ่งแม็พมาเป็นโกลว์อิ่งนิวรอนแก๊ส (Growing Neuro Gas – GNG) โดยที่โกลว์อิ่งนิวรอนแก๊สยังคงทำหน้าที่เป็นตัวแตกปัญหาและตัวกำหนดเส้นทางเหมือนเดิม โหนดแต่ละโหนดของโกลว์อิ่งนิวรอนแก๊สยังคงเชื่อมโยงกับตัวจำแนกประเภท UCS ที่แยกจากกัน เราเรียกการเชื่อมต่อกันของโกลว์อิ่งนิวรอนแก๊สและตัวจำแนกประเภท UCS ว่า "ระบบสร้างกฎแบบปรับตัวได้ – Growing UCS – GUCS"

5.2.1 กระบวนการเรียนรู้และทำงาน

SOUCS

ก่อนเริ่มกระบวนการเรียนรู้เราจำเป็นที่จะต้องกำหนดขนาดของแผนภาพ (จำนวนโหนด) เริ่มต้นของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พและลักษณะการเชื่อมโยงระหว่างโหนดในเน็ตเวิร์ก ในการทดลองเราจะทำการกำหนดขนาดของแผนภาพเป็น 2×2 และ 3×3 ซึ่งจะมีจำนวนโหนดเท่ากับ 4 และ 9 โหนด ตามลำดับ แต่ละโหนดของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พก็จะเชื่อมโยงกับตัวจำแนกประเภท UCS หนึ่งตัวหมายความว่าในแผนภาพ 2×2 จะมีจำนวนตัวจำแนกประเภท UCS 4 ตัว ในแผนภาพ 3×3 จะมีตัวจำแนกประเภท UCS 9 ตัวอิสระแยกจากกัน

ในขั้นตอนการเรียนรู้ระบบจะค่อย ๆ ทำการเรียนรู้ข้อมูลที่ได้รับเข้ามา สำหรับข้อมูลแต่ละชุดที่เข้ามา ระบบจะเริ่มจากหาโหนดที่มีระยะห่างจากอินพุตน้อยที่สุด (โหนดชนะ) โดยอาศัยฟังก์ชั่นวัดระยะห่าง จากนั้นระบบจะทำการปรับค่าเวกเตอร์น้ำหนักของโหนดชนะและโหนดใกล้เคียง (Neighborhood nodes) โดยพิจารณาจากระยะห่างของอินพุตกับเวกเตอร์น้ำหนักและพิจารณาระยะห่างระหว่างโหนดชนะกับโหนดใกล้เคียงในกรณีที่ปรับค่าเวกเตอร์น้ำหนักของโหนดใกล้เคียง หลังจากนั้นจะส่งข้อมูลต่อไปยังตัวจำแนกประเภท UCS ที่เชื่อมโยงกับโหนดชนะเพื่อให้ตัวจำแนกประเภท UCS นั้นได้ทำการเรียนรู้ ในกรณีที่หากฎที่แมตช์กับอินพุตไม่ได้ระบบจะทำการสร้างกฎ (Covering) ที่แมตช์กับอินพุตขึ้นมาใหม่ แต่หากมีกฎที่แมตช์กับอินพุตแล้วระบบก็จะทำการปรับปรุงค่าพารามิเตอร์ต่าง ๆ และเมื่อถึงเวลาที่เหมาะสมระบบจะทำการเรียกเจเนติกอัลกอริธึมขึ้นมาเพื่อสร้างกฎใหม่เพิ่มเข้าไปยังระบบ ดังนั้นสำหรับอินพุตแต่ละตัวที่เข้ามาระบบจะทำการปรับปรุงค่าต่างๆ ทั้งเซลฟ์ออร์แกนไนซ์ซิ่งแม็พและตัวจำแนกประเภท UCS

ในขั้นตอนการใช้งานหรือทดสอบระบบ เราจะทำการป้อนข้อมูลที่ใช้ในการทดสอบซึ่งจะไม่ถูกใช้ในขั้นตอนของการเรียนรู้ ระบบจะเริ่มจากหาโหนดชนะในเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ การหาโหนดชนะมีนัยยะสำคัญอีกอย่างคือการกำหนดเส้นทางให้กับอินพุตที่เข้ามา จากนั้นอินพุตที่เข้าจะถูกกำหนดไปยังตัวจำแนกประเภท UCS ที่เชื่อมโยงกับโหนดชนะนั้นเพื่อที่จะทำนายชนิดหรือประเภทของข้อมูลที่เข้ามา ในขั้นตอนนี้จะไม่มีการปรับปรุงพารามิเตอร์ใด ๆ ทั้งเซลฟ์ออร์แกนไนซ์ซิ่งแม็พและตัวจำแนกประเภท UCS

GUCS

เริ่มต้นกระบวนการเพียงแค่กำหนดจำนวนโหนดมากที่สุดเพียงเท่านั้นไม่จำเป็นต้องกำหนดการเชื่อมโยงระหว่างโหนดในเน็ตเวิร์ก โกลว์อิ่งนิวรอนแก๊สจะทำหน้าที่กำหนดการเชื่อมโยงเองอัตโนมัติรวมถึงการเพิ่มและการลบโหนดในแผนภาพด้วย แผนภาพเริ่มต้นของโกลว์อิ่งนิวรอนแก๊สจะเริ่มจากโหนดสองโหนดที่เชื่อมต่อกัน ดังนั้นระบบ GUCS จะเริ่มจากตัวจำแนกประเภท UCS สองตัวแยกอิสระจากกัน

ในขั้นตอนการเรียนรู้เช่นเดียวกับระบบ SOUCS ระบบ GUCS จะค่อย ๆ ทำการเรียนรู้ข้อมูลที่ได้รับเข้ามา สำหรับข้อมูลแต่ละชุดที่เข้ามา ระบบจะเริ่มจากหาโหนดที่มีระยะห่างจากอินพุตน้อยที่สุดเป็นลำดับที่หนึ่งและลำดับที่สอง และทำการปรับค่าเวกเตอร์น้ำหนัก และค่าพารามิเตอร์ต่าง ๆ ประจำโหนดทั้งสอง จากนั้นอินพุตจะถูกส่งต่อไปยังตัวจำแนกประเภท UCS ที่เชื่อมโยงกับโหนดที่มีระยะห่างน้อยที่สุดเป็นลำดับที่หนึ่ง เพื่อให้ตัวจำแนกประเภท UCS นั้นได้ทำการเรียนรู้เหมือนระบบ SOUCS

เมื่อโกลว์อิ่งนิวรอนแก๊สมีการเพิ่มโหนดให้กับเน็ตเวิร์ก แน่นอนว่าเราจำเป็นที่จะต้องเพิ่มตัวจำแนกประเภท UCS เข้าไปยังระบบด้วย ตัวจำแนกประเภท UCS ตัวใหม่นั้นสามารถที่จะสร้างจากฐานความรู้หรือกฎที่เรามีก่อนหน้าได้เพื่อเป็นการประหยัดเวลาในการเรียนรู้ โดยการคัดลอกกฎของตัวจำแนกประเภทที่เชื่อมโยงกับโหนดที่ใกล้ที่สุด กฎที่เราทำการคัดลอกไปนั้นมีทั้งกฎที่ดีและกฎที่ไม่ดี แต่กฎที่ไม่ดีนั้นจะถูกกระบวนการเจเนติกอัลกอริธึมในกระบวนการเรียนรู้ค่อย ๆ กำจัดออกไปเอง ส่วนขั้นตอนการลบโหนดในโกลว์อิ่งนิวรอนแก๊สเราสามารถที่จะทำการลบทั้งโหนดและตัวจำแนกประเภท UCS ที่เชื่อมโยงกับโหนดนั้นออกไปจากระบบได้

ในขั้นตอนการใช้งานหรือทดสอบระบบจะมีลักษณะเหมือนกับขั้นตอนของระบบ SOUCS โดยอินพุตที่เข้ามาจะถูกนำมาหาโหนดที่ใกล้ที่สุดจากนั้นจะส่งต่ออินพุตไปยังตัวจำแนกประเภท UCS ที่เชื่อมโยงกับโหนดนั้นเพื่อทำการจำแนกประเภทข้อมูลนั้น เช่นเดียวกันในขั้นตอนนี้จะไม่มีการปรับค่าพารามิเตอร์หรือการเพิ่มและลดโหนดใด ๆ ในโกลว์อิ่งนิวรอนแก๊ส

ข้อดีของระบบ SOUCS/GUCS นอกจากการแตกปัญหาใหญ่ให้เป็นปัญหาย่อยหลาย ๆ ปัญหาเพื่อทำให้ค่าความถูกต้องในการจำแนกประเภทเพิ่มขึ้นแล้ว หากพิจารณาแง่ของการทำงานในสภาพแวดล้อมแบบกระจาย [59] (Distributed environment) สมมุติให้ตัวจำแนกประเภท UCS แบบปกติสามารถประมวลผลข้อมูลได้จำนวน T ข้อมูลต่อวินาที ให้ R เป็นอัตราของจำนวนข้อมูลที่เข้ามาต่อวินาที ในกรณีใช้งานตัวจำแนกประเภท UCS เพียงตัวเดียว ถ้า R > T ทฤษฎีคิวรีบอกเราว่าข้อมูลที่เข้ามาใหม่จะถูกเข้าคิวรอในการประมวลผลยิ่ง R >> T มากแล้วยิ่งทำให้เวลาที่รออยู่ในคิวยิ่งนานมากขึ้น

ในกรณีที่เป็นระบบแบบกระจาย การคำนวณเวลาในการประมวลผลของสามารถคำนวณได้จากค่า min(U,M×T) คือค่าน้อยที่สุดระหว่าง ค่า U - จำนวนข้อมูลต่อวินาทีที่ตัวกำหนดเส้นทางสามารถกำหนดเส้นทางให้กับข้อมูลได้ (ในที่นี้ตัวกำหนดเส้นทางคือเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ) และ M×T โดยที่ M คือจำนวนตัวจำแนกประเภท UCS ที่มีในระบบ SOUCS/GUCS โดยปกติแล้วค่า U จะมีค่าน้อยกว่า T มาก กล่าวคือเซลฟ์ออร์แกนไนซ์ซิ่งแม็พใช้เวลาในการคำนวณน้อยกว่าตัวจำแนกประเภท UCS มาก ๆ ดังนั้นส่วนที่เป็นคอขวดคือตัวจำแนกประเภท UCS ดังนั้นถ้าเรามีตัวจำแนกประเภท UCS M ตัวก็สามารถกระจายข้อมูลไปพร้อม ๆ กันได้ M ตัว

จากเวลาในการประมวลผลที่กล่าวข้างต้นเราสมมติว่าตัวจำแนกประเภท UCS ทุกตัวไม่ว่าจะอยู่ในระบบที่ใช้งานเพียงตัวเดียวหรืออยู่ในระบบ SOUCS/GUCS ที่ใช้งานหลายตัวนั้น ทุกตัวมีจำนวนประชากร (กฎ) เท่ากันหมด ดังนั้นเวลาในการประมวลผลข้อมูลจะเท่ากันหมดด้วย และเรายังสมมติให้การกำหนดเส้นของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พนั้นกำหนดเส้นทางไปยังโหนดที่ต่างกันด้วย หากเป็นเช่นนี้ความเร็วที่เพิ่มขึ้นของระบบ SOUCS จะเพิ่มขึ้นถึง M เท่า หากสมมติฐานไม่เป็นจริงการคำนวณหาความเร็วที่เพิ่มขึ้นยากและซับซ้อน แต่อย่างไรก็ตามในการใช้งานจริงเรากำหนดให้จำนวนประชากรของตัวจำแนกประเภทแต่ละตัวในระบบ SOUCS นั้นมีจำนวนเท่ากับ 1/M ซึ่งจะทำให้เวลาที่ใช้ในการประมวลผลนั้นน้อยลงโดยเห็นได้ชัดจากการทดลองในลำดับต่อไป

5.3 การออกแบบการทดลองและผลการทดลอง

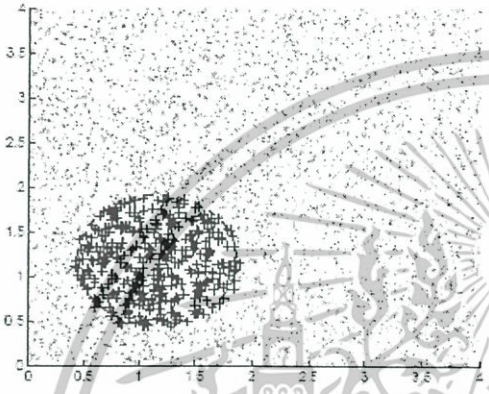
5.3.1 ข้อมูลที่ใช้ในการทดลอง

เพื่อให้ทดสอบระบบที่นำเสนอ เราจึงได้ออกแบบการทดลองออกเป็นสามการทดลอง โดยใช้ข้อมูลที่ทดสอบแตกต่างกันไปสามแบบ ชุดข้อมูลสังเคราะห์ ชุดข้อมูลขนาดเล็กและขนาดกลาง และชุดข้อมูลขนาดใหญ่ ซึ่งมีที่มาและลักษณะของข้อมูลดังนี้

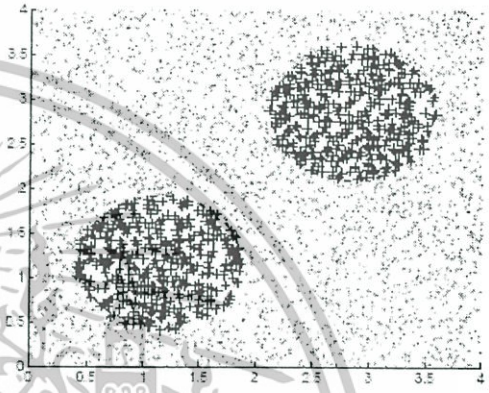
- ชุดข้อมูลสังเคราะห์ เราได้ทำสร้างและออกแบบข้อมูลสังเคราะห์ทั้งหมดหกชุด โดยที่แต่ละชุดจะมีความซับซ้อนของข้อมูลแตกต่างกันในเชิงของขอบเขตของแต่ละประเภทข้อมูล และการกระจายตัวของประเภทข้อมูล ข้อมูลแต่ละชุดจะมี 2 คุณลักษณะซึ่งทั้งสองจะมีค่าอยู่ในช่วง 0 – 4 ในแต่ละชุดข้อมูลจะทำการสุ่มสร้างโดยใช้การกระจายแบบยูนิฟอร์มขึ้นมาจำนวน 8000 ข้อมูล โดยจะมีการกำหนดประเภทของข้อมูลแตกต่างกันไป ข้อมูลแต่ละชุดมีชื่อว่า “circle 1”, “circle 2”, “circle 3”, “circle 4”, “cross circle” และ “mixed” ดังแสดงในรูป 6.2

  ชุดข้อมูล “circle” ทุกชุดจะมีข้อมูล 2 ประเภทคือ “+” และ “-” ข้อมูล “+” จะอยู่ในวงกลม ข้อมูล “-” จะอยู่ภายนอกวงกลม โดยที่ความซับซ้อนของปัญหาจะค่อย ๆ เพิ่มขึ้นตามการเพิ่มของจำนวนวงกลมตั้งแต่ปัญหา “circle 1” ถึง “circle 4” ชุดข้อมูล “cross circle” จะเป็นการซ้อนทับกันของวงกลมทั้งสี่วงซึ่งทำให้ขอบของวงกลมยิ่งซับซ้อนขึ้น ชุดข้อมูล “mixed” เป็นชุดที่มีความซับซ้อนของข้อมูลมากที่สุด โดยมีจำนวนประเภทข้อมูล 5 ประเภท ดังนี้ข้อมูลอยู่ภายในสามเหลี่ยมบนซ้าย ข้อมูลที่อยู่ภายในวงกลมบนขวา ข้อมูลที่อยู่ภายในวงกลมล่างซ้าย ข้อมูลที่อยู่ภายในสามเหลี่ยมล่างขวา และข้อมูลที่อยู่ภายนอกรูปทรงทั้งหมด

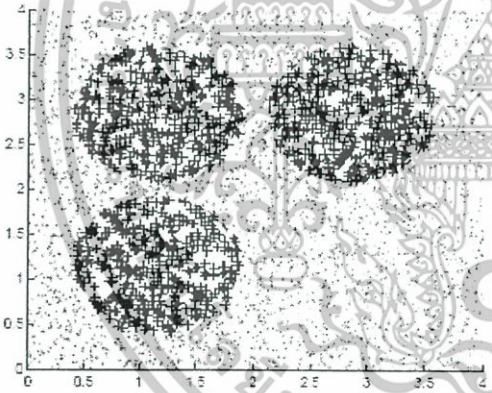
- ชุดข้อมูลขนาดเล็กและกลาง ชุดข้อมูลที่ใช้ทดสอบมีคุณลักษณะแตกต่างกันดังนี้ ประการแรกจำนวนคุณลักษณะ ในชุดข้อมูลขนาดเล็กจะมีจำนวนคุณลักษณะไม่เกิน 10 คุณลักษณะ ในชุดข้อมูลขนาดกลางจะมีจำนวนคุณลักษณะมากกว่า 10 คุณลักษณะ ประการที่สองจำนวนประเภทข้อมูลสำหรับชุดข้อมูลขนาดเล็กจะมีจำนวนประเภทข้อมูลไม่เกิน 4 ประเภท สำหรับชุดข้อมูลขนาดกลางจะมีจำนวนประเภทข้อมูลตั้งแต่ 5 ประเภทขึ้นไป คุณลักษณะอยู่ในรูปแบบของ จำนวนจริง หรือจำนวนเต็ม



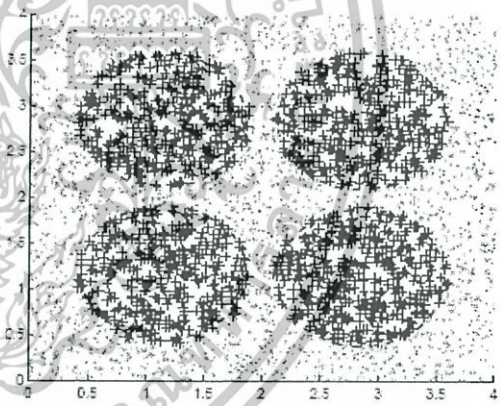
ก. ชุดข้อมูล “circle1”



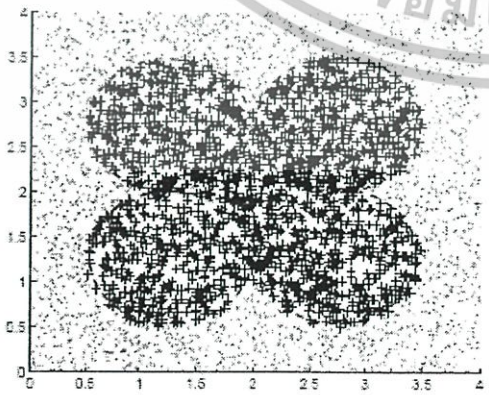
ข. ชุดข้อมูล “circle2”



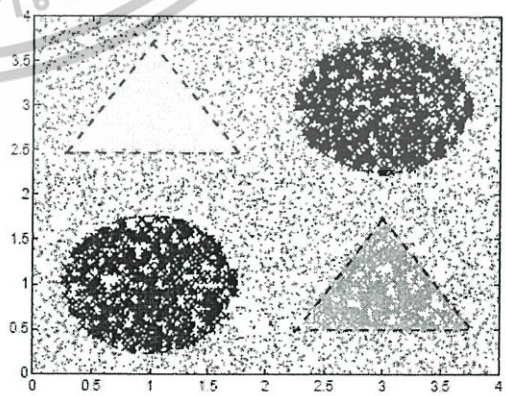
ค. ชุดข้อมูล “circle3”



ง. ชุดข้อมูล “circle4”



ค. ชุดข้อมูล “cross circle”



ง. ชุดข้อมูล “mixed”

รูปที่ 5.2 ชุดข้อมูลสังเคราะห์ทั้ง 6 ชุด

ตารางที่ 5.1 แสดงคุณสมบัติต่าง ๆ ของชุดข้อมูลที่ใช้ในงานวิจัยนี้ โดยที่ Inst แทนจำนวนข้อมูล Fs แทนจำนวนคุณลักษณะ R แทนคุณลักษณะที่เป็นจำนวนจริง I แทนคุณลักษณะที่เป็นจำนวนเต็ม C แทนจำนวนประเภทของข้อมูล

ข้อมูลชุดนี้เป็นบางส่วนของชุดข้อมูลมาตรฐานที่ใช้ในการทดสอบประสิทธิภาพในการจำแนกประเภทที่มีชื่อว่า UCI [20] โดยสามารถดาว์นโหลดได้จาก http://archive.ics.uci.edu/ml/ ยกเว้นข้อมูล Tao [48] ซึ่งนิยมใช้งานกันอย่างกว้างขวางในการทดสอบตัวจำแนกประเภทแบบเรียนรู้ได้

ตารางที่ 5.1 ชุดข้อมูลขนาดเล็กและกลางที่ใช้ทดสอบ

ข้อมูล	จำนวนข้อมูล	จำนวนคุณลักษณะทั้งหมด	คุณลักษณะจำนวนจริง	คุณลักษณะจำนวนเต็ม	ประเภทข้อมูล
balance-scale	625	4	4	0	3
breast-w	699	9	0	9	2
bupa	345	6	6	0	2
diabetes	768	8	8	0	2
glass	214	9	9	0	7
iris	150	4	4	0	3
segment	2310	19	19	0	7
sprial	195	2	2		2
tao	1888	2	2	0	2
vehicle	946	17	17	0	4
wine	178	13	13	0	3

- ชุดข้อมูลขนาดใหญ่ คือข้อมูล Forest-Covertype ซึ่งสามารถดาว์นโหลดได้ที่ UCI เช่นเดียวกัน เป็นข้อมูลที่ประเภทของป่าถูกรวมรวบในพื้นที่ประมาณ 70 ไมล์ในตอนตะวันตกเฉียงเหนือของ Denver, Colorado โดยมีประเภทของป่าทั้งหมด 7 ประเภท จำนวนข้อมูลทั้งหมด 581,012 ข้อมูล จำนวนคุณลักษณะทั้งหมด 54 คุณลักษณะ

5.3.2 การกำหนดลำดับการทดลองและค่าเริ่มต้นของระบบ ในการทดลอง

เนื่องจากมีการทดลองหลายการทดลอง สามารถแบ่งลำดับการทดลองดังนี้

- การทดลองที่หนึ่ง เริ่มจากทดสอบแนวคิดของการใช้ตัวจัดกลุ่มข้อมูลเซลฟ์ออร์แกนไนซ์ซิ่งแม็พเป็นตัวพรีเกทก่อนโดยจะทำการทดสอบตัวจำแนกประเภท UCS แบบตัวเดียวเปรียบเทียบกับระบบ SOUCS ในเชิงของประสิทธิภาพในการจำแนกประเภท

  หลังจากเสร็จสิ้นการทดลองที่หนึ่งแล้ว เราจะทำการวิเคราะห์ระบบ SOUCS แง่ของความเร็วในการเรียนรู้ที่เพิ่มขึ้น โดยจะพิจารณาจากค่าเฉลี่ยจำนวนกฎของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS หลังจากนั้นจะวิเคราะห์ในเชิงของการแตกปัญหาโดยอาศัยเครื่องมือที่ถูกพัฒนาขึ้นมาเพื่อวิเคราะห์การแตกปัญหาโดยเฉพาะ เพื่อให้เห็นว่าปัญหาที่มีความซับซ้อนขนาดใหญ่ถูกแตกเป็นปัญหาย่อยหลาย ๆ ปัญหาที่มีความซับซ้อนลดลง
- การทดลองที่สอง การทดลองเปลี่ยนตัวจำแนกประเภทของระบบ SOUCS จากตัวจำแนกประเภท UCS มาเป็นนิวรอนเน็ตเวิร์ก - ANN เพื่อทดสอบความสามารถของตัวพรีเกทในการแตกปัญหา
- การทดลองที่สาม การทดลองเปลี่ยนตัวพรีเกทจากเซลฟ์ออร์แกนไนซ์ซิ่งแม็พเป็นโกลว์อิ่งนิวรอนแก๊สเพื่อให้ระบบมีความยืดหยุ่นมากขึ้น โดยจะทำการเปรียบเทียบตัวจำแนกประเภท UCS แบบตัวเดียวเปรียบเทียบกับระบบ GUCS ในเชิงของประสิทธิภาพในการจำแนกประเภท

การกำหนดค่าพารามิเตอร์ต่าง ๆ ของตัวจำแนกประเภท UCS นั้นถูกกำหนดให้เหมือนกันในทุก ๆ ชุดข้อมูลและทุก ๆ สภาพแวดล้อม โดยมีการกำหนดค่าพารามิเตอร์ต่าง ๆ ที่สำคัญเหมือนกับในงานวิจัยของ Wilson [60] และ Bernado [23] ดังนี้คือ $\nu = 5$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 50$; $\theta_{GA} = 50$

ในการเปรียบเทียบระหว่างตัวจำแนกประเภท UCS กับระบบ SOUCS และระบบ GUCS เราจะกำหนดให้ผลรวมของจำนวนกฎในระบบ SOUCS และระบบ GUCS นั้นเท่ากับหรือใกล้เคียงกับตัวจำแนกประเภท UCS แบบตัวเดียว ตัวอย่างเช่น ถ้าเรากำหนดจำนวนกฎของตัวจำแนกประเภท UCS แบบตัวเดียว $N = n$ แล้วเราจะกำหนดจำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS หรือระบบ GUCS ให้เท่ากับ $N = n/M$ โดยที่ M คือจำนวน UCS ในแต่ละระบบ

เมื่อเปลี่ยนตัวจำแนกประเภทเป็น ANN ในทุกชุดข้อมูลจะมีจำนวนชั้นซ่อน (hidden layer) 1 ชั้น ในชุดข้อมูลขนาดเล็กและกลาง จำนวนโหนดในชั้นซ่อนจะถูกกำหนดเป็นเป็น 5 โหนด และชุดข้อมูล Forest Covertype จำนวนโหนดในชั้นซ่อนจะถูกกำหนดเป็นเป็น 20 โหนด

ในแต่ละการทดลองผลลัพธ์ของประสิทธิภาพการจำแนกประเภทได้จากการเฉลี่ยโดยใช้วิธี 10 – fold cross validation [61] ในแต่ละครั้งกำหนดค่าเริ่มต้นของการสุ่ม (Random seed) แตกต่างกันไปและทำการเรียนรู้ชุดข้อมูลเรียนรู้ทั้งหมด 500 รอบ ในแต่ละรอบหมายถึงการเรียนรู้ชุดข้อมูลเรียนรู้หมดทั้งชุด ยกเว้นชุดข้อมูล Forest covertype เนื่องจากมีขนาดใหญ่เราจะทำการเรียนรู้เพียงแค่ 10 รอบเท่านั้น ในการวิเคราะห์ประประสิทธิภาพเราใช้คณิตศาสตร์สถิติ t-test วัดความแตกต่างของประสิทธิภาพของการจำแนกประเภทซึ่งจะกล่าวถึงในหัวข้อถัดไป

5.3.3 การวิเคราะห์และเปรียบเทียบประสิทธิภาพ

ในการทดสอบความแตกต่างประสิทธิภาพระหว่างตัวจำแนกประเภท UCS แบบเดี่ยวและระบบ SOUCS เพื่อแสดงให้เห็นว่าระบบ SOUCS มีประสิทธิภาพในการจำแนกประเภทเพิ่มขึ้นหรือลดลงนั้น เราใช้ k-fold cross-validated paired t test [62] ข้อมูล S จะถูกแบ่งออกเป็นเซตที่ไม่ซ้อนทับกัน k เซต $T_1, T_2, \ldots, T_k$ โดยแต่ละเซตจะมีขนาดเท่ากันหรือใกล้เคียงกัน(ในที่นี้ k=10) ดังนั้นเราจะทำการทดลองทั้งหมด k รอบ โดยแต่ละรอบ i จะโดยใช้ข้อมูลเซตที่ T_i เป็นชุดข้อมูลทดสอบ ชุดข้อมูลส่วนที่เหลือ $T_j, j \neq i$

ตัวจำแนกประเภท UCS แบบเดี่ยวและระบบ SOUCS จะเรียนรู้ในชุดข้อมูลที่ใช้ในการเรียนรู้เดียวกันและทดสอบบนชุดข้อมูลที่ใช้ในการทดสอบเดียวกัน กำหนดให้ $p_A^{(i)}$ และ $p_B^{(i)}$ เป็นค่าประสิทธิภาพของตัวจำแนกประเภท UCS แบบเดี่ยวและระบบ SOUCSตามลำดับในรอบที่ i เริ่มจากการหาค่าความแตกต่างของประสิทธิภาพในแต่ละรอบ i : $p^{(i)} = p_A^{(i)} - p_B^{(i)}$ หลังจากนั้นสามารถหาค่า Student t test ได้ดังสมการที่ 5.1

$$t = \frac{\bar{p}.\sqrt{n}}{\sqrt{\frac{\sum_{i=1}^{n}(p^{(i)}-\bar{p})^2}{n-1}}} \qquad (5.1)$$

โดยให้ $\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p^{(i)}$ ภายใต้สมมุติฐานที่ศูนย์ (null hypothesis) คือค่าเฉลี่ยประสิทธิภาพของทั้งสองระบบไม่แตกต่างกัน และทำการทดสอบสมมุติฐานแบบข้างเดียว (one-sided) ทีละข้างที่ระดับนัยสำคัญที่ 0.05 โดยเริ่มจากด้านขวา ถ้าปฏิเสธสมมุติฐานที่ศูนย์แสดงว่าระบบ SOUCS มีค่าเฉลี่ยประสิทธิภาพในการจำแนกประเภทเพิ่มขึ้นอย่างมีนัยสำคัญ หลังจากนั้นจะทดสอบด้านซ้ายที่ระดับนัยสำคัญ 0.05 เช่นเดียวกัน ถ้าปฏิเสธสมมุติฐานที่ศูนย์แสดงว่าระบบ SOUCS มีค่าเฉลี่ยประสิทธิภาพในการจำแนกประเภทลดลงอย่างมีนัยสำคัญ ถ้ายอมรับทั้งสองสมมุติฐานแสดงว่าค่าเฉลี่ยประสิทธิภาพในการจำแนกประเภทไม่มีความแตกต่างกันอย่างมีนัยสำคัญ

5.4 การทดลองวัดประสิทธิภาพและในการจำแนกประเภทและความเร็วในการเรียนรู้

5.4.1 การทดลองที่ 1 เปรียบเทียบระหว่างตัวจำแนกประเภท UCS และระบบ SOUCS

ในการทดลองแรกเริ่มจากการเปรียบเทียบค่าความถูกต้องที่ได้ เมื่อทดสอบกับข้อมูลสังเคราะห์ทั้ง 6 ชุดข้อมูลระหว่างตัวจำแนกประเภท UCS และระบบ SOUCS โดยกำหนดแผนภาพของเซลฟ์ออร์แกนไนซ์ซึ่งแม็พเป็นขนาด 2×2 (แทนด้วย SOUCS-1) และ 3×3 (แทนด้วย SOUCS-2) การเชื่อมต่อระหว่างโหนดในแผนภาพเป็นการเชื่อมต่อแบบสี่เหลี่ยม (ดังรูป 5.8) ดังนั้นจะใช้ตัวจำแนกประเภท UCS ทั้งหมด 4 และ 9 ตัวตามลำดับ กำหนดจำนวนประชากรของตัวจำแนกประเภท UCS แบบตัวเดียว $N = 1000$ ระบบ SOUCS-1 จำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัว $N = 250$ ระบบ SOUCS-2 จำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัว $N = 111$

ตารางที่ 5.2 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลสังเคราะห์ (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=1000 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=250 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ N=111 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2)

ชุดข้อมูล	UCS(N=1000)	SOUCS-1(N=250)	SOUCS-2(N=111)
circle1	0. 981±0.006	0.981±0.004	0.982±0.004
circle2	0.965±0.023	0.966±0.009	0.962±0.023
circle3	0.942±0.008	0.956±0.012 ↑	0.946±0.015
circle4	0.940±0.010	0.951±0.007 ↑	0.931±0.007 ↓
cross circle	0.946±0.007	0.945±0.019	0.941±0.010
mixed	0.928±0.007	0.939±0.011↑	0.932±0.011

ตารางที่ 5.2 แสดงค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของค่าความถูกต้องในการจำแนกประเภทข้อมูลสังเคราะห์ที่กำหนดให้ จะเห็นได้ว่าระบบ SOUCS-1 มีประสิทธิภาพในการจำแนกประเภทดีกว่าการใช้ตัวจำแนกประเภท UCS แบบตัวเดียวสามชุดข้อมูลและมีประสิทธิภาพเทียบเท่ากันในอีกสามปัญหาที่เหลือ SOUCS-2 มีประสิทธิภาพในการจำแนกประเภทด้อยกว่าการใช้ตัวจำแนกประเภท UCS แบบตัวเดียวหนึ่งชุดข้อมูลและมีประสิทธิภาพเทียบเท่ากันในอีกห้าชุดข้อมูล

ระบบ SOUCS-1 มีประสิทธิภาพในการจำแนกดีกว่าตัวจำแนกประเภท UCS ในชุดข้อมูล circle3 circle4 และ mixed ซึ่งชุดข้อมูลเหล่านี้มีความซับซ้อนเพิ่มขึ้นตามลำดับ อย่างไรก็ตามในชุดข้อมูล circle 4 ระบบ SOUCS-2 มีประสิทธิภาพในการจำแนกด้อยกว่าตัวจำแนกประเภท UCS เราสันนิษฐานว่าอาจเกิดจากจำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2 มีจำนวนน้อยเกินกว่าที่จะจำแนกชุดข้อมูลได้

ดังนั้นเราจึงทำการทดลองเพิ่มโดยการเพิ่มจำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัวที่อยู่ในระบบ SOUCS-1 เป็น 500 และสำหรับ SOUCS-2 เพิ่มเป็น 222 ตารางที่ 5.3 แสดงการเปรียบเทียบค่าความถูกต้องของระบบ SOUCS กับตัวจำแนกประเภท UCS แบบตัวเดียวที่มีจำนวนประชากรเป็น 1000 (แทนเป็น UCS-1) ในตารางที่ 5.4 จะแสดงการเปรียบเทียบค่าความถูกต้องของระบบ SOUCS กับตัวจำแนกประเภท UCS แบบตัวเดียวที่มีจำนวนประชากรเป็น 2000 (แทนเป็น UCS-2)

ตารางที่ 5.3 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลสังเคราะห์ (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=1000 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=500 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ N=222 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2)

ชุดข้อมูล	UCS-1 (N=1000)	SOUCS-1 (N=500)	SOUCS-2 (N=222)
circle1	0. 981±0.006	0.982±0.012	0.985±0.004
circle2	0.965±0.023	0.968±0.011	0.965±0.010
circle3	0.942±0.008	0.957±0.005↑	0.961±0.012↑
circle4	0.940±0.010	0.955±0.007↑	0.953±0.007↑
cross circle	0.946±0.007	0.951±0.005↑	0.954±0.008↑
mixed	0.928±0.007	0.940±0.006↑	0.934±0.006↑

ตารางที่ 5.4 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลสังเคราะห์ (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=2000 สำหรับตัวจำแนกประเภท UCS แบบตัว

เดียว N=500 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ N=222 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2)

ชุดข้อมูล	UCS-2 (N=2000)	SOUCS-1 (N=500)	SOUCS-2 (N=222)
circle1	0. 984±0.006	0.982±0.012	0.985±0.004
circle2	0.964±0.008	0.968±0.011	0.965±0.010
circle3	0.948±0.009	0.957±0.005↑	0.961±0.012↑
circle4	0.947±0.006	0.955±0.007	0.953±0.007
cross circle	0.948±0.007	0.951±0.005	0.954±0.008↑
mixed	0.930±0.011	0.940±0.006↑	0.934±0.006

จากตารางที่ 5.3 และ 5.4 จะเห็นได้ว่าระบบ SOUCS-1 มีค่าความถูกต้องในการจำแนกประเภทดีกว่าตัวจำแนกประเภท UCS-1 ในสี่ชุดข้อมูลและดีกว่าตัวจำแนกประเภท UCS-2 หนึ่งชุดข้อมูล เช่นเดียวกันระบบ SOUCS-2 มีค่าความถูกต้องในการจำแนกประเภทดีกว่า UCS-1 ในสี่ชุดข้อมูล และดีกว่าตัวจำแนกประเภท UCS-2 สองชุดข้อมูล

สังเกตว่าเมื่อทำการเพิ่มจำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ SOUCS-2 จะทำให้ค่าความถูกต้องในการจำแนกประเภทข้อมูลเพิ่มขึ้น ข้อสันนิษฐานเรื่องของจำนวนประชากรข้างต้นจึงถูกต้อง ดังนั้นในการกำหนดจำนวนประชากรจึงต้องกำหนดให้เหมาะสมกับปัญหาด้วย การกำหนดจำนวนประชากรน้อยไปอาจจะทำให้ไม่สามารถจำแนกชุดข้อมูลได้ และการกำหนดจำนวนประชากรที่มากเกินไปจะทำให้การเรียนรู้ชุดข้อมูลใช้เวลาเพิ่มขึ้นโดยที่ค่าความถูกต้องในการจำแนกอาจจะคงที่หรือเพิ่มขึ้นเล็กน้อยอย่างไม่มีนัยยะสำคัญ

การทดลองต่อไปเราทำการทดลองกับข้อมูลขนาดเล็กและขนาดกลางที่ได้จากชุดข้อมูล UCI แผนภาพของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พจะกำหนดเช่นเดียวกับการทดลองกับข้อมูลสังเคราะห์ จำนวนประชากรของตัวจำแนกประเภท UCS แบบตัวเดียว N = 6400 ระบบ SOUCS-1 จำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัว N = 1600 ระบบ SOUCS-2 จำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัว N = 711

ตารางที่ 5.5 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดเล็กและกลาง (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=6400 สำหรับตัวจำแนกประเภท UCS แบบตัว

เดียว N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ N=711 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2)

ชุดข้อมูล	UCS(N=6400)	SOUCS-1 (N=1600)	SOUCS-2(N=711)
balance-scale	0.815±0.027	0.852±0.012↑	0.854±0.015↑
breast-w	0. 961±0.015	0.966±0.012	0.973±0.016↑
bupa	0.681±0.053	0.715±0.034	0.719±0.035↑
diabetes	0.766±0.056	0.762±0.035	0.761±0.047
glass	0.710±0.054	0.677±0.058	0.682±0.012
iris	0.953±0.045	0.960±0.034	0.947±0.042
segment	0.962±0.011	0.965±0.011	0.949±0.050↓
tao	0.880±0.012	0.940±0.021↑	0.956±0.016↑
vehicle	0.715±0.031	0.690±0.040↓	0.669±0.041↓
wine	0.966±0.047	0.967±0.060	0.949±0.050

จากตารางที่ 5.5 แสดงค่าความถูกต้องในการจำแนกประเภทระหว่างตัวจำแนกประเภท UCS และระบบ SOUCS จะเห็นว่าระบบ SOUCS นั้นให้ค่าความถูกต้องในการจำแนกประเภทดีกว่าการใช้ตัวจำแนกประเภท UCS แบบตัวเดียว ระบบ SOUCS-2 ให้ค่าความถูกต้องในการจำแนกประเภทดีกว่าสี่ชุดข้อมูลจากสิบชุดข้อมูล ให้ค่าเทียบเท่าในอีกห้าชุดข้อมูล และให้ค่าด้อยกว่าในอีกสองชุดข้อมูลที่เหลือ

ระบบ SOUCS ให้ค่าความถูกต้องด้อยกว่าในชุดข้อมูล segment และ vehicle ทั้งนี้เพราะทั้งสองชุดข้อมูลมีจำนวนประเภทข้อมูลมาก (ชุดข้อมูล segment มี 7 ประเภทและชุดข้อมูล vehicle มี 4 ประเภท) และมีจำนวนคุณลักษณะมาก (ชุดข้อมูล segment มี 19 คุณลักษณะและชุดข้อมูล vehicle มี 17 คุณลักษณะ) ซึ่งหลังจากใช้เซลฟ์ออร์แกนไนซ์ซิ่งแม็พทำการแตกปัญหาแล้ว ปัญหาย่อยบางปัญหาอาจจะยังคงมีความซับซ้อนของข้อมูลสูง จำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัวอาจจะมีกฎไม่เพียงพอในการเรียนรู้ชุดข้อมูลย่อย ๆ นั้น

ดังนั้นเราจึงทดลองเพิ่มจำนวนประชากรในระบบ SOUCS สำหรับสองชุดข้อมูลดังกล่าว ตารางที่ 5.6 แสดงการเปรียบเทียบค่าความถูกต้องระหว่างตัวจำแนกประเภท UCS แบบตัวเดียวกับระบบ SOUCS เมื่อทำการเพิ่มจำนวนประชากรของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 เป็น N=3200 และ ระบบ SOUCS-2 เป็น N= 1422 และตารางที่ 5.7 แสดงการเปรียบเทียบระบบ SOUCS ที่เพิ่มจำนวนประชากรแล้วกับตัวจำแนกประเภท UCS แบบตัวเดียวเมื่อเพิ่มประชากรเป็น N=12800 (UCS-2)

ตารางที่ 5.6 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดเล็กและกลาง (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=6400 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ N=1422 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2)

ชุดข้อมูล	UCS-1 (N=6400)	SOUCS-1 (N=3200)	SOUCS-2 (N=1422)
segment	0.962±0.011	0.971±0.009↑	0.970±0.011↑
vehicle	0.715±0.031	0.714±0.034	0.697±0.030

ตารางที่ 5.7 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดเล็กและกลาง (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=12800 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ N=1422 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2)

ชุดข้อมูล	UCS-2 (N=12800)	SOUCS-1 (N=3200)	SOUCS-2 (N=1422)
segment	0.968±0.014	0.971±0.009	0.970±0.011
vehicle	0.719±0.021	0.714±0.034	0.697±0.030

สำหรับชุดข้อมูล segment จะเห็นได้ว่าระบบ SOUCS-1 และ SOUCS-2 ให้ค่าความถูกต้องดีกว่าตัวจำแนกประเภท UCS-1 แสดงให้เห็นว่าการใช้เซลฟ์ออร์แกนไนซ์ซิ่งแม็พนั้นสามารถแตกปัญหาเพื่อลดความซับซ้อนของข้อมูลได้ และค่าความถูกต้องของระบบ SOUCS ทั้งสองสามารถเปรียบเทียบกันได้กับ UCS-2 แต่ระบบ SOUCS ทั้งสองใช้เวลาในการเรียนรู้น้อยกว่าเนื่องจากใช้จำนวนกฎในการเรียนรู้น้อยกว่า

การทดลองถัดไปเราจะทดลองระบบ SOUCS กับชุดข้อมูลขนาดใหญ่ Forest-Covertype เริ่มต้นจากกำหนดค่าต่าง ๆ เหมือนกับการทดลองกับชุดข้อมูล UCI ทั้งสิบชุดข้อมูล แต่เราทำการเพิ่มจำนวนโหนดของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พเป็น 4×4 (แทนด้วย SOUCS-3) ซึ่งทำให้มีตัวจำแนกประเภท UCS ย่อยเป็นทั้งหมด 16 ตัว แต่ละตัวมีจำนวนประชากร N=400 ตารางที่ 5.8 แสดงค่าความถูกต้องของระบบ SOUCS เปรียบเทียบกับตัวจำแนกประเภท UCS แบบตัวเดียว

ตารางที่ 5.8 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดใหญ่ (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=6400 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=1600 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 N=711 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2 และ N=400 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-3)

ชุดข้อมูล	UCS	SOUCS-1	SOUCS-2	SOUCS-3
Forest-Covertype	0.729±0.005	0.719±0.006↓	0.711±0.004↓	0.694±0.006↓

ผลการทดลองที่ได้จากตารางข้างต้นเป็นไปตามที่คาดการณ์ไว้ เนื่องจากจำนวนประชากรของตัวจำแนกประเภทแต่ละตัวในระบบ SOUCS นั้นค่อนข้างน้อยเกินไป ดังนั้นเราจึงทำการเพิ่มจำนวนประชากรของตัวจำแนกประเภท UCS ทุกตัวในระบบ SOUCS ทั้งสามระบบเป็น N=3200 หลังจากการเพิ่มจำนวนประชากรทำให้ค่าความถูกต้องของระบบ SOUCS เพิ่มขึ้นจนสามารถเทียบเท่าได้กับการใช้ตัวจำแนกประเภท UCS ตัวเดียวดังแสดงในตารางที่ 5.9 แต่ระบบ SOUCS ทั้งสามระบบใช้เวลาในการเรียนรู้และทำงานน้อยกว่าเนื่องจากจำนวนประชากรของตัวจำแนกประเภท UCS ย่อยในระบบ SOUCS นั้นใช้เพียงครึ่งหนึ่ง N=3200 เมื่อเปรียบเทียบกับจำนวนประชากรของตัวจำแนกประเภท UCS ตัวเดียว N=6400

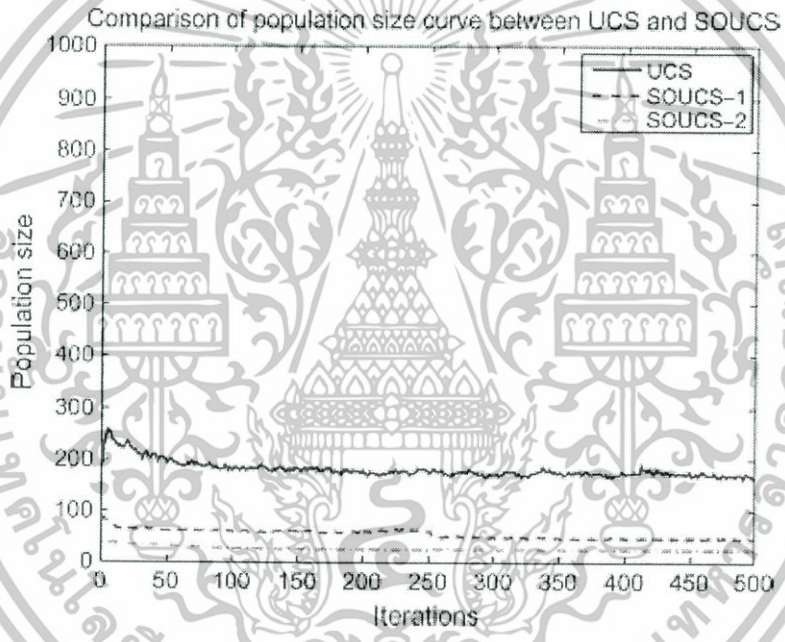
ตารางที่ 5.9 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดใหญ่ (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=6400 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2 และ N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-3)

ชุดข้อมูล	UCS	SOUCS-1	SOUCS-2	SOUCS-3
Forest-Covertype	0.729±0.005	0.728±0.003	0.725±0.003	0.724±0.005

เวลาในการประมวลผลของระบบ SOUCS เกิดจากสองส่วนคือ เวลาในการประมวลผลของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ และเวลาในการประมวลผลของตัวจำแนกประเภท UCS แต่ละตัวที่ผูกติดกับโหนดของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ ซึ่งเวลาที่เกิดจากการประมวลผลของเซลฟ์

ออร์แกนไนซ์ซิ่งแม็พนั้นจะใช้เวลาน้อยมากเมื่อเทียบกับเวลาในการประมวลผลของตัวจำแนกประเภท UCS แต่ละตัว ดังนั้นอาจกล่าวได้ว่าเวลาของระบบ SOUCS นั้นส่วนใหญ่เกิดจากการประมวลผลของตัวจำแนกประเภท UCS ที่ผูกติดกับโหนดของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ

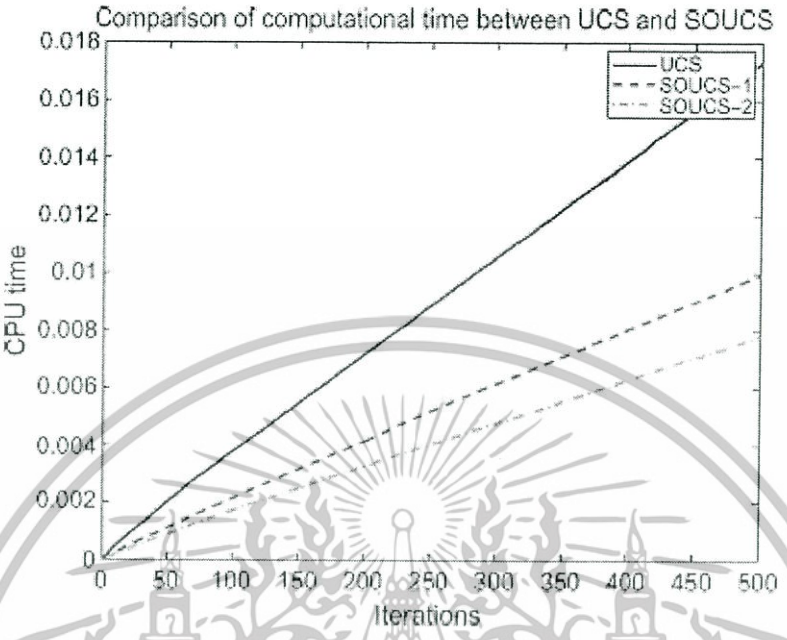
เวลาในการเรียนรู้ของตัวจำแนกประเภท UCS นั้นส่วนใหญ่จะใช้ไปในการค้นหากฎที่สอดคล้องกับข้อมูลที่เข้ามา ดังนั้นถ้าเรากำหนดประชากรกฎน้อยก็จะใช้เวลาในการค้นหาน้อยตามไปด้วย ในระบบ SOUCS-1 จะใช้เวลาในการค้นหากฎที่สอดคล้องประมาณหนึ่งในสี่ของตัวจำแนกประเภท UCS แบบตัวเดียว และระบบ SOUCS-2 จะใช้เวลาประมาณหนึ่งในเก้าของตัวจำแนกประเภท UCS แบบตัวเดียว เนื่องจากประชากรของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ SOUCS-2 ถูกกำหนดให้มีเพียงแค่หนึ่งในสี่และหนึ่งในเก้าของตัวจำแนกประเภท UCS แบบตัวเดียวตามลำดับ



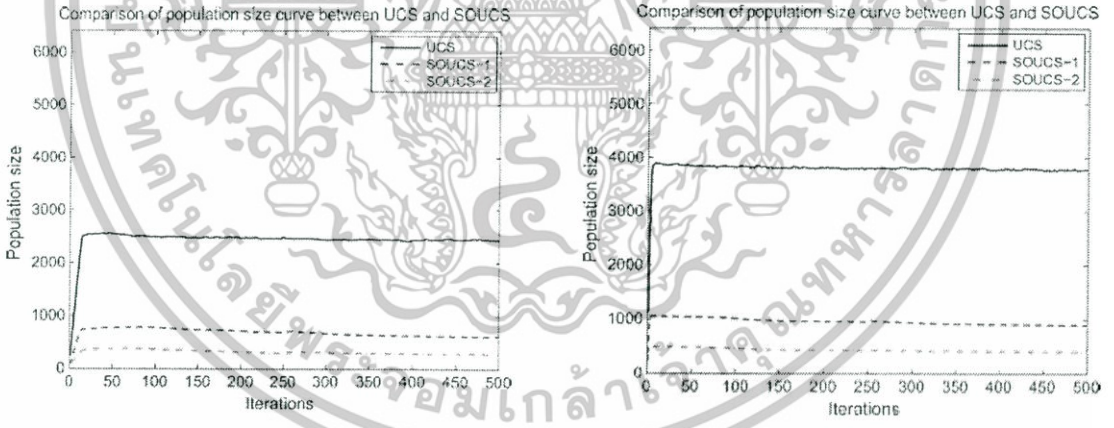
รูปที่ 5.3 กราฟขนาดจำนวนประชากรของตัวจำแนกประเภท UCS แบบตัวเดียวเปรียบเทียบกับระบบ SOUCS-1 และ SOUCS-2 ในข้อมูลสังเคราะห์ CIRCLE-4

จากรูปที่ 5.3 แสดงตัวอย่างจำนวนประชากรกฎที่เกิดขึ้นจริง (macro-classifiers) ในชุดข้อมูลสังเคราะห์ circle-4 เส้นสีน้ำเงิน เส้นประสีแดง และเส้นประสีเขียวแสดงจำนวนประชากรที่เกิดขึ้นจริงในตัวจำแนกประเภท UCS แบบตัวเดียว ระบบ SOUCS-1 และระบบ SOUCS-2 ตามลำดับ จะเห็นได้ว่าจำนวนกฎที่เกิดขึ้นจริงในระบบ SOUCS ทั้งสองจะมีจำนวนกฎน้อยกว่าตัวจำแนกประเภท UCS แบบตัวเดียว เพื่อแสดงให้เห็นถึงการประมวลผลที่เร็วขึ้นของระบบ SOUCS เราได้พล็อตกราฟของเวลาในการประมวลผลของแต่ละระบบโดยเปรียบเทียบจากเวลาของซีพียู (CPU clock) ดังรูปที่ 5.4 กราฟแสดงให้เห็นว่าเวลาที่ใช้ในการประมวลผลของระบบ

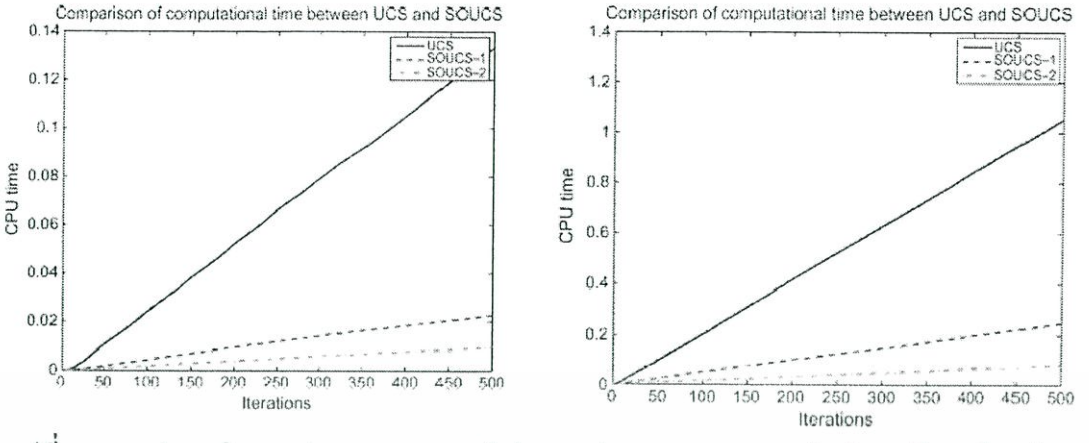
SOUCS ทั้งสองใช้น้อยกว่าตัวจำแนกประเภท UCS แบบตัวเดียว และ ระบบ SOUCS-2 ใช้เวลาในการประมวลผลน้อยกว่าระบบ SOUCS-1



รูปที่ 5.4 กราฟเวลาในการประมวลผลของตัวจำแนกประเภท UCS แบบตัวเดียวเปรียบเทียบกับระบบ SOUCS-1 และ SOUCS-2 ในข้อมูลสังเคราะห์ CIRCLE-4



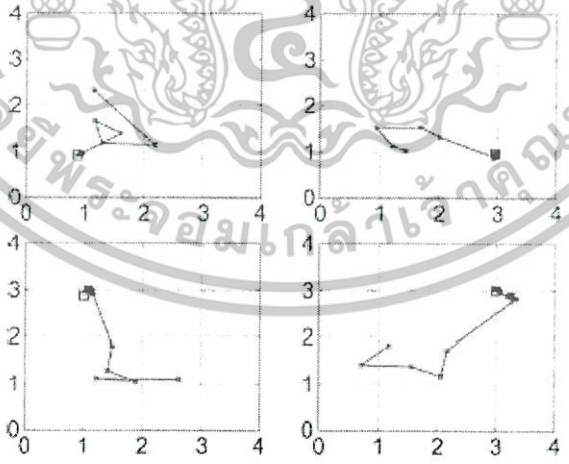
รูปที่ 5.5 กราฟขนาดจำนวนประชากรของตัวจำแนกประเภท UCS แบบตัวเดียวเปรียบเทียบกับระบบ SOUCS-1 และ SOUCS-2 ในข้อมูลขนาดเล็กและกลาง BREAST-W (ด้านซ้าย) และ SEGMENT (ด้านขวา)



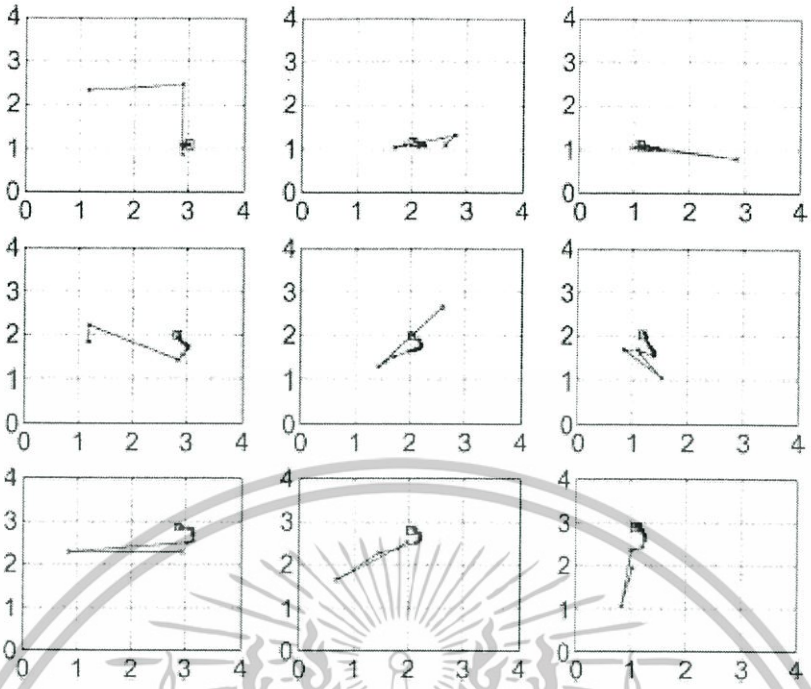
รูปที่ 5.6 กราฟเวลาในการประมวลผลของตัวจำแนกประเภท UCS แบบตัวเดียวเปรียบเทียบกับระบบ SOUCS-1 และ SOUCS-2 ในข้อมูลขนาดเล็กและกลาง BREAST-W (ด้านซ้าย) และ SEGMENT (ด้านขวา)

รูปที่ 5.5 และ 5.6 แสดงตัวอย่างจำนวนกฎที่เกิดขึ้นจริงและเวลาที่ใช้ในการประมวลผลของข้อมูลขนาดเล็กและขนาดกลาง breast-w และ segment ของระบบ SOUCS เปรียบเทียบกับตัวจำแนกประเภท UCS แบบตัวเดียว ระบบ SOUCS แสดงให้เห็นถึงการใช้เวลาในการประมวลผลที่น้อยกว่าการใช้ตัวจำแนกประเภท UCS แบบตัวเดียว

ในเชิงของการวิเคราะห์การแตกปัญหาโดยการใช้เซลฟ์ออร์แกนไนซ์ซิ่งแม็พเป็นตัวพรีเกทนั้น สามารถแสดงได้โดยสังเกตการเปลี่ยนแปลงของโหนดแต่ละโหนดในแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ

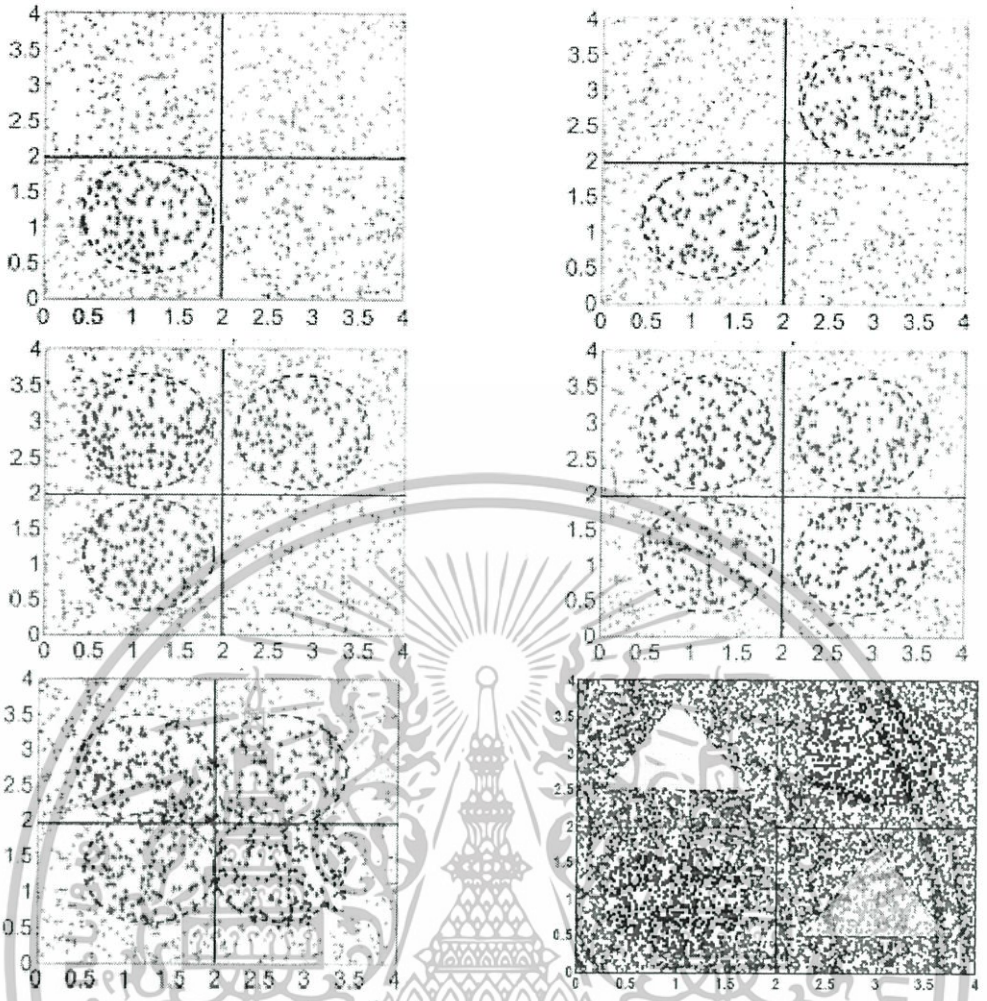


รูปที่ 5.7 กราฟเวคเตอร์น้ำหนักประจำโหนดของแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 2×2 ทดลองกับข้อมูลสังเคราะห์ CIRCLE4

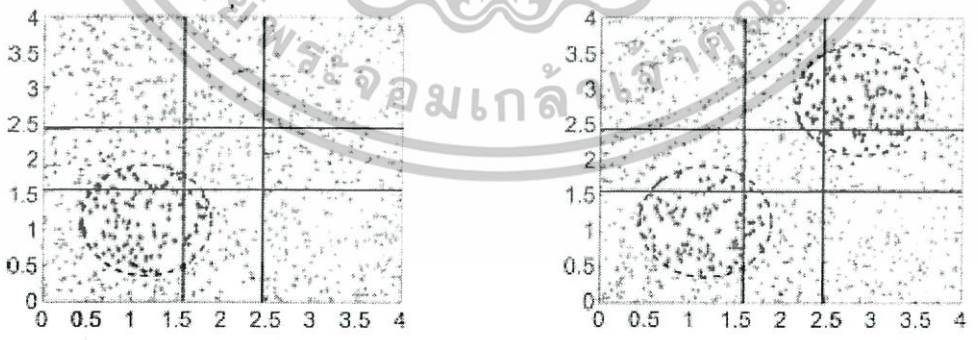


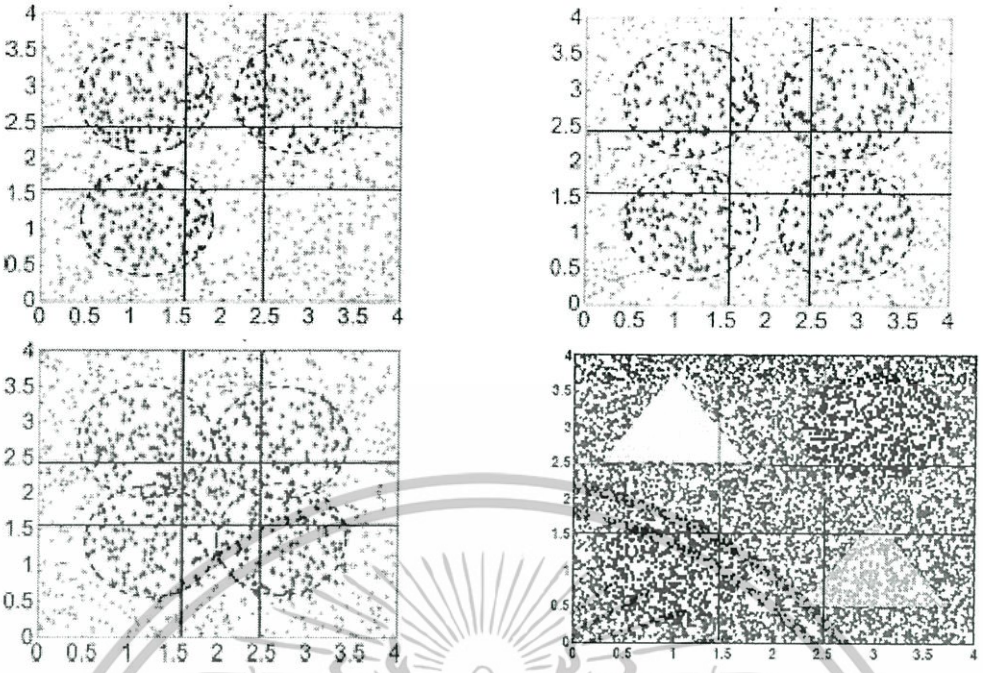
รูปที่ 5.8 กราฟเวคเตอร์น้ำหนักประจำโหนดของแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 3×3 ทดลองกับข้อมูลสังเคราะห์ CIRCLE4

จากรูปที่ 5.7 และ 5.8 แสดงแผนภาพการเปลี่ยนแปลงของโหนดแต่ละโหนดของแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 2×2 และ 3×3 ตามลำดับ เมื่อทดลองกับข้อมูลสังเคราะห์ circle4 จุดสีน้ำเงินแต่ละจุดในกราฟแสดงถึงตำแหน่งของเวคเตอร์น้ำหนักหลังจากผ่านการเรียนรู้ไปแล้ว 25 รอบ จุดสีน้ำเงินแต่ละจุดจะเชื่อมกันไปเรื่อย ๆ เป็นวิถีของการเรียนรู้ (Learning Trajectory) จนถึงจุดสี่เหลี่ยมสีดำซึ่งเป็นจุดสุดท้ายแสดงถึงตำแหน่งของเวคเตอร์น้ำหนักเมื่อการเรียนรู้เสร็จสิ้น ข้อสังเกตคือการเรียนรู้ของแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พนั้นจะเป็นการเรียนรู้ที่อยู่บนพื้นฐานของความหนาแน่นของคุณลักษณะข้อมูล ข้อมูลสังเคราะห์ที่เรานำมาทดสอบนั้นเป็นข้อมูล 2 มิติ ที่มีการกระจายแบบสม่ำเสมอ (Uniform distribution) ดังนั้นเมื่อแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พเรียนรู้เสร็จสิ้นจึงแบ่งข้อมูลออกเป็น 4 ส่วนและ 9 ส่วน ในกรณีเมื่อใช้แผนภาพขนาด 2×2 และ 3×3 ตามลำดับ



รูปที่ 5.9 ลักษณะการแตกปัญหาที่เกิดขึ้นเมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 2×2 กับข้อมูลสังเคราะห์ (CIRCLE1 CIRCLE2 CIRCLE3 CIRCLE4 CROSS CIRCLE และ MIXED เรียงจากซ้ายไปขวาและบนลงล่าง)





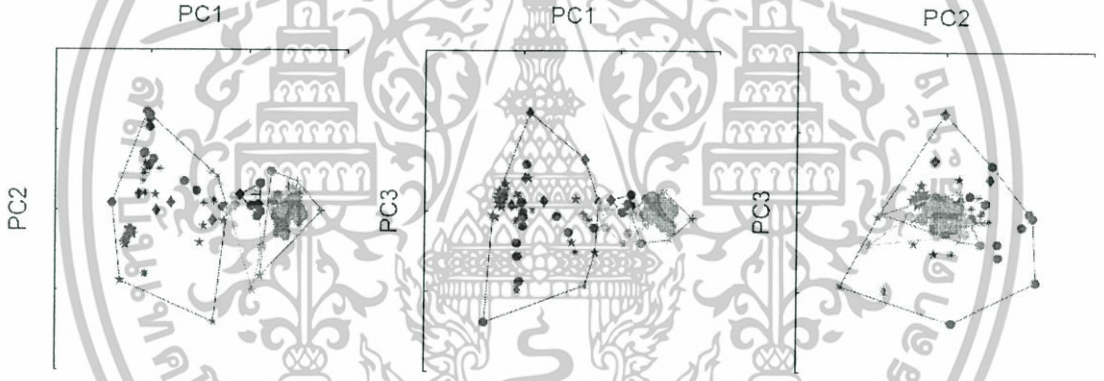
รูปที่ 5.10 ลักษณะการแตกปัญหาที่เกิดขึ้นเมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 3×3 กับข้อมูลสังเคราะห์ (CIRCLE1 CIRCLE2 CIRCLE3 CIRCLE4 CROSS CIRCLE และ MIXED เรียงจากซ้ายไปขวาและบนลงล่าง)

รูปที่ 5.9 และ 5.10 แสดงการแตกปัญหาที่เกิดขึ้นจากระบบ SOUCS ในรอบสุดท้ายของการเรียนรู้ เส้นสีดำแสดงการการแบ่งข้อมูลที่เกิดขึ้นโดยแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ เส้นประจะเป็นขอบเขตของชนิดข้อมูล สังเกตว่าในบางปัญหาเมื่อระบบ SOUCS แบ่งข้อมูลออกเป็นกลุ่ม ๆ ในบางกลุ่มนั้นจะมีเพียงชนิดข้อมูลเดียวเท่านั้นเช่นในรูปที่ 5.9 และ 5.10 ปัญหา circle1 และ circle2 ในชุดข้อมูล mixed ทั้งหมดจะมี 5 ชนิดข้อมูลเมื่อเรียนรู้โดยใช้ตัวจำแนกประเภท UCS แบบตัวเดียว UCS จะต้องเรียนรู้ทั้ง 5 ชนิดซึ่งมีความซับซ้อนค่อนข้างมาก แต่ในกรณีระบบ SOUCS จากรูปที่ 5.9 และ 5.10 แสดงให้เห็นว่าท้ายสุดตัวจำแนกประเภท UCS แต่ละตัวเรียนรู้เพียงแค่ 2-3 ชนิดข้อมูลเท่านั้น ดังนั้นจึงทำให้ค่าความถูกต้องในการจำแนกประเภทข้อมูลสูงขึ้นด้วยและในขณะเดียวกันนั้นจำนวนกฎที่ใช้ก็ลดน้อยลง

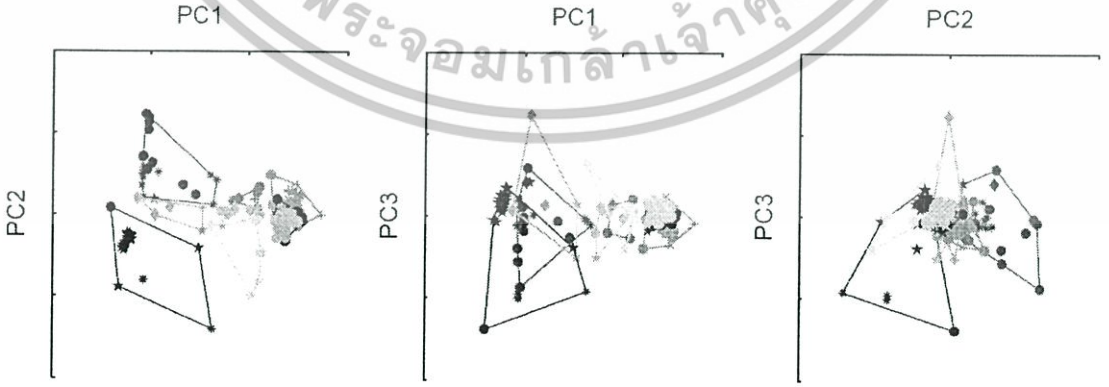
สำหรับการแสดงการแตกปัญหาในชุดข้อมูล UCI ขนาดเล็กและกลาง เราจะอาศัยเทคนิคการแสดงข้อมูลที่ถูกพัฒนาโดย Nguyen M.H. [42] เทคนิคนี้จะเป็นการแสดงข้อมูลที่มีจำนวนคุณลักษณะหรือมิติสูงโดยเลือกเฉพาะคุณลักษณะหรือองค์ประกอบพื้นฐาน (principle component) มาแสดง ในการวิเคราะห์คุณลักษณะพื้นฐานนั้นจะอาศัยหลักการวิเคราะห์องค์ประกอบพื้นฐาน (PCA - Principle Component Analysis) ที่ใช้วิเคราะห์ความสัมพันธ์ระหว่างคุณลักษณะของข้อมูลแต่ละอันกับชนิดของข้อมูล ค่าความสัมพันธ์ที่ได้จะบ่งบอกถึงอำนาจจำแนกของคุณลักษณะนั้น ในงานวิจัยนี้เราจะเลือกผลเฉพาะคุณลักษณะพื้นฐานที่มีอำนาจ

จำแนกมากที่สุดสามตัวแรก หลังจากนั้นเราจะใช้เทคนิค convex-hull เพื่อแสดงขอบเขตของกลุ่มข้อมูลโดยแสดงอยู่ในรูปแบบของกราฟ 2 มิติ

รูป 5.11 แสดงขอบเขตการตัดสินใจของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS เมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 2×2 ในชุดข้อมูล glass ชุดข้อมูลนี้จะเป็นชุดข้อมูลที่มี 9 คุณลักษณะ 7 ประเภทข้อมูล 214 ชุดข้อมูล สัญลักษณ์ของข้อมูลจะถูกแทนแตกต่างกันไปขึ้นอยู่กับประเภทข้อมูล สังเกตเห็นว่าในกรณีที่ใช้ตัวจำแนกประเภทแบบตัวเดียวจะต้องเรียนรู้ทั้งหมด 7 คลาส 214 ชุดข้อมูล แต่ในการเรียนรู้ของระบบ SOUCS ระบบจะทำการแตกปัญหาไปยังตัวจำแนกประเภท UCS ย่อยแต่ละตัวให้รับผิดชอบแตกต่างกันไป จากรูปตัวจำแนกประเภท UCS แต่ละตัวในระบบจะรับผิดชอบมากที่สุด 5 ประเภทข้อมูลและน้อยที่สุดคือ 2 ประเภทข้อมูล รูปที่ 5.12 แสดงขอบเขตการตัดสินใจของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS เมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 3×3 ในชุดข้อมูล glass จะเห็นได้ว่าระบบสามารถแตกปัญหาได้มากกว่าเมื่อใช้แผนภาพขนาด 2×2 ระบบจะรับผิดชอบมากที่สุดเพียง 3 ประเภทข้อมูลและน้อยที่สุดเพียง 1 ประเภทข้อมูล

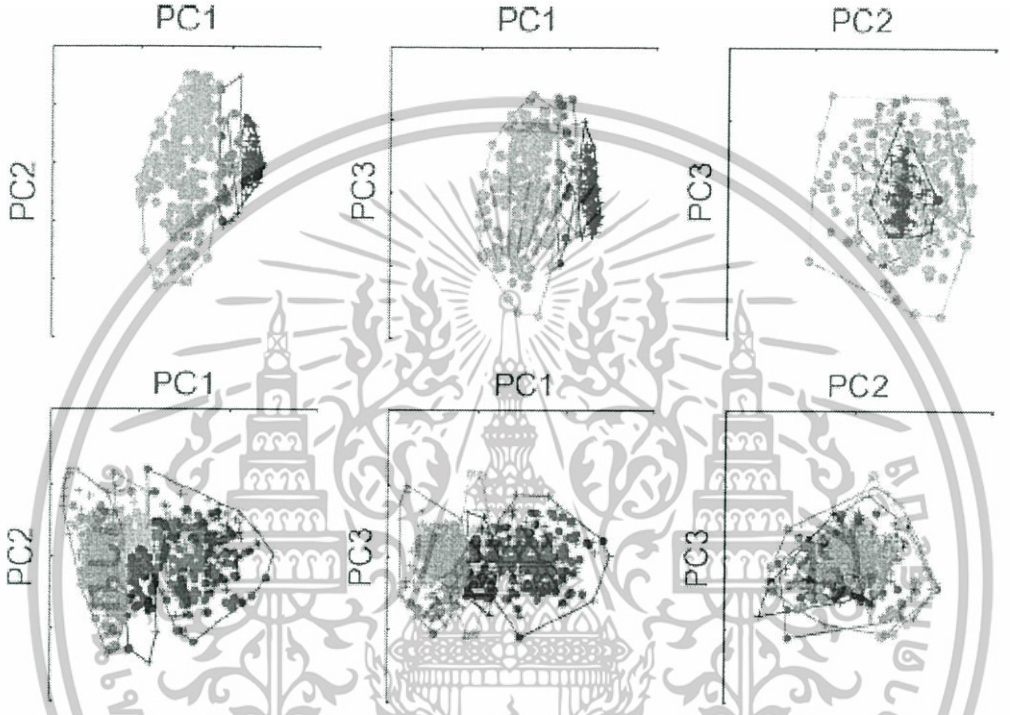


รูปที่ 5.11 ลักษณะการแตกปัญหาที่เกิดขึ้นเมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 2×2 กับข้อมูลขนาดเล็กและกลาง GLASS

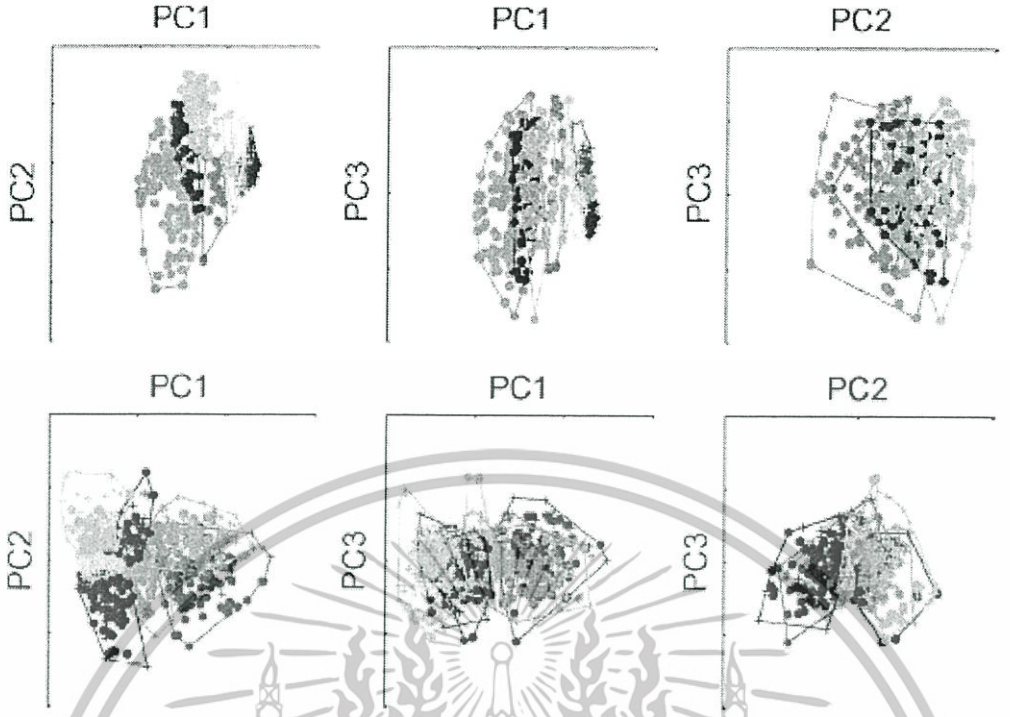


รูปที่ 5.12 ลักษณะการแตกปัญหาที่เกิดขึ้นเมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 3×3 กับข้อมูลขนาดเล็กและกลาง GLASS

รูป 5.11 และ 5.12 แสดงขอบเขตการตัดสินใจของตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS เมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 2×2 และ 3×3 ในชุดข้อมูล breast-w และ diabetes ตามลำดับ ด้วยเทคนิคการวิเคราะห์องค์ประกอบพื้นฐาน PCA ร่วมกับการแสดงข้อมูลแบบ convex-hull เราสามารถสังเกตได้ว่าความซับซ้อนในการเรียนรู้ของระบบ SOUCS ลดลงเมื่อเทียบกับการใช้ตัวจำแนกประเภท UCS เพียงตัวเดียว นอกจากนี้ยังทำให้ประสิทธิภาพในการจำแนกประเภทสูงขึ้นด้วย



รูปที่ 5.13 ลักษณะการแตกปัญหาที่เกิดขึ้นเมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 2×2 กับข้อมูลขนาดเล็กและกลาง BREAST-W (บน) และ DIABETES(ล่าง)



รูปที่ 5.14 ลักษณะการแตกปัญหาที่เกิดขึ้นเมื่อใช้แผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 3×3 กับข้อมูลขนาดเล็กและกลาง BREAST-W (บน) และ DIABETES(ล่าง)

5.4.2 การทดลองที่ 2 เปรียบเทียบระหว่างตัวจำแนกประเภท ANN และระบบ SOANN

ในการทดลองที่สองเราทำการทดลองเปลี่ยนตัวจำแนกประเภทพื้นฐานในระบบ SOUCS จากตัวจำแนกประเภท UCS เป็น ANN โดยให้ชื่อใหม่ว่า "Self-Organize Artificial Neural Network - SOANN" เพื่อที่จะทดสอบประสิทธิภาพของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พในฐานะตัวพรีเกทที่ใช้ในการแตกปัญหา จากที่กล่าวในบทที่ 4 การเปลี่ยนตัวจำแนกประเภทจำเป็นที่จะต้องเป็นตัวจำแนกประเภทเรียนรู้แบบออนไลน์เช่น ตัวจำแนกประเภท ANN เราไม่สามารถเปลี่ยนเป็นตัวจำแนกประเภทแบบออฟไลน์เช่น ตัวจำแนกประเภทต้นไม้ได้

การกำหนดแผนภาพของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พของระบบ SOANN จะเหมือนกับระบบ SOUCS โดยใน SOANN-1 ใช้แผนภาพขนาด 2×2 และ SOANN-2 ใช้แผนภาพขนาด 3×3 ในส่วนการกำหนดขนาดของ ANN ชุดข้อมูลขนาดเล็กและกลาง จำนวนโหนดในชั้นซ่อนจะถูกกำหนดเป็นจะถูกกำหนดเป็น 5 โหนด และ 20 โหนดสำหรับชุดข้อมูล Forest Covertype

ตารางที่ 5.10 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดเล็กและกลาง (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOANN ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05)

ชุดข้อมูล	ANN	SOANN-1	SOANN-2
balance-scale	0.821±0.005	0.862±0.015↑	0.865±0.008↑
breast-w	0. 961±0.015	0.964±0.018	0.973±0.017↑
bupa	0.736±0.053	0.701±0.038↓	0.681±0.054↓
diabetes	0.768±0.050	0.764±0.055	0.745±0.062
glass	0.658±0.083	0.650±0.095	0.623±0.084
iris	0.967±0.047	0.960±0.047	0.953±0.045
segment	0.948±0.015	0.958±0.010↑	0.955±0.015
tao	0.918±0.008	0.915±0.010	0.940±0.015↑
vehicle	0.785±0.051	0.807±0.046	0.786±0.048
wine	0.983±0.027	0.978±0.029	0.977±0.029

ตารางที่ 5.10 แสดงค่าความถูกต้องของตัวจำแนกประเภท ANN แบบตัวเดียว เปรียบเทียบกับระบบ SOANN จะเห็นว่าระบบ SOANN ให้ค่าความถูกต้องที่ดีกว่าการใช้ตัวจำแนกประเภท ANN แบบเดี่ยวสี่ชุดข้อมูลจากทั้งหมดสิบชุดข้อมูล ให้ค่าความถูกต้องเทียบเท่าอีกห้าชุดข้อมูลและให้ค่าความถูกต้องด้อยกว่าหนึ่งชุดข้อมูล จากการทดลองนี้แสดงให้เห็นว่าการใช้เซลฟ์ออร์แกนไนซ์ซิ่งแม็พเป็นตัวพรีเกทนั้นสามารถที่จะนำไปใช้งานกับตัวจำแนกประเภทอื่นได้ โดยให้ค่าความถูกต้องในการจำแนกประเภทอยู่ในเกณฑ์ที่น่าพอใจ

การทดลองถัดไปเราได้ทดสอบระบบ SOANN กับชุดข้อมูลขนาดใหญ่ Forest-Covertype จากตารางที่ 5.11 แสดงให้เห็นว่าระบบ SOANN-1 ให้ค่าความถูกต้องทีดีกว่าการใช้ตัวจำแนกประเภท ANN แบบตัวเดียว

ตารางที่ 5.11 ชุดข้อมูลใหญ่ที่ใช้ทดสอบ (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05)

ชุดข้อมูล	ANN	SOANN-1	SOANN-2
Forest-Covertype	0.739±0.004	0.742±0.005↑	0.735±0.006

เมื่อนำผลการทดลองกับฐานข้อมูลขนาดใหญ่ Forest-Covertype ไปเปรียบเทียบกับผลการทดลองที่ได้จากตัวจำแนกประเภทอื่นในงานวิจัยอื่นอีกสิ่งานวิจัย [63-66] ดังตารางที่ 5.12 จะเห็นได้ว่าค่าความถูกต้องที่ได้รับจากทั้งระบบ SOUCS และ SOANN ได้ค่าความถูกต้องที่ดีกว่าถึงสามจากสิ่งานวิจัย (อย่างไรก็ตามผลการเปรียบเทียบนี้เป็นเพียงการเปรียบเทียบอย่างคร่าว ๆ เพื่อแสดงให้เห็นถึงความสามารถของระบบ SOUCS และ SOANN ทั้งนี้เนื่องจากสภาพแวดล้อมของการทดลองแตกต่างกันในแต่ละงานวิจัย) อย่างไรก็ตามในงานวิจัยการใช้งานนิวรอนเน็ตเวิร์กร่วมกันหลายตัว(Neural ensemble) [66] ที่ให้ค่าความถูกต้องสูงกว่านั้นใช้นิวรอนเน็ตเวิร์ก 20 ตัวโดยที่แต่ละตัวใช้โหนดถึง 40 โหนดในชั้นซ่อน ซึ่งใช้เวลาในการเรียนรู้ค่อนข้างสูง

ในการเปรียบเทียบค่าความถูกต้องในการจำแนกประเภทของฐานข้อมูลขนาดเล็กและกลาง และฐานข้อมูลขนาดใหญ่ระหว่างตัวจำแนกประเภท UCS และระบบ SOUCS (ตารางที่ 5.5 และตารางที่ 5.9) กับ ตัวจำแนกประเภท ANN และระบบ SOANN (ตารางที่ 5.10 และ ตารางที่ 5.11) นั้นจะเห็นว่าค่าความถูกต้องของตัวจำแนกประเภท ANN และระบบ SOANN นั้นให้ค่าสูงกว่าตัวจำแนกประเภท UCS และระบบ SOUCS ในชุดข้อมูลที่มีจำนวนคุณลักษณะและจำนวนประเภทข้อมูลค่อนข้างสูง เช่น ชุดข้อมูล bupa (จำนวนคุณลักษณะ 6 จำนวนประเภทข้อมูล 22) และ vehicle (จำนวนคุณลักษณะ 17 จำนวนประเภทข้อมูล 4) รวมทั้งชุดข้อมูล Forest-Covertype (จำนวนคุณลักษณะ 54 จำนวนประเภทข้อมูล 7) ซึ่งถือเป็นแนวทางวิจัยของตัวจำแนกประเภท LCSsในอนาคตสำหรับชุดข้อมูลที่มีคุณลักษณะและจำนวนประเภทข้อมูลสูง โดยอาจจะทำการแตกคุณลักษณะออกให้แต่ละระบบเรียนรู้ได้

ตารางที่ 5.12 การเปรียบเทียบค่าความถูกต้องในการจำแนกประเภทของระบบ SOUCS และ SOANN กับงานวิจัยอื่นๆ ในชุดข้อมูลขนาดใหญ่ FOREST-COVERTYPE

ระบบอื่น	ค่าความถูกต้อง
BagCGPC [63]	0.619
BoostCGPC [64]	0.664

Neural Network Ensemble [66]	0.771
ANNCAD [65]	0.710
UCS(N=6400)	0.729
SOUCS-1(N=3200)	0.728
SOUCS-2(N=3200)	0.725
SOUCS-3(N=3200)	0.724
ANN	0.739
SOANN-1	0.742
SOANN-2	0.735

5.4.3 การทดลองที่ 3 เปรียบเทียบระหว่างตัวจำแนกประเภท UCS ระบบ SOUCS และระบบ GUCS

จากข้อจำกัดของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พในเรื่องของลักษณะการเชื่อมโยงโครงข่ายที่ส่งผลกระทบทำให้เกิดโหนดตายดังที่กล่าวในตอนต้นของบทนี้ การทดลองนี้เราจะทำการเปลี่ยนตัวพรีเกทจากเซลฟ์ออร์แกนไนซ์ซิ่งแม็พเป็นโกลว์อิ่งนิวรอนแก๊ส โดยจะทำการทดลองกับชุดข้อมูลขนาดเล็กและกลาง และชุดข้อมูลขนาดใหญ่

ในการกำหนดจำนวนโหนดสูงสุดของโกลว์อิ่งนิวรอนแก๊สเราจะกำหนดจำนวนโหนดเป็น 2, 4, 6, 9 และ 12 โดยตั้งชื่อระบบเป็น GUCS-2 GUCS-4 GUCS-6 GUCS-9 และ GUCS-12 ตามลำดับเพื่อศึกษาผลกระทบของจำนวนโหนดที่มีต่อค่าความถูกต้องในการจำแนกประเภท โดยที่เรายังคงรักษาผลรวมจำนวนกฎในแต่ละระบบย่อยของ GUCS ให้เท่ากับการใช้ตัวจำแนกประเภท UCS แบบตัวเดียว นั้นคือจำนวนประชากร N=3200 N=1600 N=1067 N=711 และ N=534 สำหรับระบบ เป็น GUCS-2 GUCS-4 GUCS-6 GUCS-9 และ GUCS-12 ตามลำดับ

ตารางที่ 5.13 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดเล็กและกลาง (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ GUCS ดีกว่าตัวจำแนกประเภท UCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ GUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=6400 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ GUCS-2 N=1600 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ GUCS-4 N=1067 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ GUCS-6 N=711 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ GUCS-9 และ N=534 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ GUCS-12)

ชุดข้อมูล	UCS	GUCS-2 (N=3200)	GUCS-4 (N=1600)	GUCS-6 (N=1067)	GUCS-9 (N=711)	GUCS-12 (N=534)
balance-scale	0.815±0.027	0.817±0.044	0.855±0.005↑	0.857±0.015↑	0.853±0.034↑	0.811±0.023
breast-w	0. 961±0.015	0.962±0.022	0.964±0.016	0.972±0.027↑	0.971±0.016↑	0.964±0.016
bupa	0.681±0.053	0.697±0.021	0.712±0.034	0.719±0.012↑	0.718±0.025↑	0.692±0.034
diabetes	0.766±0.056	0.753±0.016↓	0.768±0.022	0.769±0.043	0.763±0.038	0.758±0.032↓
glass	0.704±0.055	0.706±0.056	0.692±0.074	0.694±0.036	0.684±0.029	0.662±0.038↓
iris	0.967±0.047	0.953±0.010	0.953±0.010	0.967±0.047	0.953±0.010	0.953±0.010
segment	0.955±0.011	0.943±0.011↓	0.951±0.020	0.953±0.020	0.947±0.020↓	0.941±0.043↓
tao	0.880±0.012	0.940±0.021↑	0.956±0.016↑	0.953±0.014↑	0.956±0.016↑	0.942±0.024↑
vehicle	0.715±0.031	0.701±0.025	0.681±0.026↓	0.690±0.026↓	0.678±0.026↓	0.661±0.043↓
wine	0.966±0.047	0.964±0.038	0.965±0.028	0.964±0.016	0.962±0.021	0.958±0.017

ตารางที่ 5.13 แสดงค่าความถูกต้องในการจำแนกประเภทข้อมูลจะสังเกตเห็นว่าเมื่อเพิ่มโหนดของโกลว์อิ่งนิวรอนแก๊สจาก 2 โหนดเป็น 4 โหนดความถูกต้องในการจำแนกประเภทจะเพิ่มขึ้นอย่างเห็นได้ชัด โดยเฉพาะอย่างยิ่งในชุดข้อมูล balance-scale และ tao แต่ในชุดข้อมูล vehicle ค่าความถูกต้องในการจำแนกประเภทจะตกลงเล็กน้อย เมื่อเพิ่มโหนดของโกลว์อิ่งนิวรอนแก๊สจาก 4 โหนดเป็น 6 โหนดค่าความถูกต้องในการจำแนกประเภทเพิ่มขึ้นอย่างเห็นได้ชัดในชุดข้อมูล breast-w และ iris แสดงให้เห็นถึงความสำคัญในการแตกปัญหาเพิ่มขึ้น เมื่อเพิ่มโหนดของโกลว์อิ่งนิวรอนแก๊สจาก 6 โหนดเป็น 9 โหนดค่าความถูกต้องในการจำแนกประเภทของข้อมูลทุกชุดเริ่มที่จะคงที่หรือมีการเปลี่ยนแปลงน้อย เมื่อเพิ่มโหนดของโกลว์อิ่งนิวรอนแก๊สจาก 9 โหนดเป็น 12 โหนดค่าความถูกต้องในการจำแนกประเภทในชุดข้อมูลเกือบทั้งหมดจะลดลง

จากการสังเกตในการทดลองเมื่อเราเพิ่มจำนวนโหนดเป็น 12 โหนดหรือมากกว่านั้น โกลว์อิ่งนิวรอนแก๊สจะมีการเพิ่มขึ้นของกระบวนการลบโหนดและเพิ่มโหนด ทำให้เกิดตัวจำแนกประเภท UCS ใหม่ขึ้นบ่อยครั้ง ถึงแม้ว่าประชากรกฎของตัวจำแนกประเภท UCS ใหม่ที่เกิดขึ้นจะคัดลอกมาจากโหนดที่ใกล้ที่สุดเพื่อลดเวลาในการเรียนรู้ก็ตาม แต่ก็ยังต้องอาศัยเวลาในการปรับเงื่อนไขของกฎรวมถึงค่าพารามิเตอร์ต่าง ๆ ประจำกฎจึงอาจเป็นสาเหตุที่ทำให้ค่าความถูกต้องลดลงเมื่อเทียบที่จำนวนรอบของการเรียนรู้เท่ากัน

ตารางที่ 5.14 ค่าความถูกต้องและส่วนเบี่ยงเบนมาตรฐานของชุดข้อมูลขนาดเล็กและกลาง (↑ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ GUCS-9 ดีกว่าระบบ SOUCS และ ↓ หมายถึงประสิทธิภาพในการจำแนกประเภทของระบบ SOUCS ด้อยกว่าตัวจำแนกประเภท UCS อย่างมีนัยยะสำคัญที่ระดับความเชื่อมั่น 0.05 – N=6400 สำหรับตัวจำแนกประเภท UCS แบบตัวเดียว N=3200 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-1 และ N=711 สำหรับตัวจำแนกประเภท UCS แต่ละตัวในระบบ SOUCS-2)

ชุดข้อมูล	SOUCS (*N*=711)	GUCS-9 (*N*=711)
balance-scale	0.854±0.015	0.853±0.034
breast-w	0.973±0.016	0.971±0.016
bupa	0.719±0.035	0.718±0.025
diabetes	0.761±0.047	0.763±0.038
glass	0.682±0.012	0.684±0.029
iris	0.947±0.042	0.953±0.010
segment	0.949±0.050	0.947±0.020

tao	0.956±0.016	0.956±0.016
vehicle	0.669±0.041	0.678±0.026
wine	0.949±0.050	0.962±0.021

ตารางที่ 5.14 แสดงการเปรียบเทียบค่าความถูกต้องในชุดข้อมูลขนาดเล็กและกลางของระบบ SOUCS ที่ใช้แผนภาพของเซลฟ์ออร์แกนไนซ์ซิ่งแม็พขนาด 3×3 และระบบ GUCS-9 ที่กำหนดจำนวนโหนดสูงสุด 9 โหนด จะเห็นว่าค่าความถูกต้องของระบบทั้งสองไม่แตกต่างกันอย่างมีนัยสำคัญที่ระดับความเชื่อมั่น 0.05

จากสมมติฐานเบื้องต้นของการทดลองที่ในการนำโกลว์อิ่งนิวรอนแก๊สมาใช้แทนเซลฟ์ออร์แกนไนซ์ซิ่งแม็พในการจัดกลุ่มเพื่อไม่ให้เกิดโหนดตายขึ้น การเกิดโหนดตายในเซลฟ์ออร์แกนไนซ์ซิ่งแม็พโดยปกติคือ ไม่มีอินพุตไปตกยังโหนดนั้น ทำให้อินพุตไปตกที่โหนดใดโหนดหนึ่งเพิ่มขึ้น ส่งผลให้จำนวนกฎของตัวจำแนกประเภท UCS มีจำนวนมากขึ้น การเรียนรู้ที่ตัวจำแนกประเภท UCS ตัวนั้นใช้เวลามากขึ้น ในบางครั้งอาจจะทำให้ค่าความถูกต้องของระบบลดลงด้วยเนื่องจากการกำหนดจำนวนกฎน้อยเกิดไป เมื่อเราเปลี่ยนตัวพรีเททเป็นโกลว์อิ่งนิวรอนแก๊สซึ่งมีกลไกในการป้องกันการเกิดโหนดตาย ดังนั้นตัวจำแนกประเภท UCS แต่ละตัวที่ผูกติดแต่ละโหนดในระบบ GUCS จะมีจำนวนกฎใกล้เคียงกัน ส่งผลให้ความเร็วในการเรียนรู้นั้นเร็วขึ้นด้วย

บทที่ 6
สรุปผลการทดลอง

6.1 สรุปผลการดำเนินงานวิจัยและแนวทางในการพัฒนาต่อ

ระบบจำแนกประเภท UCS (เป็นระบบ LCS ที่ได้รับการปรับปรุงเพื่องานทางด้านการจำแนกประเภทโดยเฉพาะ) เป็นจักรกลเรียนรู้ที่อยู่ในรูปแบบของกฎที่เราสามารถทำความเข้าใจได้ โดยธรรมชาติของระบบจำแนกประเภทแบบกฎจะสร้างกฎออกมาเป็นจำนวนมาก โดยเฉพาะเมื่อข้อมูลมีปริมาณมากและมีความไม่เป็นเชิงเส้นสูง จำนวนกฎที่เพิ่มขึ้นทำให้ระบบต้องใช้เวลาในการประมวลผลข้อมูลเพิ่มขึ้นด้วย เนื่องจากในแต่ละครั้งที่มีข้อมูลเข้ามาระบบจะต้องทำการค้นหากฎทั้งหมดเพื่อหากฎที่สอดคล้อง

วิทยานิพนธ์ฉบับนี้นำเสนอโครงข่ายงานระบบสร้างกฎแบบปรับตัวได้ 2 ระบบคือ SOUCS และ GUCS ทั้งสองระบบจะอาศัยเซลฟ์ออร์แกนไนซ์ซิ่งแม็พและโกลว์อิ่งนิวรอนแก๊สเป็นพรีเกทตามลำดับ พรีเกทมีหน้าที่ในการแตกปัญหาใหญ่หนึ่งปัญหาออกเป็นปัญหาย่อยหลายปัญหาที่มีความซับซ้อน และถูกเรียนรู้ด้วยตัวจำแนกประเภทแต่ละตัวแยกจากกัน ซึ่งสามารถลดเวลาที่ใช้ในการเรียนรู้เนื่องจากระบบ UCS แต่ละตัวจัดการกับปัญหาที่มีขนาดเล็กกว่าและมีความซับซ้อนน้อยกว่า อีกทั้งยังสามารถเพิ่มประสิทธิภาพในการจำแนกประเภทได้ด้วย

ระบบ SOUCS เป็นโครงข่ายงานแรกที่นำเสนอ โดยจะต้องมีการกำหนดขนาดของแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พก่อน กล่าวคือถ้ากำหนดขนาดของแผนภาพเซลฟ์ออรแกนไนซ์ซิ่งแม็พเล็กไปจะทำให้กระบวนการแตกปัญหาทำงานได้ไม่เต็มประสิทธิภาพ แต่ถ้ากำหนดขนาดใหญ่เกินไปจะทำให้เกิดโหนดตายในขณะการเรียนรู้ (โหนดที่ไม่มีข้อมูลตก) ซึ่งทำให้ตัวจำแนกประเภท UCS ที่ผูกติดอยู่กับโหนดที่ใกล้เคียงต้องรับภาระในการเรียนรู้เพิ่มขึ้น ส่งผลให้จำนวนกฎของตัวจำแนกประเภท UCS ตัวนั้นมีปริมาณสูงกว่าตัวอื่น ๆ และทำให้ความเร็วในการเรียนรู้โดยรวมของระบบลดลง แต่อย่างไรก็ตามระบบ SOUCS ก็สามารถแสดงให้เห็นถึงความเป็นไปได้ที่จะเชื่อมโยงกันระหว่างกระบวนการจัดกลุ่มข้อมูลและระบบ UCS ทั้งในแง่ของประสิทธิภาพในการจำแนกประเภทข้อมูลและช่วยลดเวลาที่ใช้ในการเรียนรู้

โครงข่ายงานถัดมาที่นำเสนอคือ GUCS ซึ่งใช้โกลว์อิ่งนิวรอนแก๊สเป็นตัวพรีเกทแทนเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ อัลกอริธึมการเรียนรู้ของโกลว์อิ่งนิวรอนแก๊สทำหน้าที่กำหนดการเชื่อมโยงเองอัตโนมัติรวมถึงทำการลบโหนดที่ไม่จำเป็นและเพิ่มโหนดในส่วนที่มีข้อมูลหนาแน่นเองอัตโนมัติ จึงสามารถช่วยแก้ปัญหาในการออกแบบแผนภาพและการเกิดโหนดตายที่เป็นข้อด้อยสำคัญของแผนภาพเซลฟ์ออร์แกนไนซ์ซิ่งแม็พ

เพื่อให้เข้าใจถึงพฤติกรรมของระบบที่นำเสนอ เราได้ทดสอบระบบ SOUCS/GUCS กับข้อมูลสังเคราะห์และข้อมูลจริงหลายชุดข้อมูลทั้งข้อมูลขนาดเล็กจนไปถึงข้อมูลขนาดใหญ่มีจำนวนข้อมูลมากกว่าห้าแสนเรคอร์ด ผลการทดลองแสดงให้เห็นว่าโครงข่ายงานที่นำเสนอให้ค่าความถูกต้องในการจำแนกประเภทเทียบเท่าหรือดีกว่าการใช้งานตัวจำแนกประเภท UCS และยังใช้เวลาเรียนรู้ข้อมูลลดลงสูงสุดเกินกว่าครึ่งของเวลาการเรียนรู้ของตัวจำแนกประเภท UCS แบบใช้ตัวเดียว

ในการแสดงขอบเขตที่ตัวจำแนกประเภท UCS แต่ละตัวในโครงข่ายงาน เราได้ใช้เทคนิคการแสดงข้อมูลที่ถูกพัฒนาโดย Nguyen M.H. และใช้เทคนิค convex-hull เพื่อแสดงขอบเขตของกลุ่มข้อมูลโดยแสดงอยู่ในรูปแบบของกราฟ 2 มิติ จากกราฟแสดงให้เห็นว่าตัวจำแนกประเภทแต่ละตัวเรียนรู้ข้อมูลที่มีความซับซ้อนน้อยลงในแง่ของจำนวนชนิดและจำนวนข้อมูลเมื่อเปรียบเทียบกับการใช้งานตัวจำแนกประเภท UCS เพียงตัวเดียว

นอกจากนี้แล้วเรายังได้ทดสอบโครงข่ายงานที่นำเสนอโดยการเปลี่ยนตัวจำแนกประเภท UCS เป็น ANN ผลปรากฏว่าสามารถให้ค่าประสิทธิภาพในการจำแนกประเภทที่สูงขึ้นในบางชุดข้อมูล อีกทั้งยังใช้จำนวนโหนดซ่อนน้อยกว่าหรือเท่ากับเมื่อเปรียบเทียบต่อโครงข่าย

แนวทางพัฒนาโครงข่ายงานที่นำเสนอต่อในอนาคต เราสามารถพัฒนาในส่วนของการปรับขนาดประชากรหรือกฎของตัวจำแนกประเภท UCS แต่ละตัวที่ผูกติดอยู่กับโหนดของโกลว์อิ่งนิวรอนแก๊ส โดยจะทำการปรับขนาดของประชากรให้เหมาะสมกับข้อมูลที่ตกลงยังโหนดนั้นและให้มีความเป็นอิสระต่อตัวจำแนกประเภท UCS ตัวอื่น เพื่อให้ระบบมีความเร็วในการเรียนรู้เพิ่มขึ้น

เอกสารอ้างอิง

[1] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, "Knowledge Discovery in Databases: An Overview," *AI Magazine,* vol. 13, no. 3, pp. 57-70, 1992.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *Communications of the ACM,* vol. 39, no. 11, pp. 27-34, 1996.

[3] D. J. Hand, H. Mannila, and P. Smyth, "Principle of Data Mining," *MIT Press*, 2001.

[4] J. Han, and M. Kamber, *Data Mining: Concepts and Techniques*, 2 ed.: Morgan Kaufmann, 2006.

[5] D. H. Wolpert, and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation,* vol. 1, no. 1, pp. 67-82, 1996.

[6] H. H. Dam, H. A. Abbass, C. Lokan *et al.*, "Neural-Based Learning Classifier Systems," *IEEE Transactions on Knowledge and Data Engineering,* vol. 20, no. 1, pp. 26-39, 2008.

[7] T. M. Mitchell, *Machine learning*: McGraw-Hill, 1997.

[8] R. Setiono, W. K. Leow, and J. M. Zurada, "Extraction of Rules from Artificial Neural Networks for Nonlinear Regression," *IEEE Transactions on Neural Networks,* vol. 13, pp. 564-577, 2002.

[9] G. G. Towell, and J. W. Shavlik, "Extracting refined rules from knowledge-based neural networks," *Machine Learning.* pp. 71-101.

[10] R. Xu, and W. D. II, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks,* vol. 16, no. 3, pp. 645-678, 2005.

[11] G. J. McLachlan, and K. E. Basford., *Mixture Models: inference and application to clustering*: New York, N.Y. : M. Dekker, 1988.

[12] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algoritms*: Plenum Press, New York, 1981.

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[14] S. W. Wilson, "Classifier Fitness Based on Accuracy," *Evolutionary Computation,* vol. 3, no. 2, pp. 149-175, 1995.

[15] S. W. Wilson, "Generalization in the XCS Classifier System," *Proceedings of the Third Annual Conference on Genetic Programming*, Morgan Kaufmann, 1998, pp. 665-674.

[16] M. Studley, and L. Bull, "X-TCS: accuracy-based learning classifier system robotics," *Congress on Evolutionary Computation.* pp. 2099-2106

[17] Y. J. Cao, N. Ireson, L. Bull *et al.*, "Design of a Traffic Junction Controller Using Classifier Systems and Fuzzy Logic," *Fuzzy Days.* pp. 342-353.

[18] S. Saxon, and A. Barry, "XCS and the Monk's Problems," *Learning Classifier Systems, From Foundations to Applications*, Springer-Verlag, 2000, pp. 223-242.

[19] E. Bernadó-Mansilla, X. Llor\`a, and J. M. Garrell-Guiu, "XCS and GALE: a Comparative Study of Two Learning Classifier Systems with Six Other Learning Algorithms on Classification Tasks," *Proceedings of the 4th International Workshop on Learning Classifier Systems (IWLCS-2001)*, 2001, pp. 337-341.

[20] A. Asuncion, and D. J. Newman. "UCI Machine Learning Repository."

[21] P. W. Dixon, D. Corne, and M. J. Oates, "A Preliminary Investigation of Modified XCS as a Generic Data Mining Tool," *Advances in Learning Classifier Systems: 4th International Workshop, IWLCS*, Springer-Verlag Berlin Heidelberg, 2001, pp. 133-150.

[22] M. V. Butz, "Rule-based Evolutionary Online Learning Systems: Learning Bounds, Classification, and Prediction," University of Illinois at Urbana-Champaign, 2004.

[23] E. Bernadó-Mansilla, and J. M. Garrell-Guiu, "Accuracy-based learning classifier systems: models, analysis and applications to classification tasks," *Evolutionary Computation*, vol. 11, no. 3, pp. 209-238, 2003.

[24] A. J. Bagnall, and G. C. Cawley, "Learning Classifier Systems for Data Mining : A Comparison of XCS with other Classifiers for the Forest Cover Dataset," *Proceedings of the IEEE/INNS International Joint Conference on Artificial Neural Networks (IJCNN-2003)*, 2003, pp. 1802-1807.

[25] H. H. Dam, P. Rojanavasu, H. A. Abbass *et al.*, "Distributed learning classifier systems," *Learning Classifier Systems in Data Mining*, Springer-Verlag, 2007.

[26] H. H. Dam, H. A. Abbass, and C. Lokan, "DXCS: an XCS System for Distributed Data Mining," *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'05)*, 2005.

[27] L. Bull, M. Studley, T. Bagnall *et al.*, "On the use of Rule-Sharing in Learning Classifier System Ensembles," *Proceedings of IEEE Congress on Evolutionary Computation* 2005.

[28] L. Bull, M. Studley, T. Bagnall *et al.*, "Learning Classifier System Ensembles With Rule-Sharing," *IEEE Transections on Evolutionary Computation,* vol. 11, no. 4, pp. 496-502, 2007.

[29] L. Bull, "Two simple learning classifier systems," *Foundations of Learning Classifier Systems*, L. Bull and T. Kovac, eds., Berlin, Germany: Springer-Verlag, 2005.

[30] O. Maimon, and L. Rokach, *Decomposition Methodology for Knowledge Discovery and Data Mining: Theory and Applications*: World Scientific Publishing, 2005.

[31] W. Buntine, "Graphical Models for Discovering Knowledge," *Advances in Knowledge Discovery and Data Mining* U. F. G. Piattsky-Shapiro, P. Smyth and R. Uthurusamy, eds., pp. 59-82: AAAI/MIT Press, 1996.

[32] Anand R., Methrotra K., Mohan C.K. *et al.*, "Efficient classification for multiclass problems using modular neural networks," *IEEE Trans Neural Networks,* vol. 6, no. 1, pp. 117-125, 1995.

[33] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.

[34] Y. Freund, and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *European Conference on Computational Learning Theory*, 1995, pp. 23-37.

[35] R. A. Jacobs, M. I. Jordan, S. J. Nowlan *et al.*, "Adaptive mixtures of local experts," *Neural Computation,* vol. 3, pp. 79-87, 1991.

[36] M. I. Jordan, and R. A. Jacobs, "Hierachical mixtures of experts and the em algorithm," *Neural Computation,* vol. 6\(2\), pp. 181-214, 1994.

[37] H. A. Abbass, "Pareto Neuro-Ensembles " *Advances in Artificial Intelligence*, Springer Berlin / Heidelberg, 2003, pp. 554-566.

[38] R. Ranawana, and V. Palade, "Multi-classifier systems: Review and a roadmap for developers," *Int. J. Hybrid Intell. Syst.,* vol. 3, no. 1, pp. 35-61, 2006.

[39] J. R. Quinlan, "Bagging, Boosting, and C4.5," *In Proceedings of the Thirteenth National Conference on Artificial Intelligence.* pp. 725-730.

[40] K. Cherkauer, "Human Expert-Level Performance on a Scientific Image Analysis Task by a System Using Combined Artificial Neural Networks," *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models.* pp. 15-21.

[41] M.-H. Nguyen, H. A. Abbass, and R. I. McKay, "A Novel Mixture of Experts Model Based on Cooperative Coevolution," *Neurocomputing,* vol. 1-3, pp. 155-163, 2006.

[42] M.-H. Nguyen, H. A. Abbass, and R. I. McKay, "Analysis of CCME: Coevolutionary Dynamics, Automatic Problem Decomposition and Regularization," *IEEE Transactions on Systems, Man, Cybernetics, Part C,* vol. 38, no. 1, pp. 100-109, 2008.

[43] L. Rokach, O. Maimon, and I. Lavi, "Space Decomposition in Data Mining: A Clustering Approach," *Foundations of Intelligent Systems* Springer-Verlag, 2003, pp. 24-31.

[44] J. H. Holland, and J. S. Reitman, "Cognitive systems based on adaptive algorithms," *SIGART Bull.*, no. 63, pp. 49-49, 1977.

[45] S. F. Smith, "A Learning System Based on Genetic Adaptive Algorithms," University of Pittsburgh, 1980.

[46] K. A. De Jong, W. M. Spears, and D. F. Gordon, "Using Genetic Algorithms for Concept Learning," *Machine Learning,* vol. 13, no. 2-3, pp. 161-188, 1993.

[47] D. E. Goldberg, "Computer-Aided Gas Pipeline Operation using Genetic Algorithms and Rule Learning," The University of Michigan, 1983.

[48] X. Llor\`a, "Genetics-based machine learning using fine-gained parallelism for data mining," Enginyeria i Arquitectura La Salle. Ramon Llull University, Barcelona, Catalonia, European Union, 2002.

[49] J. Bacardit, and M. V. Butz, "Data Mining in Learning Classifier Systems: Comparing XCS with GAssist," *Learning Classifier Systems*, Springer-Verlag Berlin / Heidelberg, 2007.

[50] T. Kovacs, "XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions," *Soft Computing in Engineering Design and Manufacturing (WSC2)*, Springer-Verlag, 1997, pp. 59-68.

[51] W. S. McCulloch, and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology,* vol. 4, pp. 115-133, December, 1943.

[52] S. Haykin, *Neural Networks: A Comprehensive Foundation*: Prentice-Hall Engineering/Science/Mathematics, 1999.

[53] G. P. Zhang, "Neural networks for classification: a survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 30, no. 4, pp. 451-462, 2000.

[54] A. E. Bryson, and Y.-C. Ho, *Applied Optimal Control*: Blaisdell, New York, 1969.

[55] T. Kohonen, *Self-Organizating Maps*: New York : Springer-Verlag, 1997.

[56] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics,* vol. 43, no. 1, pp. 59-69, 1 January 1982, 1982.

[57] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," *Advances in Neural Information Processing Systems 7*. pp. 625--632.

[58] J. Holmstrom, "Growing Neural Gas Experiments with GNG, GNG with Utility and Supervised GNG," Uppsala University, 2002.

[59] E. Cantuz-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, Boston, MA, USA: Kluwer, 2000.

[60] S. W. Wilson, "Mining Oblique Data with XCS," *Proceedings of the Third International Workshop (IWLCS-2000), Lecture Notes in Artificial Intelligence*, 2002, pp. 158-174.

[61] R. Picard, and D. Cook, "Cross-Validation of Regression Models " *Journal of the American Statistical Association* vol. 79, no. 387 pp. 575-583. , 1997.

[62] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Computation,* vol. 10, pp. 1895-1923, 1998.

[63] G. Folino, C. Pizzuti, and G. Spezzano, "Ensemble techniques for Parallel Genetic Programming based Classifiers," *Genetic Programming, Proceedings of EuroGP'2003*, Springer-Verlag, 2003, pp. 59-69.

[64] G. Folino, C. Pizzuti, and G. Spezzano, "Boosting Technique for Combining Cellular GP Classifiers," *Genetic Programming, Proceedings of EuroGP'2004* Springer-Verlag, 2004, pp. 47-56.

[65] Y.-N. Law, and C. Zaniolo, "An Adaptive Nearest Neighbor Classification Algorithm for Data Streams," *PKDD*, Springer, 2005, pp. 108-120.

[66] K. Peng, Z. Obradovic, and S. Vucetic, "Towards Efficient Learning of Neural Network Ensembles from Arbitrarily Large Datasets," *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, 2004, pp. 623-627.

ภาคผนวก

ภาคผนวก ก
งานวิจัยที่ได้รับการตีพิมพ์

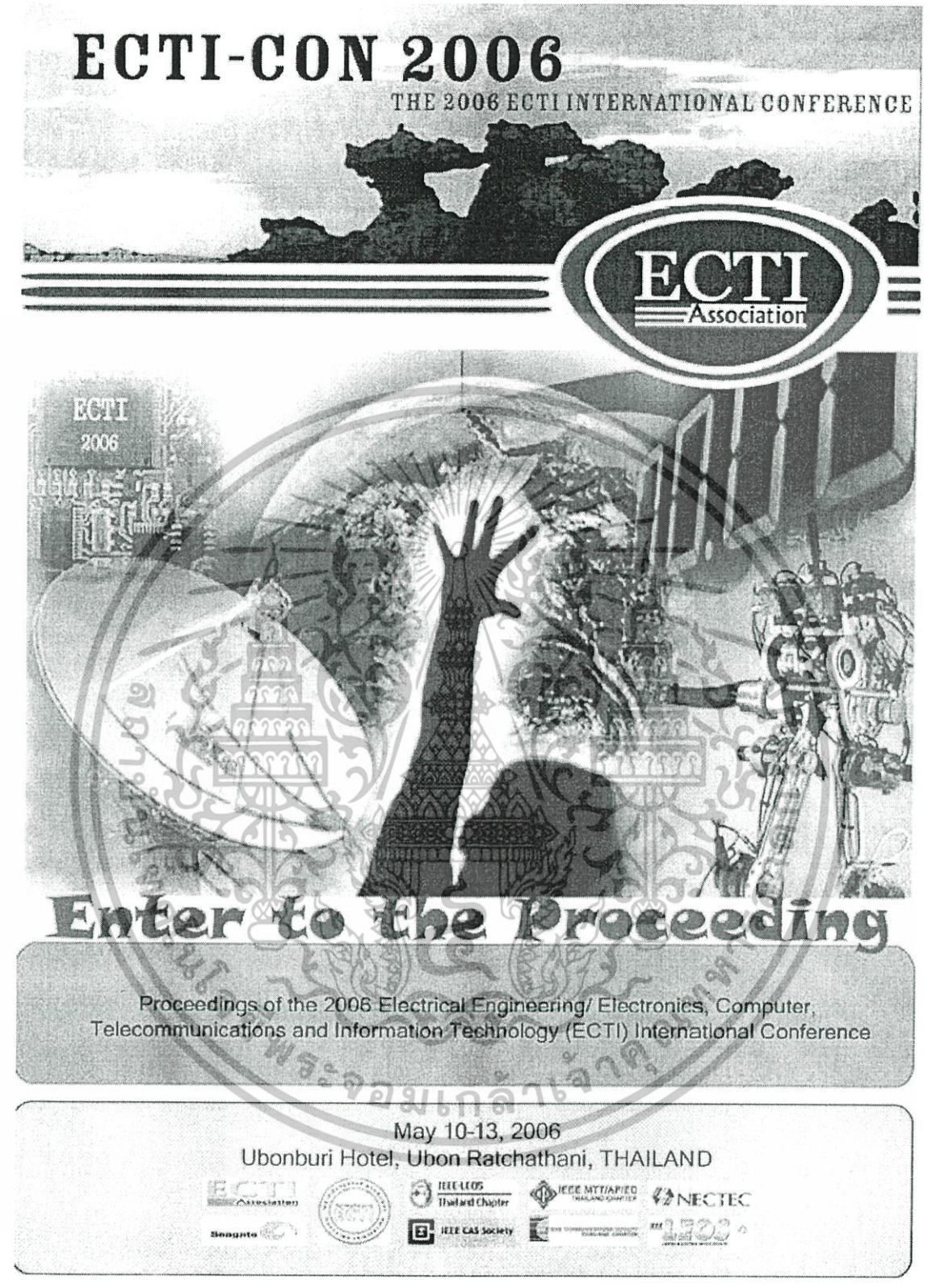
วารสารนานาชาติ

1. P. Rojanavasu, H. H. Dam, H. A. Abbass, Lokan, C. , Pinngern, O., "A Self-Organize, Distributed and Adaptive Rule-Based Inductive System," IEEE Transaction Neural Networks, vol. 20, no. 3, pp. 446-459, March, 2009.

งานประชุมวิชาการนานาชาติ

1. P. Rojanavasu, Srinil, P., Tamee, K., Pinngern, O., "Rules Reduction of XCS using Genetic Algorithms," Proceeding of the 2006 ECTI International Conference. pp. 517-520, 2006.
2. P. Rojanavasu, Attachoo, B., Pinngern, O., "Growing Rule-Based Induction System," Proceeding of 2009 the 2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT) pp. 91-101, 2009.

หมายเหตุ ผู้ที่สนใจสามารถอ่านงานวิจัยที่ได้รับการตีพิมพ์ทุกชิ้น(อยู่ในรูปแบบไฟล์ pdf ที่มีคุณภาพสูง) ได้จากซีดีรอมที่แนบกับวิทยานิพนธ์ฉบับนี้



Rules Reduction of XCS using Genetic Algorithms

Pornthep Rojanavasu[1] Phaitoon Srinil[2] Kreangsak Tamee[3] Ouen Pinngern[1]

[1]Department of Computer Engineering, Faculty of Engineering, Research Center for Communication and Information Technology, King Mongkut's Institute of Technology Ladkrabang Bangkok, Thailand 10520.
[2]Faculty of Science and Art, Burapha University, Chanbutaburi, Thailand 22170
[3]Department of Computer Science and Information Technology, Faculty of Science, Naresuan University, Phisanulok, Thailand 65000
s8060022@kmitl.ac.th, phaitoon@buu.ac.th, kreangsak1@nu.ac.th, kpouen@kmitl.ac.th

ABSTRACT

XCS is currently focused as recently and most powerful learning classifier system. It combines reinforcement learning and genetic algorithm to create a set of rules representing the extracted knowledge from dataset. In practice for real number XCS, we found that the discovered rules are superfluous. As a result, it is difficult to understand the extracted knowledge. In this paper, we propose a technique to reduce superfluous rules after training process. The proposed method uses genetic algorithms to find a minimum set of rules that can preserve classification accuracy. The efficiency of the proposed method is illustrated by the number of existent rules and compared with previous reduction algorithm of XCS.

Keyword Machine Learning, Learning Classifier System, Genetic Algorithms, Rules Reduction

1. INTRODUCTION

Learning Classifier System[LCS] is a machine learning system that learns iteratively from interaction with the environment by using combining reinforcement learning and genetic algorithms. The result of LCS is inform of a set of if *<state>* then *<action>* rules describing the extracted knowledge from dataset. The LCS was first described by John Holland [1]. It consists of a population of ternary rules that evolved by genetic algorithms to search the possible searching space while the reinforcement learning uses to decide the rule utility.

XCS [2,3] was developed from LCS. XCS differs from traditional LCS in that a classifier's fitness in XCS depends on the prediction of its expected payoff instead of the actual prediction itself. The key to success is using a set of maximally general rules that cover many state-action-prediction region of the problem space as possible without loss of accuracy.

In XCSR [4] uses for classify real number attributes. In these types of problems, XCSR uses a hyper-rectangle represent for each rules that lead to increase number of rules. In fact, most of rules are superfluous rules. As a result, we can reduce these rules to present human readable rule sets while preserves classification accuracy. In Dixon et al [6] proposed the algorithm try to non-qualified classifiers. A qualified classified has high experience, low error and high prediction. A useful classifier is considered for classification some traning examples.

In fact, the problem of rule reduction is finding the minimum set of rules that can preserve classification accuracy. In the multiple objective problems, it seems that genetic algorithms are more suitable. In this paper, we propose method to reduce superfluous rules of XCSR by using genetic algorithms. A set of rules in XCSR is coded into an individual in population. The fitness of each individual is computed by optimizing both maximize classification accuracy and minimize number of rules. After finish the evolve process, the individual that have maximum fitness is selected to create a set of reduced rules which represent the extracted knowledge.

This paper is organized as follows. In Section 2, we give a brief description of XCS architecture. In Section 3, we propose an adaptive method of genetic algorithms that use to remove unnecessary rules. In Section 4, provides the performance evaluation of the proposed method by applying them to the iris data and compare with previous reduction algorithm of XCSR. In Section 5, gives conclusions.

2. XCS ARCHITECTURE

XCS is a classifier system that was introduced by Wilson. In many research [4,7] shown that XCS perform well on data mining, recognition systems and many field areas.

XCS represent the extracted knowledge from dataset in a set of rules. The rule set is evaluated by means of interacting with the environment, through reinforcement learning, and is improved by a search mechanism based on genetic algorithms. In [5] Wilson developed XCSR for classify real number data. The input of system change from ternary {1,0,#} into real number vector. In following, we describe in term of XCSR as show in Fig. 1, however it's differ from XCS only input interface, mutation operator and covering algorithm.

2.1 Representation

In XCSR system, each rule consist of condition part and action part: if *<condition>* then *<action>* where the condition part is a set of interval range $\{(c_1,s_1),(c_2,s_2),..(c_m,s_n)\}$ where c is the *center* of interval

range and s is the *spread* of center, the action part is decided class. Input vector is formed by $\{x_1, x_2, \ldots, x_n\}$ where x is real number and n is number of attributes.

Evaluation of each classifier consists of three main parameters : the payoff prediction(p) is an estimate payoff for that classifier if its condition matches to the input and its action is selected, the prediction error(ε) is estimates the average error made in the predictions, the fitness (F) is estimates the accuracy of the payoff prediction given by p.

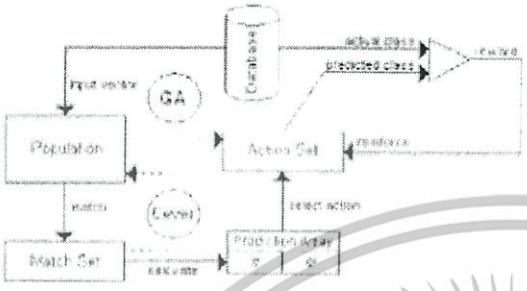


Fig. 1 The XCSR architecture for single step

2.2 Performance Component

At each time step, the system builds a match set [M] that is produced by all classifier in population [P] whose conditions are matched to input from the environment. If the size of a match set [M] is less than θ_{mna}, then covering is invoked for creates new classifiers with the same condition of current input and selected action randomly. After get match set [M], select action a_i in [M] by compute the system prediction $P(a_i)$ for an action a_i. The system prediction is compiled as a fitness weighted average of the predictions of all classifiers in [M] as follow:

$$P(a_i) = \frac{\sum_{cl_j \in [M]_{a_i}} p_j \times F_j}{\sum_{cl_j \in [M]_{a_i}} F_j} \qquad (1)$$

where $[M]_{a_i}$ represent subset of classifier of [M] with action a_i, p_j means the prediction of classifier cl_j, and F_j means the fitness of class cl_j. The system chooses the winning action based on those $P(a_i)$ and create [A] from classifier in [M] which have same action as a winning action. The selected action is acted to the environment and receives reward r.

2.3 Reinforcement Component

After received the reward r, parameters of each classifier in [A] were updated in three steps. First, prediction p is updated:

$$p = p + \beta(r - p) \qquad (2)$$

where β is the learning rate ($0 \le \beta \le 1$) usually 0.2.

Second, update prediction error:

$$\varepsilon = \varepsilon + \beta(|r - p| - \varepsilon) \qquad (3)$$

Third, updates the fitness which consists of three steps.

(i) Update classifier's accuracy:

$$k = \begin{cases} \alpha(\varepsilon / \varepsilon_0)^{-\nu}, \varepsilon \ge \varepsilon_0 \\ 1 \quad otherwise \end{cases} \qquad (4)$$

where $\varepsilon_0 > 0$ is the threshold error under which a classifier is considered to be accurate. $\alpha (0 < \alpha < 1)$ and ν control the degree of decline in accuracy if the classifier is inaccurate.

(ii) Update the relative accuracy over the action set as follow

$$k' = \frac{k}{\sum_{cl \in [A]} k_{cl}} \qquad (5)$$

(iii) Update the fitness F of each classifier as follow:

$$F = F + \beta(k' - F) \qquad (6)$$

2.4 Discover Component

On regular basis (dependent on parameter θ_{ga}), the genetic algorithm is performed to each classifier in [A]. It selects two classifiers with probability proportional to their fitness, copies them, and with probability χ performs crossover on the copies; then, with probability μ it mutates each allele. The resulting offspring are inserted into the population and two classifiers are deleted to keep the population size constant.

3. GENETIC ALGORITHMS FOR RULES REDUCTION

Genetic algorithms constitute a universal optimization method. The problem search a minimum set of rules that can preserve classification accuracy of XCS yields very large search space. As a result genetic algorithms is selected to find optimal solution of this problem. The method of rule selection was first introduced by Ishibuchi [8]. In this study, we applied GA to search a minimum set of rules for XCS. GA was operated after finishing the learning process of XCS.

3.1 Encoding of the individual

The defined individual is represented in form of binary value {0,1} to represent select or discard the regarded rule. The length of string is equal to number of individual of [P] in XCS as follow:

$$\text{INV} = \{ i_1, i_2, \ldots i_n \}, i = \{0,1\}$$

where n number of rules in XCS.

3.2 Fitness evaluation

The problem has two objectives. First, preserve or maximize classification accuracy of XCS. It means that after reduce rules the system must have the same recognition rate as before reduction for all training patterns. Second, minimize the number of rules in reduced set of rules. We combine these two objectives into a single scalar fitness functions as follow;

$$Fitness(INV_k) = reg.rate \times w_{re} - num.rules \times w_{nr} \quad (7)$$

where

reg.rate - recognition rate of patterns.
w_{re} weight of reg.rate usually positive weight.
num.rules – number of selected rules.
w_{nr} weight of num.rate usually positive weight

Both weight defined by user. We set w_{re} 100, w_{nr} 1 i.e. recognition rate is more significant than number of rule. If the recognition rate is stable, the fitness value relies on number of rules. If we can minimize a set of rules then we can get better fitness of individual.

3.3 Initial parameters & Genetic Operators

- **Initialize**: Each individuals in population N is randomly generated. This population size is fixed during the evolution process.
- **Selection**: After ranking process the population according to the individual's fitness. Each individual is selected as a parent to create new generation based on roulette wheel selection strategy to its fitness value.
- **Crossover**: Crossover operates on selected genes from parent individual and creates new offspring. We used one-point crossover to the selected parents.

INV1: $i_1 i_2 ... i_j i_{j+1} i_{j+2} ... i_n$ | INV2: $i_1 i_2 ... i_j i_{j+1} i_{j+2} ... i_n$
product offspring as:
OFF1: $i_1 i_2 ... i_j i_{j+1} i_{j+2} ... i_n$ | OFF2: $i_1 i_2 ... i_j i_{j+1} i_{j+2} ... i_n$

- **Mutation**: After a crossover is performed, mutation takes place. Mutation is intended to prevent falling of all solutions in the population into a local optimum of the solved problem. The mutation operation is employed to each bit of the offspring. The mutation probability is 0.04 for each bit.
- **Elitism**: When creating a new population by crossover and mutation, we have a big chance, that we will loose the best individual. Elitism is the name of the method that first copies the best individual (or few best individuals) to the new population. The rest of the population is constructed in ways described above. Elitism can rapidly increase the performance of GA, because it prevents a loss of the best found solution. The elite rate is 10% of population
- **Termination criteria**: The evolution process is terminated after 1000 generations.

4. EXPERIMENTS AND DISCUSSION

We first illustrate the simulation result of the proposed reduction method for simple separated data (2-dimensions, 3-classes). After finish the training of XCSR process, each rule is represented by hyper-rectangle as show in Fig. 2. The most of rules are overlap partially with other rules and are superfluous rules. We can remove those rules while preserve classification accuracy.

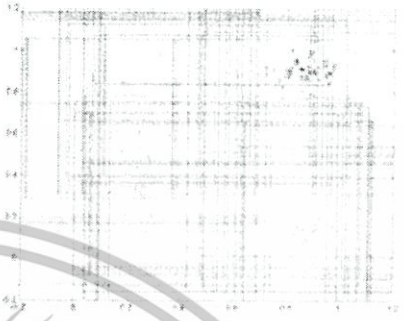


Fig. 2: Example of simple hyper-rectangle rules in two features. (Data: 3-class 2-features, XCSR: N=200; β=0.2; α=0.1; ε_0=10; ν=5; χ=0.8; μ=0.04, θ_{GA}=20; θ_{del}=25; δ=0.1;)

Fig. 3: Fewer rules remain after reduction using genetic algorithms. (N=100, 500 iter., $P_{crossover}$=0.50, $P_{mutation}$=0.04, elite size=0.10.)

After reduction using genetic algorithms, number of rules remains three rules as show in Fig. 3.Then we obtain simpler explanation of the dataset than before reduction while preserve classification accuracy.

In second experiment, we illustrated the simulation result for the UCI repository datasets: iris and wine data sets. The iris data is a three-class problem with four features and the wine data is also a three-class problem with thirteen features.

In First step, we use XCSR to classify both data sets with following parameter setting: iter. 100,000, N=1000, β 0.2, ε_0 0.001, α=0.1, ν 5, θ_{GA} 25, θ_{del} 25, μ 0.04, χ=0.5, GA-subsumption is on with θ_{sub}=10. The initial GA with following parameters setting: N=1000, 1000 iter., $P_{crossover}$= 0.50, $P_{mutation}$ 0.04, *elite size* 0.10.

Table. 1 show the average prediction accuracy of iris and wine data sets.

Table 1: The average prediction accuracy of iris and wine data set

Data Set	The average prediction accuracy
iris	98.0 ± 2%
wine	95.5 ± 3.5%

In Fig. 4. shows the average performance of remain rules in each generations. The average performance stable in early generation. However, Fig. 5. shows the average number of rules in each generations, as you can see the number of rules still continuous decreasing. That's mean, the remain generations, GA operations performs to find the set of minimum rules. In lastly generation, the number of rules continuous stable. In fact, if we increase the number of generation , the number of remain rules still stablize.

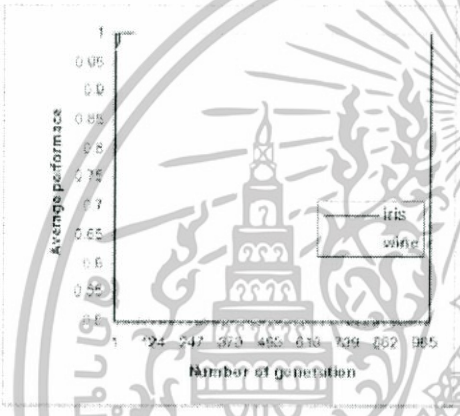


Fig. 4: The average performance of remain rules compare with before reduction

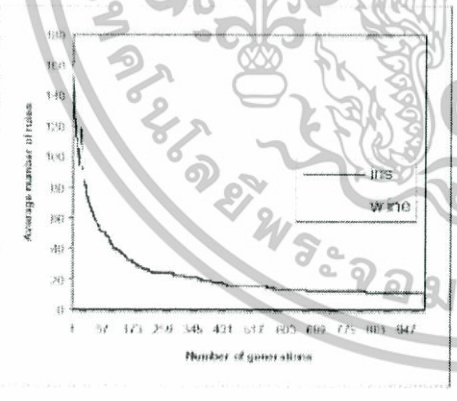


Fig. 5: The average number of rules in each generations

In Table 2, show the average number of rules compare with Dixon method. In iris dataset, it can be seen that our method could reduce number of rules slightly less than Dixon method. However, in wine dataset, the number of reduced rules are significantly less than of Dixon method.

Table 2: The average number of rules

Dataset	Dixon	GA(proposed method)
iris	14.66	12.33
wine	33.66	22.00

5. CONCLUSIONS

Our research focus on reduce the ruleset evolved by XCS classifier system to make human readable the extracted knowledge. The difficulty is finding the minimum set of rules while preserve classification accuracy. We propose genetic algorithms to solve these objectives. By computer simulations on the iris data and wine data showed that the proposed method could find the minimum set of rules less than Dixon method.

6. REFERENCES

[1] Holland J., "Escaping brittleness: the possibilities of general- purpose learning algorithms applied to parallel rule based systems", *Machine Learning an artificial Intelligence approach*. Los Altos, California: Morgan Kaufmann.

[2] Wilson, S.W., "Classifier fitness based on accuracy", *Evolutionary Computation*, Vol. 3(2), 1995, pp. 149-175.

[3] Butz, M. V., and Wilson, S. W., "An algorithmic description of XCS" In Lanzi, P. L., Stolzmann, W., and S. W. Wilson (Eds.), *Advances in Learning Classifier Systems: Third International Workshop (IWLCS-2000)*, 2000.

[4] Mansilla E.B., Tin Kam Ho, "Domain of competence of XCS classifier system in complexity measurement space", *IEEE Transactions on Evolutionary Computation*, Vol. 9, Issue 1, Feb 2005, pp. 82 - 104.

[5] Wilson, S.W., "Get Real! XCS with Continuous-Valued Inputs" In L. Booker, S. Forrest, M.Mitchell, and R Riolo, editors, *Festschrift in Honor of John H. Holland*, 1999, pp. 111-121.

[6] Dixon P.W., Corne D.W. and Oates M. J., "A Ruleset Reduction Algorithm for the XCS Learning Classifier System", *In Fifth International Workshop on Learning Classifier Systems (IWLCS-2002)*,2002.

[7] Wilson, S. W., "Mining oblique data with XCS" In Lanzi, P. L., Stolzmann, W., and S. W. Wilson (Eds.), *Advances in Learning Classifier Systems: Third International Workshop (IWLCS-2000)*, 2000.

[8] Ishibuchi H., Nozaki K., Yamamoto N.,"Selecting Fuzzy If-Then Rules for Classification Problems using Genetic Algorithm ", *IEEE Transactions on Fuzzy System*, Vol. 3, No. 3, 1995, pp. 260-270.

IEEE TRANSACTIONS ON
NEURAL NETWORKS

A PUBLICATION OF THE IEEE COMPUTATIONAL INTELLIGENCE SOCIETY

www.ieee-cis.org/pubs/tnn

IEEE Computational Intelligence Society

MARCH 2009 VOLUME 20 NUMBER 3 ITNNEP (ISSN 1045-9227)

(Contents Continued on Back Cover)

A Self-Organized, Distributed, and Adaptive Rule-Based Induction System

Ponthep Rojanavasu, Hai Huong (Helen) Dam, Hussein A. Abbass, *Senior Member, IEEE*, Chris Lokan, and Ouen Pinngern

Abstract—**Learning classifier systems (LCSs) are rule-based inductive learning systems that have been widely used in the field of supervised and reinforcement learning over the last few years. This paper employs sUpervised Classifier System (UCS), a supervised learning classifier system, that was introduced in 2003 for classification tasks in data mining. We present an adaptive framework of UCS on top of a self-organized map (SOM) neural network. The overall classification problem is decomposed adaptively and in real time by the SOM into subproblems, each of which is handled by a separate UCS. The framework is also tested with replacing UCS by a feedforward artificial neural network (ANN). Experiments on several synthetic and real data sets, including a very large real data set, show that the accuracy of classifications in the proposed distributed environment is as good or better than in the nondistributed environment, and execution is faster. In general, each UCS attached to a cell in the SOM has a much smaller population size than a single UCS working on the overall problem; since each data instance is exposed to a smaller population size than in the single population approach, the throughput of the overall system increases. The experiments show that the proposed framework can decompose a problem adaptively into subproblems, maintaining or improving accuracy and increasing speed.**

Index Terms—**Adaptive system, learning classifier systems, problem decomposition, rule-based system, self-organized map, sUpervised Classifier System (UCS).**

I. Introduction

TRADITIONALLY, a learning classifier system (LCS) [22] is a rule-based machine learning system that combines reinforcement learning and genetic algorithms. The reinforcement learning component is responsible for adjusting the strength of rules in the system according to some reward obtained from the environment. The genetic algorithm acts as an innovation discovery component which is responsible for discovering new—hopefully better and/or more general—rules. The main advantages of this approach are as follows: 1) the rule-based representation, which makes it easy to understand the learned knowledge; 2) the online learning ability, where the system is updated after the presentation of each data instance, which opens opportunities for this approach to be used in stream data mining; and 3) its robustness due to the use of genetic algorithms, which has been shown to explore the rule-space efficiently.

Traditionally, the fitness of a classifier in LCS is a function of the actual prediction. In 1995, eXtended Classifier System (XCS) [39], [40] emerged as a type of LCS where the classifier's fitness depends on the prediction of its expected payoff, which worked better on both reinforcement and supervised learning problems. An investigation [38] on a benchmark synthetic test bed, the Monk's problem, showed that XCS's performance is at least as good as traditional machine learning techniques. Subsequently, three studies [5], [9], [10], [16] undertook comparisons using real-world problems and showed that XCS is competitive for data-mining problems in comparison to other nonevolutionary learning algorithms in terms of their predictive accuracy. The first study compared XCSs with 0-R, IB1, IBk, NBa, C4.5, PART, and SMO on 15 data sets. The second study compared XCS with C4.5, the Naive Bayes classifier, PART, the instance-based learning algorithm with one and three nearest neighbor settings, and the support vector machine on 42 data sets. Statistical tests of significance showed that XCS outperforms other algorithms in some data sets. The last study applied XCS to 12 data sets and also found that XCS has the potential to be a powerful data mining tool.

In 2003, Bernadó-Mansilla *et al.* [4] proposed a specialized version of XCS for supervised learning, which they called UCS (supervised classifier system). UCS uses the actual labels in the data to directly judge on the accuracy of a classifier. The study found that UCS is able to learn faster than XCS with smaller population size, especially on problems with multiple classes.

With the increase in the volume of data collected in organizations, data mining techniques need to be able to handle large volume of data. When LCSs are applied to these problems, their rule-based representation normally requires a large number of classifiers. For problems with nonlinear boundaries, the population size is even larger because of the axis-parallel hyperplane rule representation. A large population size has a dramatic impact on the speed for processing incoming data traffic. Each time a data instance arrives, the population needs to be scanned to

Manuscript received May 15, 2007; revised January 29, 2008 and June 20, 2008; accepted July 08, 2008. First published February 10, 2009; current version published February 27, 2009. This work was supported by King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, the Australian Research Council Linkage under Grant LP0453657, and the Australian Research Council (ARC) Centre for Complex Systems under Grant CEO0348249.

P. Rojanavasu and O. Pinngern are with the Department of Computer Engineering, Faculty of Engineering, Research Center for Communication and Information Technology (ReCCIT), King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand (e-mail: kpouen@kmitl.ac.th).

H. H. Dam, H. A. Abbass, and C. Lokan are with the Artificial Life and Adaptive Robotics Laboratory (ALAR), School of Information Technology and Electrical Engineering, University of New South Wales, Australian Defence Force Academy, Canberra, N.S.W. 2600, Australia (e-mail: h.dam@adfa.edu.au; h.abbass@adfa.edu.au; c.lokan@adfa.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2008.2008334

find those rules that match the incoming data instance. Consequently, using LCSs for real-time stream data mining is hindered by the required large population size.

Bagnall and Cawley [3] and [15] investigated the performance of XCS on large data sets, and found that they pose a challenge for XCS. This problem has been studied by a number of researchers, trying to introduce mechanisms to cope with large data sets in LCSs. Some proposed to compact the population by changing the representation [14], [29], some proposed distributed and ensemble frameworks [8], [12], while others offered some efficient encoding schemes [31].

Automatic problem decomposition of the learning problem is one direction that holds much promise, but has not been investigated in LCSs. The idea is to automatically decompose the problem so that each subproblem is handled by a different learning machine. This is hoped to reduce the complexity of the learning task; thus it will require a less complex learning machine, which can then boost the speed of processing incoming traffic. A study by Maimon and Rokach [33] showed several benefits of decomposing data mining problems decomposition: 1) improves the classification accuracy; 2) provides conceptual simplification of the problem; 3) enhances the feasibility of being able to handle larger databases; and 4) provides clearer and more comprehensible results. There are also benefits in terms of reducing the runtime by solving smaller problems, and the opportunity to use different learning machines for individual subproblems.

The objective of this paper is to provide a framework to adaptively decompose a problem using a pregate that is tightly coupled with a classifier. In this paper, we hypothesize that having a self-organized map (SOM) as a pregate for decomposing the search space would reduce the complexity of the problem. Moreover, having a separate classifier to learn each of these subproblems could improve the accuracy in a few cases because each classifier only handles a possibly simpler problem. We investigate two different machine learning classification methods: UCS as a representative of the evolutionary school and feedforward artificial neural network (ANN) as a traditional nonevolutionary classifier. Several visualization techniques are employed to shed light on the resultant decomposition.

The rest of this paper is organized as follows. Section II provides a brief literature review, followed by a description of the proposed framework in Section III. Section IV describes the design of the experiments, while the following three sections focus on the analysis of the results. Conclusions are drawn and future work is discussed in Section VIII.

II. Background Materials

A. Problem Decomposition

A decomposition approach in classification can be categorized into one of two types [32].

- Intermediate concept decomposition, where the problem is projected onto an intermediate space upon which the decomposition takes place to partition that space into smaller subspaces. For example, one may use Gaussian kernels to do the mapping to the intermediate space. Another example is decomposing the weight space of a neural network to get the networks specializing on doing different activities.
- Original concept decomposition, where problem decomposition is undertaken directly on the input space without a mediator. The original problem is divided into several subproblems by partitioning the training set. The mixture of experts model is an example of this category, where a gate is used to weight the classifiers differently based on different input instances. Thus, decomposition occurs on the input space.

This paper will focus on the second approach. Original concept decomposition can be divided further into the sample-based and space-based approaches. Bagging is one of the most well-known methods using the sample-based approach [7]. In bagging, sampling with replacement from the training data is used to create different data sets, each with its own bias. A classifier is built using each of these data sets, then a simple voting gate is used to combine the output.

Alternatively, boosting [19] is used to allow data instances that are incorrectly predicted by previous classifiers to be weighted higher than other data instances, thus allowing them to be chosen more often than examples that were correctly predicted to build a new classifier. Therefore, boosting attempts to produce new classifiers that are better able to predict data for which the current ensemble's performance is poor.

In ensemble learning, a problem is decomposed (either explicitly or implicitly) into several subproblems. Each subproblem is then handled by a classifier. The final decision is made by an appropriate classifier. The mixture of experts (ME) model [23] is an example of the space-based decomposition. In this approach, the mixture of experts decomposes the input space, such that each expert examines a different part of the space. Each output of an expert is seen as a conditional probability for the given input vector. A gating network is responsible for combining the opinions of the various experts by assigning a weight to each network. Jordan and Jacobs [24] introduced the hierarchical mixtures of experts (HME) as an extension of the ME model. HME decomposes the space into subspaces, and then recursively decomposes each subspace to further subspaces.

Automatic problem decomposition demands an explicit mechanism to ensure that the ensemble members are specialized on different (but potentially overlapping) subspaces. An ensemble of learning machines would normally require a gate. According to Abbass [1], there are two types of gate: a pregate and a postgate. An explanation of each is given below and also see Fig. 1.

1) A pregate acts as a router that assigns the incoming traffic to different directions based on some criteria. Research on pregates is mainly focusing on getting members of the ensemble to specialize on different parts of the input space through a partitioning or clustering function. The decision on which learning machine will get which instances can be done independently from the individual learning machines. However, this approach suffers from its inflexibility in adapting the decomposition when concept drift occurs. The separation between the decomposition from the learning problem also has its own limitation in the sense

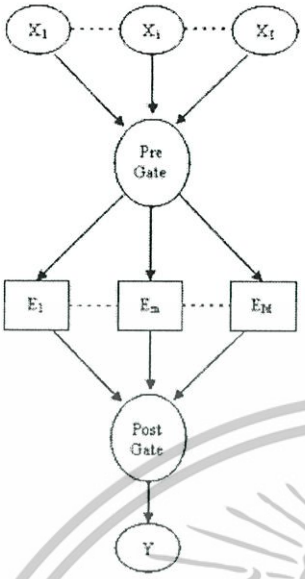


Fig. 1. Generic description to ensemble research.

that the decomposition assumes that all learning machines are homogenous and are performing equally well. Another branch of research on pregates is *competitive learning*, where different learning systems compete on the input until it is taken up by one of them.

2) A postgate acts as an aggregate function that combines the (potentially different) opinions made by the individual members of the ensemble. The motivation for this learning approach comes from the area of expert systems, where a committee of experts is used to make decisions. If only one member responds to each situation, there will be no conflict within the committee. The problem becomes similar to the use of pregates with one-to-one or many-to-one mapping between clusters in the input space and ensemble members. However, if a group of committee members responds to the same input, conflicts between their opinions may happen. The issue is then how to combine these opinions into a single coherent opinion. Postgates used in the literature can be very simple such as voting and winner-take-all, or more complex such as mixture of experts [23] and coevolutionary mixture of experts [34], [35].

Problem decomposition does not need to rely on a pregate alone or a postgate alone. It can involve an ensemble of ensembles with a pregate to decompose a problem and a postgate to combine opinions.

Recently, Rokach *et al.* [37] proposed the K-classifier algorithm, which employs the k-means clustering algorithm to decompose a data set into mutually exclusive subsets. Each subset is then handled by a classifier. This type of decomposition can be seen as a pregate, where an incoming data instance is routed to an appropriate classifier based on which cluster it belongs to. Their experiments, using decision trees, neural networks, and naive Bayes, showed that the proposed method is well suited for data sets of numeric input attributes.

One main disadvantage of this approach is the separation between the decomposition problem and the classification problem. By using the K-means clustering algorithm, the data set needs to be processed completely offline before being fed to the classifier.

In this paper, we overcome this problem by coupling the decomposition with the classifier. The idea is that by using a clustering algorithm that is self-organizing itself on the data set, the classifiers associated with the different (changing) clusters adaptively learn the classification problem.

The tight coupling between the decomposition and the classifier has a number of advantages. 1) If the decomposition is separated from the classification in a two stage process, all data need to exist beforehand to decompose the problem, then classification is applied on each subproblem. This makes it difficult to use this system for stream data mining, where the data arrives in real time in a huge quantity. 2) If there is a concept drift (i.e., the decision boundary changes), a system that separates decomposition from classification would require redecomposing the problem. But this system would also need a mechanism to detect that the environment has changed, to decompose the problem, and then allow the classifier to recover from the changes in the environment. Tightly coupling the decomposition with the classification would make the system adaptive, where the decomposition is continuously occurring as the classifier learns. 3) the separation between decomposing a problem and learning a proper classification sounds artificial; after all, it is a single problem that we are tackling, of how to learn a proper classification model.

This tight coupling can be seen in the eyes of competitive learning, which is unsupervised learning that uses the concept of competition to group input patterns into classes based on their features. A common goal of competitive learning algorithms is to distribute a certain number of vectors in a possibly high-dimensional space. The distribution of these vectors should reflect (in one of several possible ways) the probability distribution of the input signals, which, in general, is not given explicitly but only through sample vectors [20].

For example, in Fig. 2, the symbol "■" represents the weight vectors in the output layer. The initial state of weight vectors is random. After learning, the weights move towards the centroid of each cluster of the input space.

There are many models in the area of competitive learning, which are usually categorized in two main categories: hard and soft competitive learning. In hard competitive learning, there is only one winning unit, that is, a *winner-take-all* strategy. However, in soft competitive learning, there can be more than one unit adapting its weight vector at a time. Each unit will adapt its weight vector relative to its proximity to the input pattern, that is, a *winner-take-most* strategy.

In this paper, we use Kohonen's SOM for soft competitive learning, and tightly couple it with an LCS. We experiment with UCS and ANN, as representatives of evolutionary and nonevolutionary classifiers, respectively. The next three sections will introduce SOM, UCS, and ANN.

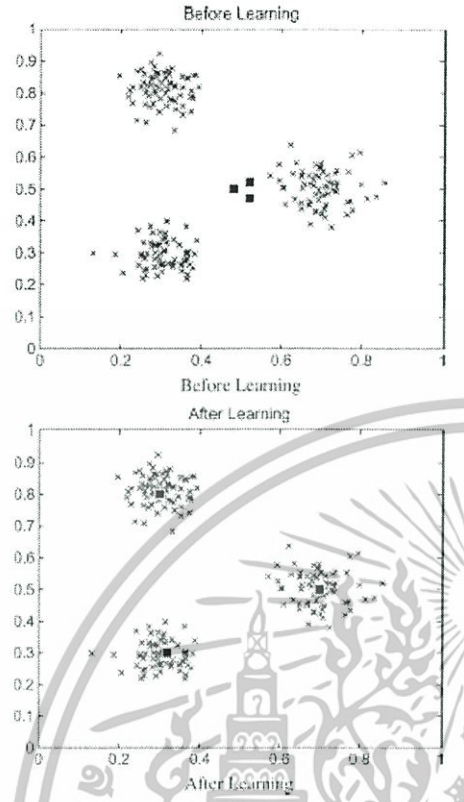


Fig. 2. Example to show how the weight vector is changed in competitive learning. (■) represents the weight vectors in the output layer. (×) represents a data point.

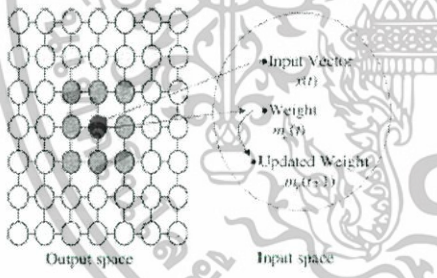


Fig. 3. Structure of SOM and learning process.

B. Self-Organizing Map

One of the most well-known algorithms for soft competitive learning is SOM artificial neural network [25], [26]. Fig. 3 shows a 7×7 SOM.

The model consists of two layers. First, the output layer consists of neural elements called nodes. Each node i is assigned an n-dimensional weight vector m_i. That is, $m_i \in \mathbb{R}^n$, where $\mathbb{R}^n$ is an n-dimensional space. It is necessary to note that the weight vectors have the same dimensionality as the input vector. Second, the input layer consists of input vectors for training the map. The learning process of a SOM can be seen in terms of continuous adaptation of the nodes for input vectors. In each learning iteration t, input vector $x(t)$ is fed into every node in the map to identify the output vector's winning node. It is common to use the Euclidean distance as the basis to measure similarities. The winning node c can thus be defined as the one with the smallest distance (maximum similarity) to the input vector

$$c : m_c(t) = \min_i \|x(t) - m_i(t)\|. \tag{1}$$

The weight of the winning node c is tuned by the difference between the input vector and the weight vector. Not only the winning node is learning but also its neighborhood nodes are learning as well. Therefore, their weight vectors become more similar to the input pattern. As a result, the respective node is more likely to win at future presentations of this input pattern. The weight vectors of the winning node and its neighbors at time $t+1$ are updated as follows:

$$m_i(t+1) = m_i(t) + \alpha(t) \times h_{ci}(t) \times [x(t) - m_i(t)] \tag{2}$$

where t is the current time, $x(t)$ is the current input vector, and $m_i(t)$ is the current weight vector. $\alpha(t)$ is the learning rate, which is reduced gradually after each learning iteration as follows:

$$\alpha(t) = \alpha(0) \times \frac{T-t}{T} \tag{3}$$

where T is the total number of iterations. $h_{ci}(t)$ is the neighborhood function. For convergence, it is necessary that $h_{ci}(t) \to 0$ as $t \to \infty$. This means that as learning progresses, the neighborhood within which the nodes are activated will shrink and the rate for modifying the reference vectors will decrease. Usually, a Gaussian function is used as the neighborhood function, as follows:

$$h_{ci}(t) = \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right) \tag{4}$$

where the vectors r_c and r_i represent the coordinates of the winning node c and neighbor node i. $\|r_c - r_i\|$ is the distance between nodes c and i, and $\sigma(t)$ is the radius of the neighborhood node, updated as follows:

$$\sigma(t+1) = 1 + (\sigma(t) - 1) \times \frac{T-t}{T}. \tag{5}$$

C. UCS

UCS is a rule-based evolutionary learning classifier system, in which each classifier represents a partial solution to the target classification problem. The goal of UCS is to evolve a population of classifiers that represents a complete solution to the problem.

Each classifier in UCS consists of a *condition* (the body of the rule), an *action* (the prediction of the classifier), and some parameters. The *condition* refers to several environmental states, to which the classifier may match. The *action* is the outcome if the classifier fires. The key parameter associated with a classifier is the fitness F, which measures how good the classifier is relative to the rest of the population. Another two important parameters are the *numerosity* and *experience*. A classifier of UCS is a macro-classifier, which holds a distinct rule (a unique pair of *condition : action*) within the population $[P]$. The numerosity parameter records the number of copies of the classifier in the population. Whenever a new classifier is introduced, the population is scanned through to check if a copy of the classifier already exists. If it exists, the numerosity value is incremented by one, otherwise the classifier is added to the population and its numerosity value is initialized to 1. The *experience* parameter indicates how often the classifier is chosen for making a prediction, which represents how general a classifier is.

During the learning cycle, the system receives a series of training data instances. Since UCS is a supervised learner, a training data instance contains also the target class. First, UCS finds a match set $[M]$ with all classifiers in the population whose condition matches the incoming instance. A *correct set* $[C]$ is then formed, containing those classifiers in $[M]$ that have the same action as the input. If $[C]$ is empty, *covering* is activated, wherein a classifier that matches the input is created and assigned the same class as the input. UCS is an incremental learner, where knowledge is updated as more data becomes available. All parameters of the classifiers in $[M]$ are revised for each training instance according to whether they belong to $[C]$, reflecting the system's updated belief after being exposed to new observations.

The fitness of a classifier is based on accuracy, which is computed as the number of times a classifier correctly predicted the class divided by the number of times the classifier matched an incoming data instance

$$acc = \frac{N_{correct}}{N_{matches}}. \tag{6}$$

The fitness is computed as a function of accuracy

$$F = (acc)^{\nu} \tag{7}$$

where ν is a predefined constant.

Genetic algorithm (GA) is invoked in $[C]$ if the average time since the last application of GA to classifiers in $[C]$ exceeds a user-defined threshold. If GA is activated, two parents are selected from $[C]$ with a probability proportional to their fitness. Two offspring are generated by reproducing, crossing over, and mutating the parents with certain probabilities. Offspring are inserted in $[P]$ if they are not subsumed by the parents. If the population size reaches a predefined limit, some classifiers are removed by voting within the population.

In the exploitation phase, an input instance is given with unknown class. After forming the match set $[M]$, UCS proposes a class based on voting among the classifiers in $[M]$; the votes are weighted by the fitness of each classifier in the match set.

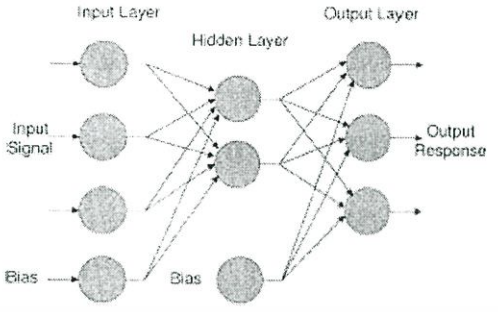


Fig. 4. Multilayer perceptron (MLP) with three input nodes, three output nodes, and a single hidden layer with two hidden nodes.

D. Feedforward Artificial Neural Networks

Feedforward ANNs are popular in machine learning because of their versatility for solving a wide range of forecasting, classification, and pattern recognition problems. Haykin [21] presented several powerful properties and characteristics of neural networks including their ability to learn nonlinear decision boundaries, ability to adapt, being robust, and being a compact representation of the target model.

An ANN is a directed graph with several layers, including an input layer, hidden layers, and an output layer. A bias unit is connected to each hidden unit in the hidden layers. The number of neurons in the input layer is the same as the number of input features. The outputs are encoded using m output neurons corresponding to m classes. The output node with the highest activation designates the class. A neuron employs a sigmoid function. The architecture of an ANN is shown in Fig. 4.

Training an ANN occurs by tuning the network weights to minimize the output error of the network. The backpropagation algorithm is popular in training ANN and showed success in many applications such as handwriting character recognition [28].

III. A Framework for Competitive Classification Environments

Fig. 5 depicts the proposed framework. As a pregate, we use a SOM neural network to decompose the problem. A classification algorithm is employed to handle each subproblem independently. We experiment with two different types of classification system: UCS and ANN. We call the combined system self-organized UCS (SOUCS) or self-organized ANN (SOANN). Decomposition and learning occur hand-in-hand; thus, the system decomposes the problem as it learns the class boundaries.

The SOM, the pregate in the proposed framework, can be seen in two different ways. First, it decomposes the problem by clustering the data (note that the class label is not presented to the SOM). Second, the SOM can be seen as a router in a distributed environment, which assigns the data as they arrive to a node that will take care of classifying the incoming instance. Given any data instance, there is a single UCS responsible for it.

Before starting the framework, the number of nodes in SOM needs to be determined. For example, one can decide to use a

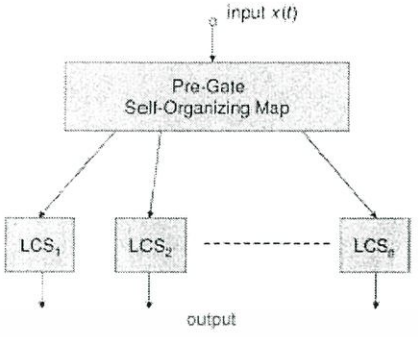


Fig. 5. Proposed framework. SOM is used to route the data instances as they arrive to one of the LCS.

2×2 or a 3×3 SOM. Each node in the SOM is associated with a complete classifier. Thus, there are four independent classifiers for the 2×2 SOM, and nine independent classifiers for the 3×3 SOM.

The system is trained incrementally in the training phase. For each training instance, the winning node is first identified based on the similarity function. The weights of the winning node are updated, based on proximity to the instance. The instance is then fed into the LCS coupled with the winning node. A complete training process is carried out at that node. In the case of UCS, covering might occur if the match set is empty, parameters of classifiers in the match set are updated, and GA might be activated. In the case of ANN, the network weights are updated. With each training instance, both the SOM and its classifiers are updated on the fly.

In the testing phase and given a testing instance, SOM first chooses the winning node. The instance is then routed to the LCS coupled with the winning node to make a prediction. The testing phase does not involve any learning in either the SOM or the LCSs.

It is well known [11], however, that in distributed environments, we cannot simply scale the population size linearly with the number of processors. In other words, a population size of 1000 on a single processor does not map to population sizes of 250 in a four-processor environment. In the worst case, the population size of each individual node may equal the population size of the single UCS. One may wonder then why we need SOUCS. Apart from the nice features of problem decomposition, discussed in Section II in terms of increased accuracy and better representation of the problem, we present a simple queuing analysis below. The analysis is described for SOUCS, but the idea applies just as well to SOANN.

Assume that the traditional UCS is able to process T instances per second. Let R instances per second denote the rate of data arrival. Assume a single pass learner (for stream data mining) as presented in [13]. We now need to compare between the single population approach and the SOUCS environment. In this analysis, we will assume that the interarrival time is uniform for simplicity. In the single population approach, if $R > T$, queuing theory tells us that we need an infinite queue to accumulate the data waiting for processing. In a multiserver environment, the condition for handling this data set without the need for any queues is that the router can keep up with the arrival rate, and each node can keep up with its share of the arriving instances. This condition is $R < \min(U, M \times T)$, where M is the number of UCS nodes in SOUCS and U is the number of instances a router can route per second (in SOUCS, the router is a SOM) to distribute the incoming data to one of the M UCS populations.

We assume here that all UCS nodes have the same population size, and therefore, the processing time of an instance across the nodes is homogeneous. We also assume that the instances are routed evenly by the SOM to the different nodes. In this case, the speedup in SOUCS is essentially proportional to M. If these assumptions do not hold, the mathematics is more complex and the speedup is smaller. The point still holds that the parallelism in SOUCS increases the throughput of the system.

IV. Experimental Design

A. Testing Problems

To understand the behavior of the proposed framework, we conduct three sets of experiments: one using synthetic problems, one using a number of small and medium size data sets, and a third using a large data set.

1) Synthetic Data Sets: We designed five artificial problems, with different complexity in terms of class boundaries and class distribution. All problems have two attributes with boxing constraints between 0 and 4. In each problem, 8000 data points are sampled using a uniform random number generator.

The problems are called "circle 1," "circle 2," "circle 3," "circle 4," and "cross circle" as shown in Fig. 6. The idea is to start with a simple classification problem with a nonlinear class boundary (circle 1) and nonuniform class distribution—noting that the ratio of the positive cases to the negative cases is proportional to the area of the circle(s) to the area of the square. The complexity of the learning problem is then increased incrementally by adding more disjoint areas for the positive class (circles 2–4). In the fifth problem, the class boundary is more complex.

2) Small and Medium Data Sets: This set of problems was selected to give us a range of important characteristics, such as i) number of features: low (up to ten features) or high (more than ten features); ii) different representations: real, integer, nominal, or a mixture; iii) number of classes: binary (two possible outcomes), small (three or four outcomes), or large (five or more outcomes).

Table I shows the properties of the chosen data sets used in this paper: Inst (the number of instances in the data set), Fs (the number of features), R (the number of real-valued features), I (the number of integer-valued features), N (the number of nominal-valued features), and C (the number of classes).

Except for the Tao problem, all these problems are available from the UCI data repository [2]. The Tao problem was first proposed in [30] and is used widely in LCS research.

3) Large Data Set: The Forest-Cover-type data set of the Roosevelt National Forest in northern Colorado, available at [2], is chosen for testing. According to [6], the collected data covers an area of 70 miles northwest of Denver in Colorado [30 m ×

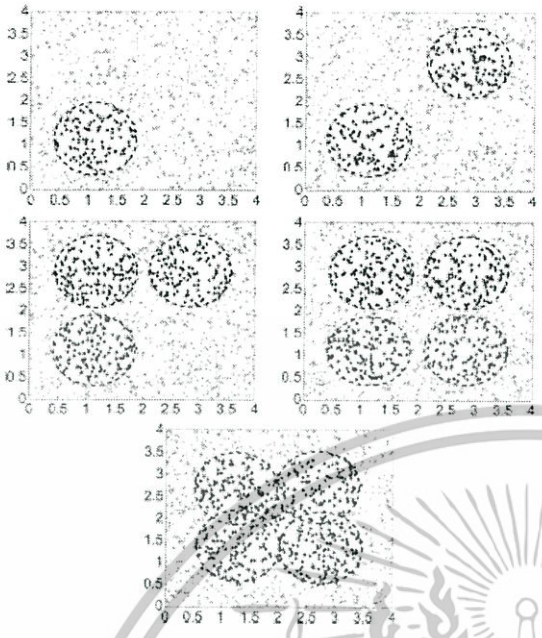


Fig. 6. Five synthetic problems: circle 1 (first row, left-hand side), circle 2 (first row, right-hand side), circle 3 (second row, left-hand side), circle 4 (second row, right-hand side), cross circle (last row). The "+" represents positive data and "." represents negative data. Each circle represents the boundary for a positive label area except in *cross circle*, where the positive label area is the intersection of four circles.

TABLE I
PROPERTIES OF TESTING DATA SETS

Problem	# Inst.	# Fs.	# R	# I	# N	# C
breast-w	699	9	0	9	0	2
bupa	345	6	6	0	0	2
diabetes	768	8	8	0	0	2
glass	214	9	0	0	0	6
iris	150	4	4	0	0	3
segment	2310	19	19	0	0	7
tao	1888	2	2	0	0	2
vehicle	946	17	17	0	0	4
wine	178	13	13	0	0	3

30 m cells obtained from United States Forest Service (USFS) Region 2 Resource Information System (RIS)], which has seven major Forest Cover types (or seven classes). The data was obtained from the United States Geological Survey.

The data set has 581 012 observations, 54 attributes, and no missing values. Each observation is labeled as one of seven different classes (Forest Cover types).

B. Experimental Setup

To evaluate the proposed framework, we experiment with two structures: 2×2 SOM and 3×3 SOM.

The parameter settings for UCS are fixed for all problems and all environments, similar to those used in [41] and [4] as follows: $\upsilon = 5$, $\theta_{\mathrm{GA}} = 50$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{\mathrm{del}} = 50$, and $\theta_{\mathrm{sub}} = 50$.

Four systems are tested in this paper to verify the proposed framework: traditional UCS, SOUCS, traditional ANN, and SOANN.

For UCS and SOUCS, we first make the sum of the total population sizes of SOUCS equal (or almost equal) to the population size of a single UCS. For example, if we set the population size as $N = n$ in a single UCS environment, the population size for each UCS in the SOUCS environment is set to $N = n/M$, where M is the number of UCS environments. For a second comparison, we double the population size for each UCS in the SOUCS environment, while leaving the population size of the single UCS unchanged. There is now some "overhead" in SOUCS, since the total population in all UCSs in the SOUCS environment is more than for the single UCS. Each individual UCS in the framework still has a smaller population than the single UCS, however, so the overall system should still be faster. This gives us an indication of the effect on accuracy if we are prepared to accept the overhead. Finally, we double the population size of the single UCS as well, to give a direct comparison again with the SOUCS environment.

In the first set of experiments, we analyze UCS and SOUCS on the synthetic data sets. The population size of UCS is set to $N = 1000$. For SOUCS-1, the SOM is of size 2×2 and the population size for each UCS is set to 250. For SOUCS-2, the SOM is of size 3×3 and the individual population size of each UCS is set to 111. Next, we double the population size of each individual UCS to $N = 500$ and $N = 222$ for SOUCS-1, SOUCS-2, respectively, while leaving the population size of the traditional UCS unchanged. Finally, we increase the population size for the traditional UCS to $N = 2000$.

The second set of experiments is carried out on the UCI data sets. We use a population size of 6400 for the single UCS. For SOUCS-1 and SOUCS-2, the SOM is of size 2×2 and 3×3, respectively, and the population size for each UCS is set to 1600 and 711, respectively. Next, we double the population size for each UCS to $N = 3200$ and $N = 1422$ for SOUCS-1 and SOUCS-2, respectively, while leaving the population size for the single UCS at 6400. Finally, we increase the population size for the traditional UCS to $N = 12\,800$.

The third set of experiments is tested on the Forest data set. We use a population size of 6400 for the single UCS. For SOUCS-1, SOUCS-2, and SOUCS-3, the SOM is of size 2×2, 3×3, and 4×4, respectively, and the population size for each UCS is set to 1600, 711, and 400, respectively. Next, we increase the population size for each UCS to $N = 3200$ for all SOUCS.

All results, unless stated otherwise, are the average over ten-fold cross validation. Each run uses different random seeds that are consistent in all experiments. We use the term *iteration* to refer to a single pass through the training set. The statistical data is collected after 500 iterations in each experiment, except for the Forest data set; due to the large size of this data set, we collect the statistical data after five iterations.

When UCS is replaced with an ANN, we maintain the same number of hidden units in all networks. For the University of California at Irvine (UCI) data sets, five hidden nodes are used, while 20 nodes are used for the Forest data set.

The statistical test of significance (t-test) is used with a significance level of 0.05.

TABLE II
MEAN AND STANDARD DEVIATION OF THE ACCURACY OF UCS AND SOUCS ENVIRONMENTS ON THE SYNTHETIC DATA SETS

	UCS	SOUCS-1	SOUCS-2
circle 1	0.981±0.006	0.981±0.004	0.982±0.004
circle 2	0.965±0.023	0.966±0.009	0.962±0.023
circle 3	0.942±0.008	0.956±0.012 •	0.946±0.015
circle 4	0.940±0.010	0.951±0.007 •	0.931±0.007 ○
cross circle	0.946±0.007	0.945±0.019	0.941±0.010

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 1000$ for the single UCS, $N = 250$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 111$ for each UCS in SOUCS-2 with 3×3 SOM.

TABLE III
MEAN AND DEVIATION OF THE ACCURACY OF UCS AND SOUCS ON A SET OF ARTIFICIAL DATA SETS

	UCS-1	SOUCS-1	SOUCS-2
circle 1	0.981±0.006	0.982±0.012	0.985±0.004
circle 2	0.965±0.023	0.968±0.011	0.965±0.010
circle 3	0.942±0.008	0.957±0.005 •	0.961±0.012 •
circle 4	0.940±0.010	0.955±0.007 •	0.953±0.007 •
cross circle	0.946±0.007	0.951±0.005 •	0.954±0.008 •

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 1000$ for the single UCS, $N = 500$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 222$ for each UCS in SOUCS-2 with 3×3 SOM.

TABLE IV
MEAN AND DEVIATION OF THE ACCURACY OF UCS AND SOUCS ON A SET OF ARTIFICIAL DATA SETS

	UCS-2	SOUCS-1	SOUCS-2
circle 1	0.984±0.006	0.982±0.012	0.985±0.004
circle 2	0.964±0.008	0.968±0.011	0.965±0.010
circle 3	0.948±0.009	0.957±0.005 •	0.961±0.012 •
circle 4	0.947±0.006	0.955±0.007	0.953±0.007
cross circle	0.948±0.007	0.951±0.005	0.954±0.008 •

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 2000$ for the single UCS, $N = 500$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 222$ for each UCS in SOUCS-2 with 3×3 SOM.

V. AN INVESTIGATION OF THE PREDICTIVE ACCURACY

A. Self-Organized UCS

In this section, we compare the predictive accuracy of UCS and SOUCS using different setups.

Table II presents the mean and the standard deviation (over ten runs) of the predictive accuracy in different environments on the synthetic data sets.

Table II shows that the accuracy of SOUCS-1 is significantly better than that of UCS in two problems and equivalent in three problems. SOUCS-2 performs worse than UCS in one problem and equivalent in the other four problems.

SOUCS-1 performs better than UCS in the circle 3 and circle 4 problems, where the problem complexity increased with the increase in the discontinuity in class boundaries. In the circle 4 problem, however, SOUCS-2 performs significantly worse than a single UCS. We hypothesize that this occurred because of the small population size of each UCS in SOUCS-2.

Therefore, we undertake an extra experiment by increasing the population size of each UCS to 500 individuals for SOUCS-1 and 222 for SOUCS-2. Table III compares the accuracy of SOUCS with a single UCS whose population size is still 1000, and Table IV gives the comparison when the population size for the single UCS is increased to 2000 (called UCS-2).

From Tables III and IV, we can see that SOUCS-1 is significantly better than UCS-1 in three problems and better than UCS-2 in one problem. Similarly, the accuracy of SOUCS-2 is significantly better than UCS-1 in three problems and better than UCS-2 in two problems.

Increasing the population size in each local UCS of SOUCS-1 and SOUCS-2 results in better accuracy. It is worth mentioning that achieving an accuracy of 1 is very difficult in this case because UCS uses axis-parallel hyperplanes, which will misclassify some data around the nonlinear decision boundaries of the circles.

We then test the SOUCS environment on the small and medium size UCI data sets. Table V shows the accuracy of a single UCS and the SOUCS environment.

Once more, the SOUCS environment does well. SOUCS-2 is significantly better on three out of nine problems, comparable results are achieved on four problems, and SOUCS-2 performs worse in two problems.

SOUCS performs worse than UCS in the segment and vehicle problems, because both problems have a high number of classes (seven classes for segment and four classes for vehicle) and a

TABLE V
MEAN AND DEVIATION OF THE ACCURACY OF UCS AND SOUCS ON UCI DATA SETS

	UCS	SOUCS-1	SOUCS-2
Breast-w	0.961±0.015	0.966±0.012	0.973±0.016 •
Bupa	0.681±0.053	0.715±0.034	0.719±0.035 •
Diabetes	0.732±0.056	0.736±0.031	0.713±0.043
Glass	0.710±0.054	0.677±0.058	0.682±0.112
Iris	0.953±0.045	0.960±0.034	0.947±0.042
Segment	0.962±0.011	0.965±0.011	0.949±0.050 ○
Tao	0.880±0.012	0.940±0.021 •	0.956±0.016 •
Vehicle	0.715±0.031	0.690±0.040 ○	0.669±0.041 ○
Wine	0.966±0.047	0.967±0.060	0.949±0.050

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 6400$ for the single UCS, $N = 1600$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 711$ for each UCS in SOUCS-2 with 3×3 SOM.

TABLE VI
MEAN AND DEVIATION OF THE ACCURACY OF UCS AND SOUCS ON UCI DATA SETS

	UCS-1	SOUCS-1	SOUCS-2
Segment	0.962±0.011	0.971±0.009 •	0.970±0.011 •
Vehicle	0.715±0.031	0.714±0.034	0.697±0.030

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 6400$ for the single UCS, $N = 3200$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 1422$ for each UCS in SOUCS-2 with 3×3 SOM.

TABLE VII
MEAN AND DEVIATION OF THE ACCURACY OF UCS AND SOUCS ON UCI DATA SETS

	UCS-2	SOUCS-1	SOUCS-2
Segment	0.968±0.014	0.971±0.009	0.970±0.011
Vehicle	0.719±0.021	0.714±0.034	0.697±0.030

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 12800$ for the single UCS, $N = 3200$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 1422$ for each UCS in SOUCS-2 with 3×3 SOM.

TABLE VIII
MEAN AND DEVIATION OF THE ACCURACY OF UCS AND SOUCS ON FOREST-COVER-TYPE DATA SET

UCS	SOUCS-1	SOUCS-2	SOUCS-3
0.729±0.005	0.719±0.006○	0.711±0.004○	0.694±0.006○

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 6400$ for the single UCS, $N = 1600$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 711$ for each UCS in SOUCS-2 with 3×3 SOM, $N = 400$ for each UCS in SOUCS-3 with 4×4 SOM.

TABLE IX
MEAN AND DEVIATION OF THE ACCURACY OF UCS AND SOUCS ON FOREST-COVER-TYPE DATA SET

UCS	SOUCS-1	SOUCS-2	SOUCS-3
0.729±0.005	0.728±0.003	0.725±0.003	0.724±0.005

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOUCS environment is better, and by ○ if the SOUCS environment is worse. $N = 6400$ for the single UCS, $N = 3200$ for each UCS in SOUCS-1 with 2×2 SOM, $N = 3200$ for each UCS in SOUCS-2 with 3×3 SOM, $N = 3200$ for each UCS in SOUCS-3 with 4×4 SOM.

high number of features (19 features for segment and 17 features for vehicle). After decomposing the problem by the SOM, each subproblem is still complex and the small population size of the local classifiers is insufficient to learn the concept.

Table VI compares the accuracy of UCS and SOUCS when the population size in each local UCS in SOUCS is doubled to $N = 3200$ in SOUCS-1 and $N = 1411$ in SOUCS-2. Table VII compares the accuracy when the population size of the single UCS is also doubled to 12800 (called UCS-2).

In the "Segment" problem, both SOUCS-1 and SOUCS-2 achieve better accuracy than UCS-1 and comparable accuracy with UCS-2. In the "Vehicle" problem, SOUCS-1 and SOUCS-2 are able to achieve similar accuracy as both UCS-1 and UCS-2.

In conclusion, SOUCS is promising in terms of its ability to generalize better if the population size and the number of cells in the SOM are chosen carefully.

We then challenge SOUCS with a large problem data set: that is, the Forest-Cover-type data set. Table VIII shows the accuracy with a single UCS and the SOUCS environment. We can see that every SOUCS environment performs worse than the single UCS, especially with the 4×4 SOM.

The result is not surprising to us because as we saw before, the population size needs to be selected carefully. Therefore, we increase the population size of each UCS in all SOUCS to $N = 3200$. The accuracy of each SOUCS environment has no significant difference with a single UCS, as shown in Table IX, while they use less time for learning data. In fact, each local UCS is using half the population size of the single UCS, which implies that the local UCS will process data faster and the distributed environment will allow more data to be processed.

TABLE X
MEAN AND DEVIATION OF THE ACCURACY OF ANN AND SOANN ENVIRONMENTS ON UCI DATASETS

	ANN	SOANN-1	SOANN-2
breast-w	0.966±0.020	0.964±0.018	0.973±0.017•
bupa	0.736±0.053	0.701±0.038○	0.681±0.054○
diabetes	0.768±0.050	0.764±0.055	0.745±0.062
glass	0.658±0.083	0.650±0.095	0.623±0.084
iris	0.967±0.047	0.960±0.047	0.953±0.045
segment	0.948±0.015	0.958±0.010•	0.955±0.015
tao	0.918±0.008	0.915±0.010	0.940±0.015•
vehicle	0.785±0.051	0.807±0.046	0.786±0.048
wine	0.983±0.027	0.978±0.029	0.977±0.029

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOANN environment is better, and by ○ if the SOANN environment is worse.

TABLE XI
MEAN AND DEVIATION OF THE ACCURACY OF ANN AND SOANN ON THE FOREST DATA SET

ANN	SOANN-1	SOANN-2
0.739±0.004	0.742±0.005•	0.735±0.006○

[a] Differences that are statistically significant at a significance level of 0.05 are denoted by • if the SOANN environment is better, and by ○ if the SOANN environment is worse.

TABLE XII
COMPARISON OF THE PERFORMANCE OF SOUCS AND SOANN WITH OTHER PUBLISHED WORK ON THE FOREST DATA SET

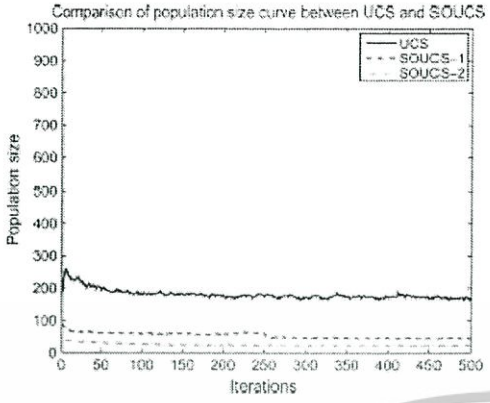
Approach	Performance
BagGGPC [17]	0.619
BoostGPC [18]	0.664
Neural Network Ensemble [36]	0.771
ANNCAD [27]	0.710
UCS (N=6400)	0.729
SOUCS-1 (N=3200)	0.728
SOUCS-2 (N=3200)	0.725
SOUCS-3 (N=3200)	0.724
ANN	0.739
SOANN-1	0.742
SOANN-2	0.735

B. Artificial Neural Network

Similar to SOUCS, SOANN-1 uses a 2×2 SOM, while SOANN-2 uses a 3×3 SOM. From Table X, we can see that the SOANN environment performs significantly better on three out of nine problems, equivalently on five problems, and worse on only one problem. This experiment confirms that using a SOM as a pregate can possibly be generalized to other classification methods easily.

Similar to SOUCS, we test SOANN on the Forest-Cover-type data set. As can be seen from Table XI, SOANN-1 performs better than ANN.

Table XII shows the performance obtained on the Forest data set in this paper, and the performance obtained with other evolutionary and nonevolutionary approaches in other papers (standard deviation is not tabulated because it was not available in all of the papers we compare against). The accuracy we obtained



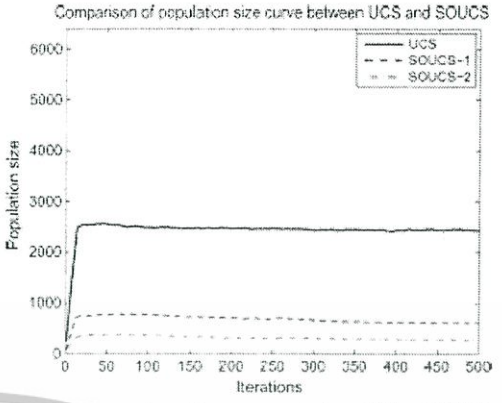


Fig. 7. Population size of UCS, SOUCS-1, and SOUCS-2 on the circle 4 problem.

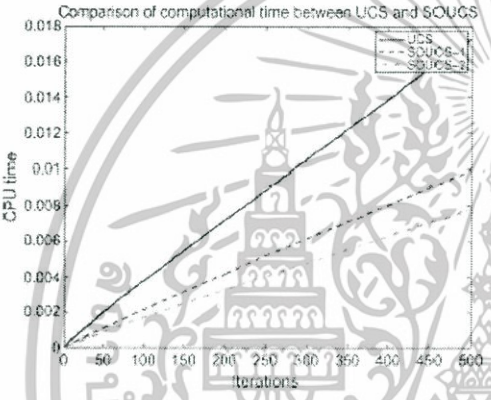


Fig. 8. Computational time of UCS, SOUCS-1, and SOUCS-2 on the circle 4 problem.

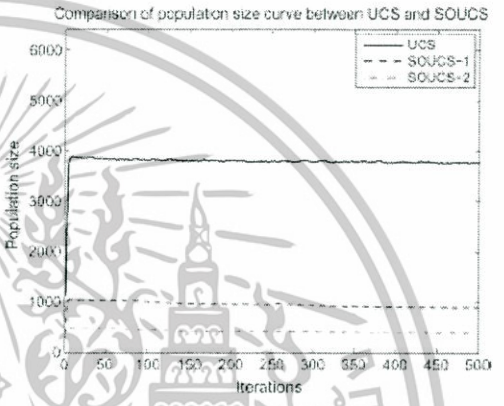


Fig. 9. Population size of UCS, SOUCS-1, and SOUCS-2 on breast-w and segment problems.

appears to be better than three of the other four papers. However, these results should be seen as indicative only, because it was not possible to unify the experiment setup for all methods. The neural ensemble approach presented in [36] achieved higher accuracy, using 20 neural networks each of which had 40 hidden nodes in the hidden layer. This setup implies a huge training cost.

C. Comparing UCS and ANN

Tables V and IX show the accuracy with UCS/SOUCS environments on the real data sets. Tables X and XI show the accuracy with ANN/SOANN environments on the same data sets.

Comparing the results, ANN/SOANN do better than UCS/SOUCS on the data sets that are a challenge for UCS due to the large number of classes or features. Otherwise, there is generally either no difference in accuracy, or else the results based on UCS are better.

In other words, the type of LCS that performs better depends on the characteristics of the problem. This applies generally in LCS research.

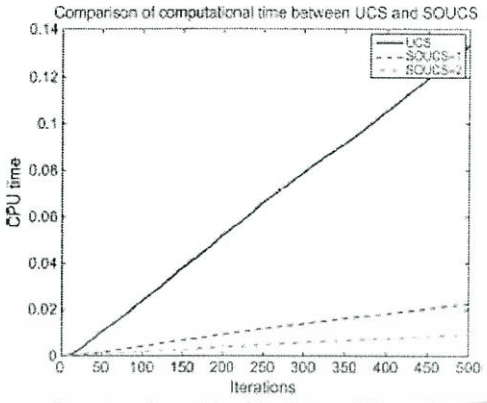
D. Summary

The results show that the proposed framework has good potential. In some cases, SOUCS/SOANN were able to achieve better accuracy than UCS/ANN, but most of the time they are able to get equivalent accuracy.

The results also showed that initial parameters' values—in particular, the population size—are important as they can improve or worsen the performance. This is also a problem in traditional UCS.

VI. An Investigation of the Execution Speed

In terms of computational time, it is obvious that SOUCS is much faster. An instance that gets passed from the SOM to one of the UCS would only need to scan little more than one quarter (for SOUCS-1) or one ninth (for SOUCS-2) of the population size of the single UCS. This is a significant reduction in processing time because LCSs spend most of the processing time scanning the population to form the match set.



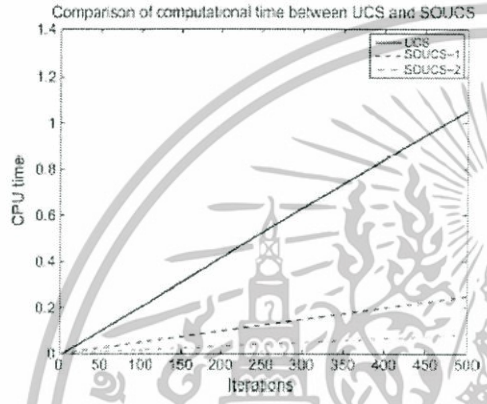


Fig. 10. Computational time of UCS, SOUCS-1, and SOUCS-2 on breast-w and segment problems.

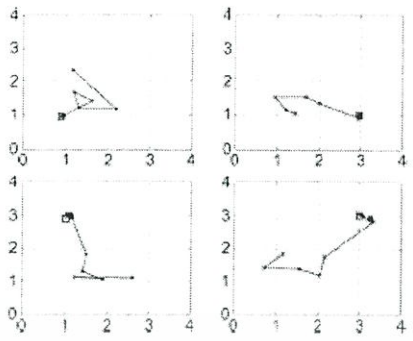


Fig. 11. Weight vector of each node in SOM 2×2 map in the circle 4 problem.

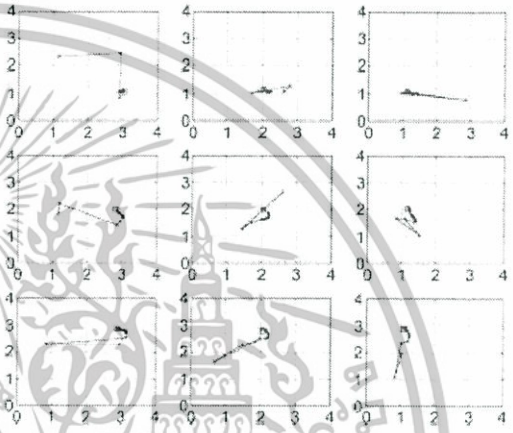


Fig. 12. Weight vector of each node in SOM 3×3 map in the circle 4 problem.

Fig. 7 shows the number of macro-classifiers generated in the circle problems. We can see that the number of macro-classifiers for both SOUCS environments is less than the corresponding number in the single UCS environment.

To show the saving in computational time resultant from using SOUCS, we plot the runtime of each system [measured by central processing unit (CPU) clock] in Fig. 8. The graph confirms that the SOUCS environment uses less computational time than the single UCS, and the SOUCS-2 environment uses still less.

Figs. 9 and 10 show the number of macro-classifiers and the computational time of the UCS and SOUCS environments on the UCI problems. Once more, the SOUCS environments use less computational time than the single UCS environment. It is important to notice the higher number of rules evolved by UCS after training.

VII. Search Space Decomposition

In this section, we first analyze problem decomposition using a SOM. Figs. 11 and 12 illustrate the dynamics of the SOM over time for the circle 4 problem. We use a small box to denote the state of the weight vector after each pass of the data, and a heavily outlined box to denote the final steady state. The trajectories show the self-organization as it happens for each cell. It is important to remind the reader that the SOM does not see the label for each instance; thus it works on the density of the data in the feature space. Since the data was generated uniformly in this 2-D space, it is logical to see that the steady state for the SOM is simply reached by dividing the space into four and nine uniform quadrants in the case of 2×2 and 3×3 SOM, respectively.

Figs. 13 and 14 show the decomposition of SOUCS as a whole in the last generation. The solid lines show the division of the data using the SOM, the dashed circles show the approximate decision boundaries generated by the individual UCS, and the class labels are shown as reported by the individual UCS. It is interesting to see that in some cases the classification problem in a cluster can be as trivial as everything is positive or negative, while the classification problem in other clusters can have the same complexity as the original problem.

To analyze the decomposition in the case of the UCI data sets, we use the visualization technique developed in [35]. This technique projects the high-dimensional feature space onto lower dimensional principal components space, then uses a convex-hull

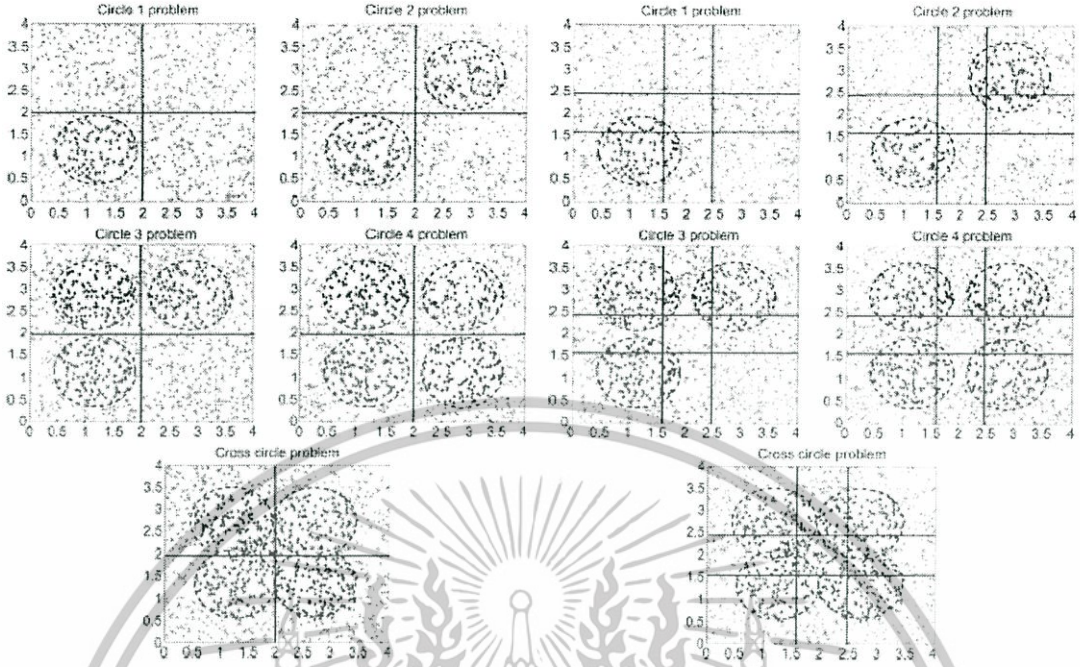


Fig. 13. Search space decomposition in SOM 2 × 2 map in the circle problems. Each rectangle represents data corresponding to each UCS in SOUCS. Circle 1 (first row, left-hand side), circle 2 (first row, right-hand side), circle 3 (second row, left-hand side), circle 4 (second row, right-hand side), and cross circle (last row).

Fig. 14. Search space decomposition in SOM 3 × 3 map in the circle problems. Each rectangle represents data corresponding to each UCS in SOUCS. Circle 1 (first row, left-hand side), circle 2 (first row, right-hand side), circle 3 (second row, left-hand side), circle 4 (second row, right-hand side), and cross circle (last row).

approximation to visualize the boundaries of the clusters. We only visualize the first three principal components by visualizing each combination of the 2-D graphs.

Figs. 15 and 16 show two example plots for the diabetes and breast-w problems in 2 × 2 and 3 × 3 SOUCS environments, respectively. The convex-hull boundary shows the different clusters, while the dots and crosses show the negative and positive labels, respectively. It is worth mentioning that as the clustering is occurring in the original feature space, some clusters may become overlapping in the principal component space. We can conclude that a SOM is able to decompose the problem, as useful patterns can be observed through the visualization techniques.

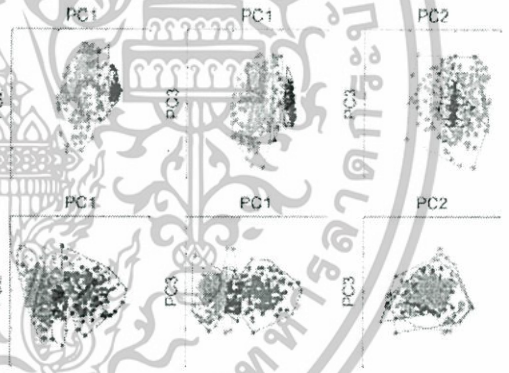


Fig. 15. Plot of each UCS's responsibility in PCA-plot (SOM 2 × 2 environment) on the breast-w (first row) and diabetes (second row) problems. Different shades represent different UCS, "+" and "o" markers represent class "0" and "1," respectively.

VIII. Conclusion and Future Work

In this paper, we introduced a new self-organized adaptive rule-induction system. The framework uses a SOM ANN as a pregate to decompose a problem into several simpler subproblems. Several classifiers are also employed to learn these subproblems.

The proposed architecture was tested on several synthetic and real-world data sets. It is shown that the accuracy of the proposed framework is better or equivalent to traditional ANN/UCS if the population size (in the case of UCS) is chosen carefully.

There are two important factors of SOUCS: the number of nodes in the SOM and the population size in each UCS. In some data sets, we cannot set the aggregate population size for all local UCSs in SOUCS equivalent to the population size of a single UCS, because the subproblems may have similar complexity as the overall problem.

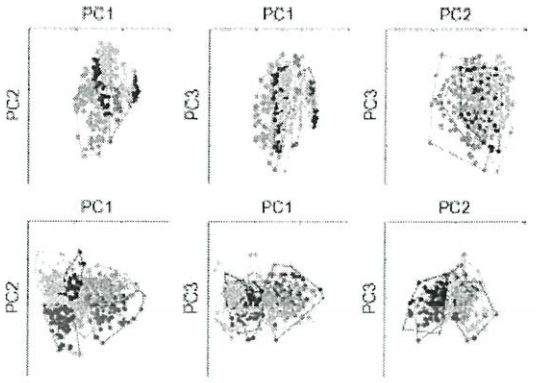


Fig. 16. Plot of each UCS's responsibility in PCA-plot (SOM 3 × 3 environment) on the breast-w (first row) and diabetes (second row) problems. Different shades represent different UCS. "+" and "o" markers represent class "0" and "1," respectively.

This suggests a possible future direction, where we will look into how to adapt the population size for each local UCS independently. Also, it will be interesting to come up with a mechanism to dynamically grow the SOM—maybe using the growing hierarchical SOM to automatically adjust the size of the SOM and make the interaction between the pregate and the classifiers more useful. At some point, there is a limit to the size of the SOM that makes sense, given the amount of data—it would be interesting to investigate the relationship between the number of available data instances and possible sizes for the SOM.

REFERENCES

[1] H. A. Abbass, "Pareto neuro-ensembles," in *Advances in Artificial Intelligence*, T. D. Gedeon and L. C. C. Fung, Eds. Berlin, Germany: Springer, 2003, pp. 554–566.

[2] A. Asuncion and D. J. Newman, UCI Machine Learning Repository. Univ. California Irvine, Irvine, CA, 2007.

[3] A. J. Bagnall and G. C. Cawley, "Learning classifier systems for data mining: A comparison of XCS with other classifiers for the Forest Cover dataset," in *Proc. IEEE/INNS Int. Joint Conf. Artificial Neural Netw.*, Portland, OR, July 20–24, 2003, vol. 3, pp. 1802–1807.

[4] E. Bernadó-Mansilla and J. M. Garrell-Guiu, "Accuracy-based learning classifier systems: Models, analysis and applications to classification tasks," *Evol. Comput.*, vol. 11, no. 3, pp. 209–238, 2003.

[5] E. Bernadó-Mansilla, X. Llorà, and J. M. Garrell-Guiu, "XCS and GALE: A comparative study of two learning classifier systems with six other learning algorithms on classification tasks," in *Proc. 4th Int. Workshop Learn. Classifier Syst.*, 2001, pp. 337–341.

[6] J. A. Blackard, "Comparison of neural networks and discriminant analysis in predicting Forest Cover types," Ph.D. dissertation, Dept. Forest Sci., Colorado State Univ., Fort Collins, CO, 1998.

[7] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[8] L. Bull and T. Kovacs, "Foundations of learning classifier systems: An introduction," in *Foundations of Learning Classifier Systems*. New York: Springer-Verlag, 2005, vol. 183, pp. 1–17.

[9] M. V. Butz, "Rule-based evolutionary online learning systems: Learning bounds, classification, and prediction," Ph.D. dissertation, Dept. General Eng., Univ. Illinois at Urbana-Champaign, Urbana, IL, 2004.

[10] M. V. Butz, *Rule-Based Evolutionary Online Learning Systems, A Principled Approach to LCS Analysis and Design*, ser. Studies in Fuzziness and Soft Computing. New York: Springer-Verlag, 2006, vol. 191.

[11] E. Cantuz-Paz, *Efficient and Accurate Parallel Genetic Algorithms*. Boston, MA: Kluwer, 2000.

[12] H. H. Dam, H. A. Abbass, and C. Lokan, "DXCS: An XCS system for distributed data mining," in *Proc. Gen. Evol. Comput. Conf.*, Washington, DC, 2005, pp. 1883–1890.

[13] H. H. Dam, H. A. Abbass, and C. Lokan, "Investigation on DXCS: An XCS system for distribution data mining, with continuous-valued inputs in static and dynamic environments," in *Proc. IEEE Congr. Evol. Comput.*, Edinburgh, Scotland, 2005, pp. 618–625.

[14] H. H. Dam, H. A. Abbass, C. Lokan, and X. Yao, "Neural-based learning classifier systems," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 26–39, Jan. 2008.

[15] H. H. Dam, K. Shafi, and H. A. Abbass, "Can evolutionary computation handle large datasets? A study into network intrusion detection," in *Austral. Conf. Artif. Intell.*, Sydney, Australia, 2005, pp. 1092–1095.

[16] P. W. Dixon, D. Corne, and M. J. Oates, "A preliminary investigation of modified XCS as a generic data mining tool," in *Advances in Learning Classifier Systems*. Berlin, Germany: Springer-Verlag, 2001, pp. 133–150.

[17] G. Folino, C. Pizzuti, and G. Spezzano, "Ensemble techniques for parallel genetic programming based classifiers," in *Lecture Notes in Computer Science*, C. Ryan, T. Soule, M. Keijzer, E. Tsang, R. Poli, and E. Costa, Eds. Berlin, Germany: Springer-Verlag, 2003, vol. 2610, pp. 59–69.

[18] G. Folino, C. Izzuti, and G. Spezzano, "Boosting technique for combining cellular GP classifiers," in *Lecture Notes in Computer Science*, M. Keijzer, U.-M. O'Reilly, S. M. Lucas, E. Costa, and T. Soule, Eds. Berlin, Germany: Springer-Verlag, 2004, vol. 3003, pp. 47–56.

[19] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. Eur. Conf. Comput. Learn. Theory*, 1995, pp. 23–37.

[20] B. Fritzke, "Some competitive learning methods," Inst. Neural Comput., Ruhr-Universität Bochum, Bochum, Germany, April 1997.

[21] S. Haykin, *Neural Networks: A Comprehensive Foundation*, ser. Engineering/Science/Mathematics. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[22] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975, republished by the MIT press, 1992.

[23] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, pp. 79–87, 1991.

[24] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural Comput.*, vol. 62, pp. 181–214, 1994.

[25] T. Kohonen, *Self-Organization and Associative Memory*. New York: Springer-Verlag, 1989.

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

[27] Y.-N. Law and C. Zaniolo, "An adaptive nearest neighbor classification algorithm for data streams," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2005, vol. 3721, pp. 108–120.

[28] Y. L. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. Jackel, "Back-propagation applied to handwritten zip code recognition," *NeurComp*, vol. 1, pp. 541–551, 1989.

[29] X. Llorà, K. Sastry, and D. E. Goldberg, "The compact classifier system: Scalability analysis and first results," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 596–603.

[30] X. Llorà, "Genetics-based machine learning using fine-gained parallelism for data mining," Ph.D. dissertation, Enginyeria i Arquitectura La Salle, Ramon Llull Univ., Barcelona, Catalonia, Spain, 2002.

[31] X. Llorà, K. Sastry, and D. E. Goldberg, "Binary rule encoding schemes: A study using the compact classifier system," in *Proc. Workshop Gen. Evol. Comput.*, New York, NY, 2005, pp. 88–89.

[32] O. Maimon and L. Rokach, "Improving supervised learning by feature decomposition," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2002, pp. 178–196.

[33] O. Maimon and L. Rokach, *Decomposition Methodology for Knowledge Discovery and Data Mining: Theory and Applications*. Singapore: World Scientific, 2005.

[34] M.-H. Nguyen, H. A. Abbass, and R. I. McKay, "A novel mixture of experts model based on cooperative coevolution," *Neurocomputing*, vol. 1–3, pp. 155–163, 2006.

[35] M.-H. Nguyen, H. A. Abbass, and R. I. McKay, "Analysis of CCME: Coevolutionary dynamics, automatic problem decomposition and regularization," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 38, no. 1, pp. 100–109, Jan. 2008.

[36] K. Peng, Z. Obradovic, and S. Vucetic, "Towards efficient learning of neural network ensembles from arbitrarily large datasets," in *Proc. 16th Eur. Conf. Artif. Intell.*, L. S. R. López de Mántaras, Ed., Aug. 2004, pp. 623–627.

[37] L. Rokach, O. Maimon, and I. Lavi, "Space decomposition in data mining: A clustering approach," in *Foundations of Intelligent Systems*, N. Zhong, Z. W. Ras, S. Tsumoto, and E. Suzuki, Eds. New York: Springer-Verlag, 2003, pp. 24–31.

[38] S. Saxon and A. Barry, "XCS and the Monk's problems," in *Learning Classifier Systems, From Foundations to Applications*. London, U.K.: Springer-Verlag, 2000, pp. 223–242.

[39] S. W. Wilson, "Classifier fitness based on accuracy," *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, 1995.

[40] S. W. Wilson, "Generalization in the XCS classifier system," in *Proc. 3rd Annu. Conf. Gen. Programm.*, J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, Eds., Madison, WI, 1998, pp. 665–674.

[41] S. W. Wilson, "Mining oblique data with XCS," in *Lecture Notes in Artificial Intelligence*, P. L. Lanzi, W. Stolzmann, and S. W. Wilson, Eds. Berlin, Germany: Springer-Verlag, 2002, pp. 158–174.

Hussein A. Abbass (SM'05) received the BA, B.Sc., PG-Dip, and M.S degrees from Cairo University, Cairo, Egypt, the M.Sc. degree from Edinburgh University, Edinburgh, Scotland, and the Ph.D. degree from the Queensland University of Technology, Australia.

Currently, he is a Professor and Chair of Information Technology at the University of New South Wales, Australian Defence Force Academy (UNSW@ADFA), Canberra, N.S.W., Australia. He is the Director of the Artificial Life and Adaptive Robotics Laboratory at UNSW@ADFA, an Advisory Professor at Vietnam National University, Ho-Chi Minh City, and an Adjunct Professor with Cairo University. He has over 170 refereed papers and is particularly interested in artificial neural networks, ensemble learning, evolutionary computation, multiagent systems, and multiobjective optimization.

Dr. Abbass is a senior member of the Australian Computer Society (ACS), the chair of ACS National Committee on Complex Systems, the chair of the IEEE Task Force on Complex Adaptive Systems and Artificial Life, and a member of a number of national and international committees including the IEEE technical committee on Data Mining and the IEEE working group on soft computing in the SMC society. He has been a technical co-chair and a member of the technical committee for numerous conferences in the field including a PC co-chair for CEC'07, SEAL 06, and IEEE-Alife'07.

Pornthep Rojanavasu received the B.Eng. degree in computer engineering from Chiang Mai University, Chiang Mai, Thailand, in 1999 and the M.Eng. degree in computer engineering from King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 2004, where he is currently working towards the Ph.D. degree in electrical engineering.

His research interests are large scale data mining, distributed data mining, problem decomposition, learning classifier system, and neural network.

Hai Huong (Helen) Dam received the B.Sc. degree in computer science (with honors) from Curtin University of Technology, Perth, W.A., Australia, in 2002 and the M.S. degree in computer science from University of Western Australia, Perth, W.A., Australia, in 2004. She is currently working towards the Ph.D. degree in computer science at the Artificial Life and Adaptive Robotics Laboratory, University of New South Wales, Australian Defence Force Academy, Canberra, N.S.W., Australia.

Her research interests are stream data mining, distributed data mining, learning classifier systems, and evolutionary algorithms.

Chris Lokan received the B.Sc. degree (with honors) and the Ph.D. degree from the Australian National University, Canberra, Australia, in 1980 and 1985, respectively, both in computer science.

Currently, he is a Senior Lecturer at the University of New South Wales (Australian Defence Force Academy campus), Canberra, N.S.W., Australia. His teaching and research concentrate on software engineering and software metrics. His main research interests are software size measures, software effort and cost estimation, and software benchmarking. Recently, his research has concentrated on the use of multicompany data sets for estimation and data mining.

Dr. Lokan is a member of the Association for Computing Machinery (ACM), the Computer Society of the IEEE, and the Australian Software Metrics Association.

Ouen Pinngern received the M.S. degree from Oregon State University, Corvallis, OR, in 1983 and the Ph.D. degree from University of Nebraska at Lincoln, Lincoln, in 1986, both in computer science.

He is an Associated Professor in the Department of Computer Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. His current research interests lie in the areas of soft computing, machine learning, data mining, information retrieval, and model of computation.

Proceedings
2009 2nd IEEE International Conference on Computer Science and Information Technology

August 8-11, 2009
Beijing, China

Edited by
Prof.Wenzheng Li. Dr.Jianhong Zhou

IEEE PRESS

Growing Rule-Based Induction System

Pornthep Rojanavasu and Boonwat Attachoo
Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Email: s8060022@kmitl.ac.th and boonwat@kmitl.ac.th

Ouen Pinngern
Department of Computer Science
Faculty of Science
Ramkhambaeng University
Email: ouen@ru.ac.th

***Abstract*—Learning Classifier Systems (LCSs) are rule-based systems that have widely been used in data mining over the last few years. This paper employs UCS, a supervised learning classifier system, that was a version of LCSs for classification in data mining tasks. In this paper, we propose an adaptive framework of a rule-based competitive learning environment. In this framework, a growing neural gas (GNG) is used to adaptively cluster the data instances as they arrive. Each instance is then assigned to based classifier, the UCS responsible for the corresponding cluster. Through this mechanism, the complexity of a classification problem is decomposed adaptively into sub-problems, each with a lower or equal complexity to the overall problem. Since each instance is exposed to a smaller population size than the single population approach, the throughput of the system increases. The experiments show that the proposed framework can decompose a problem adaptively into several subproblems. The accuracy rate of UCS in the distributed environment can also be better than the normal environment.**

I. Introduction

A Learning Classifier System (LCS) is a rule-based machine learning system that combines reinforcement learning and genetic algorithms. The reinforcement learning component is responsible for adjusting the strength of rules in the system according to some reward obtained from the environment. The genetic algorithm acts as an innovation discovery component which is responsible for discovering new - hopefully better - rules. The advantages of this approach are its rule-based representation, that can be easily interpreted by human; online learning ability; robustness due to the use of genetic algorithm.

In 1995, XCS [12] emerged as a type of LCS where the classifier's fitness depends on the prediction of its expected payoff instead of the actual prediction itself as in traditional LCS. XCS has since been wildly applied both reinforcement and data mining. In [6], [9] compared XCS with other machine learning such as the IB, C4.5, PART and the support vector machine on several datasets. The experiments showed that XCSs outperform the other learners in some datasets. In 2003, Bernadó at. el. [4] proposed UCS (supervised classifier system) based on XCS for supervised classification. UCS no longer relies on a feedback from the environment in terms of reward function. It is evaluated, however, on the associated class of the training instance as the general case for supervised classification. The study found that UCS is able to learn faster than XCS with smaller population size.

Nowadays, researchers in data mining have paid more attentions on large-scale problems. When LCSs are applied to these problems, their rule-based representation is normally required a large number of classifiers in order to capture the whole search space. As a result, LCSs suffer from a long computational time, which slows down the execution and limits their uses in the real world. Many researchers have proposed different methods to cope with large-scale problems such as distributed frameworks [7], ensemble framework [5].

When faced with complex data mining problems, automatic problem decomposition has been a relatively new area of research looking at turning a complete classification problem into smaller ones that can be handled easier by independent learning machines. In this paper, we propose the adaptive framework of UCS on top of a growing neural gas (GNG). The framework uses a growing neural gas as a pre-gate to decompose a problem into several simpler subproblems which is handled by difference UCS. The GNG can be seen in two different ways. First, it decomposes the problem by clustering the data (the class label is not presented to the GNG). Second, the GNG can be seen as a router in a distributed environment, which assigns the data as they arrive to a node that will take care of classifying the incoming instance.

The rest of this paper is organized as follows. Section II provides a short review of problem decomposition, GNG and LCSs. Section III describes our proposed framework. The experimental design and results are described in Section IV. Finally, conclusion and future works are given in Section V.

II. Background Materials

A. Problem Decomposition

The ongoing rapid growth of technology has created larger and more complex problems and therefore challenges many traditional data mining algorithms in terms of execution time. One possible remedy to this challenge is to decompose a whole complex problem into several subproblems, where each simpler subproblem is solved by a data mining technique.

There are several advantages for decomposing data mining problems as pointed in [11]: increasing the performance, simplifying conceptual problems, enhancing feasibility for huge databases, providing clearer and comprehensible results, reducing an execution time, and providing an opportunity of using different techniques for individual subproblems.

A decomposition approach in classification can be categorized into one of two types [11]:

978-1-4244-4520-2/09/$25.00 ©2009 IEEE

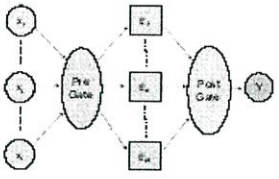


Fig. 1. A generic description to ensemble research.

- Intermediate concept decomposition: here, the problem is projected onto an intermediate space, where the decomposition takes place to define several subproblems with different and simpler concepts. For example, one may use Gaussian kernels to do the mapping to the intermediate space.
- Original concept decomposition: here, problem decomposition is undertaken directly on the problem without a mediator. The original problem is divided into several subproblems by partitioning the training set. The mixture of experts model is an example of this category, where a gate is used to weight the classifiers differently based on different input instances.

Abbass [1] summarized the literature of ensemble learning in a simple graph, presented in figure 1. Two types of gates can be identified in the system: pre-gate and post-gate. A pre-gate is used as a mechanism to decide on which input instance affects which member of the ensemble. Research in this area is mainly concerned with getting members of the ensemble to specialize on different parts of the input space. The study of this pre-gate falls under the umbrella of *competitive learning*, where the main objective is to compete between different learning systems for each input.

A post-gate is used to decide on how to combine different outputs made by the members in the ensemble. The motivation to this learning approach comes from the area of expert systems, where a committee of experts is used to make decisions by simple voting or weight voting. If one member responses to each situation, there will be no conflict within the committee.

This paper will focus on the original concept decomposition by using competitive learning and adaptive classification. The idea is that as the clustering algorithm is self-organizing itself on the dataset, the classifiers associated with the different (changing) clusters adaptively learn the classification problem. The problem is decomposed (either explicitly or implicitly) into several subproblems. Each subproblem is then handled by a classifier. The final decision is made by an appropriate classifier.

B. Growing Neural Gas

Competitive learning or clustering can be described as the process of organizing a collection of k-dimensional data into groups whose members have some similar features. The most well-know clustering algorithm is K-mean clustering algorithm and Self-Organizing Map(SOM). These two methods, however, need to determine a priori the number of nodes and the network to be trained is fixed throughout, in case of SOM. In this paper, we use Growing Neural Gas which is soft competitive learning, dynamic network where the new nodes are added or deleted based on the error characteristics of the nodes and network, tightly couple it with an UCS.

Growing Neural Gas is a self-organizing neural network which is introduced by Bernd Fritzke [8]. Network layer consists of a set of units connected by edges and distributed over the input space. The learning algorithm can be concluded as follows.

The network is started with two units a and b at random weight w_a and w_b The input vector $x(t)$ is fed into every unit in the network to identify the nearest unit s_1 and the second-nearest unit s_2. If there is a connection between s_1 and s_2 then set the age of edge to 0. If there is no connection then the edge is created. The age of all edges emanating from s_1 are increased by 1. Then, add the error between $x(t)$ and the nearest unit to a local variable as follows:

$$\Delta error(s_1) = ||w_{s_1} - x(t)||^2 \tag{1}$$

The weight vector of the nearest unit and its direct topological neighbors are adapted by fraction ϵ_b and ϵ_n of the distance to $x(t)$, respectively. The edges which have age larger than a given a_{max} are removed and also the nodes which have no more emanate edges are removed. If the number of input exceed an integer multiple of given parameter λ a new unit q is inserted between unit n which has the largest error and its neighbor unit m which has largest error. The weight of new unit q is set to the average weight between unit n and m. The connection between unit q and unit m and n are created and the original connection between unit m and n is removed. The error of unit n and m is decreased by a given fraction, while the error of new unit is set to the average of unit n and m. Lastly, the error of all unit is decreased by a given fraction. The process is repeated until found the stopping criterion.

C. Learning Classifier Systems

LCS is a rule-based evolutionary learning classifier system, in which each classifier implements a partial task to the target problem. A typical goal of LCS is to evolve a population of classifiers $[P]$ to represent a complete solution to the target problem. LCS can be used either for reinforcement learning (e.g XCS) or supervised learning (e.g UCS). This paper focuses mainly on UCS (supervised classifier system).

Each classifier of UCS consists of the *Condition* (the body of the rule), the *Action* (the prediction of the classifier) and some parameters. The *Condition* refers to several environmental states, to which the classifier may match. The *Action* is an outcome if the classifier is fired/activated. The key parameter associated with a classifier is the fitness F, which measures how good the classifier is relative to the rest of the population. Another two important parameters are the *numerosity* and

experience. The *numerosity* parameter records the number of copies of the classifier. Whenever a new classifier is introduced, the population is scanned through to check if a copy of it already exists. If it does not, the classifier is added to the population; otherwise, the *numerosity* value is incremented by one. The *experience* parameter indicates how often the classifier is chosen for making the prediction; in other words, how general the classifier is.

During the learning cycle, the system repeatedly receives an input from the environment. Since UCS [2] [4] is a supervised learner, an input instance is associated with a target class.

Firstly, UCS finds a match set $[M]$ with all the matching classifiers in the population. A *correct set* $[C]$ is formed, containing those classifiers in $[M]$ that have the same *action* as the input. If $[C]$ is empty, *covering* is applied, where a classifier that matches the input is created and assigned the same outcome as the input. UCS is an incremental learner, where knowledge is updated as more data becomes available. All parameters of the classifiers in $[M]$ are revised for each training instance according to whether they belong to $[C]$ or not , reflecting the system's response to new knowledge.

The fitness of classifiers is based on the accuracy, which is computed as the ratio of correct classifications by the classifier to the number of times the classifier has been in the match set:

$$acc = \frac{N_{correct}}{N_{matches}} \tag{2}$$

The fitness is computed as a function of accuracy:

$$F = (acc)^{\nu} \tag{3}$$

where ν is a predefined constant.

GA is invoked in $[C]$ if the average time since the last application of the GA of classifiers in $[C]$ surpasses a user-defined threshold. Two parents are selected from $[C]$ with probability proportional to their fitness. Two offspring are generated by reproducing, crossing-over, and mutating the parents with certain probabilities. Offspring are inserted in $[P]$ if they are not subsumed by the parents. If the population size hits a predefined limit, some classifiers are removed by voting within the population.

In the exploitation phase, an input instance is given with unknown class. After finding the match set $[M]$, UCS proposes a class, from the vote weighted by fitness of each classifier in the match set.

III. The Proposed Framework

Figure 2 depicts the proposed framework. As a pre-gate, we use a growing neural gas (GNG) to decompose the problem. A classification algorithm is employed to handle each sub-problem independently. UCS is used as the classifier leaning environment. We call the combined system Growing-UCS (GUCS). Both decomposition and learning occur together; thus the system decomposes the problem as it learns the class boundaries.

GNG is used as the pre-gate in the proposed framework. It has two main roles. First, it decomposes the problem by clustering the data (note that the class label is not presented to GNG). Second, GNG can be seen as a router in a distributed environment that assigns the data as they arrive to a node that will take care of classifying the incoming instance. Given any data instance, there is only one UCS responsible for it.

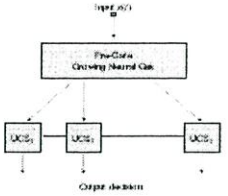


Fig. 2. The proposed framework. SOM is used to routes the data instances as they arrive to one of the LCS.

In training phase, the system is trained incrementally. For each training instance, the nearest unit is first identified based on the similarity function. The weights of the nearest unit and second-nearest are updated and the other parameters are also updated. The instance is then fed into the UCS coupled with the winning node. A complete training process is carried out at that node. In the case of UCS, covering might occur if the match set is empty, parameters of classifiers in the match set are updated, and GA might be activated.

There are two questions in training phase. The first question is what would happen when we insert a new unit in GNG? Exactly, we need to create the new UCS bundled with the new unit. The new UCS can start from some exist knowledge by copy the existing population of UCS which is bundled with the nearest unit, instead of starting from the scratch without any knowledge. Indeed, the copied population of new UCS maintains some useful classifiers and also some unuseful classifiers. These unuseful classifiers, however, will be deleted by GA in the future. The second question is what would happen when we delete a unit in GNG? We can simply delete the UCS from the system.

In the testing phase and given a testing instance, GNG first chooses the nearest unit. The instance is then routed to the UCS coupled with the nearest unit to make a prediction. The testing phase does not involve any learning in either the GNG or the UCS.

IV. Experimental Result

To illustrate the benefits of the proposed framework, we will set a maximum number of node of GNG equal to 2 and 4 nodes. Each node in GNG is associated with a complete UCS; thus, for 2 nodes GNG, there are two independent UCS environments forming the GUCS. The parameter settings for UCS are fixed for all environments similar to those used in Bernado [4] as follows: $\nu = 5, \theta_{GA} = 50, \chi = 0.8, \mu = 0.04, \theta_{del} = 50, \theta_{sub} = 50$. All results - unless stated otherwise - are the average over 10-fold cross validation.

We first attempted to make the sum of the population sizes of GUCS equal to the population size of a single UCS. For

TABLE I
ACCURACY RATES OF UCS, GUCS-2, GUCS-4 ON ARTIFICIAL PROBLEM

	Single UCS	GUCS-2	GUCS-4
Circle - Triangle	0.928±0.007	0.929±0.011	0.937±0.008

example, if we set the population size as $N = n$ in a UCS normal environment, the population size for each UCS in the GUCS environment is set to $N = n/x$, where x is the number of UCS in GUCS environments.

A. Analyzing the Proposed Framework on Synthetic Dataset

To understand the behavior of GUCS, we first test it on a synthetic problem as shown in figure 3. The problem has 2 attributes with boxing constraints between 0 and 4. In the problem, 8000 data points are sampled using a uniform random number generator. The problem has 5 classes, two triangles and two circles with all different class plus one more class which is the data outside all shapes.

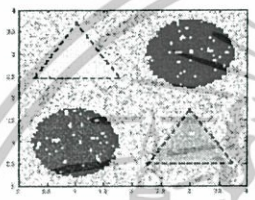


Fig. 3. The 5 class Circle-Triangle problem. Each *circle*, each *triangle* and data outside boundary have different classes.).

In artificial problems, the population size N is set to 1000 for single UCS. For 2 nodes GNG, called GUCS-2, the population size for each UCS is set to 500. For 4 nodes GNG, called GUCS-4, the individual population size is set to 250.

Table I lists the accuracy rates of the different environments on the synthetic datasets. This table shows that GUCS works fine. In GUCS-4, it is better than a single UCS. In terms of computational time complexity, it is obvious that GUCS is much faster since an instance that gets passed from the GNG to one of the UCS would need to scan the equivalent of 0.5 and 0.25 - for GUCS-2 and GUCS-4 respectively - of the population size of the single UCS. This is a significant reduction in time complexity.

Figure 4 shows the number of macro classifiers of circle problems. The number of macro classifier of both GUCS environments less than single UCS. Then both GUCS environments use less searching time.

Figure 5 shows a possible decomposition of GUCS as a whole in the last generation. The green and blue dash lines show the division of the data using GNG, the black dash line shows the approximated decision boundary generated by the individual UCS. In this case, we can reduce complexity of each UCS from five classes problem into three classes problem and two classes problem for GUCS-2 and GUCS-4, respectively.

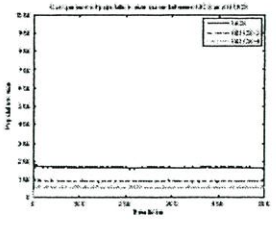


Fig. 4. The pop. size of UCS, GUCS-2 and GUCS-4 on artificial problem.

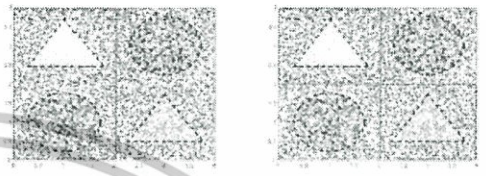


Fig. 5. Each rectangle represent data corresponded by each UCS for 2 nodes GUCS(left) and 4 node GUCS(right), respectively.).

B. UCI Datasets

In this section, we test GUCS on a number of datasets from the UCI repository [3], except the tao could be found in [10]. Table II shows the properties of the datasets used in this paper. *Inst* (the number of instances), *Fs* (the number of features), *R* (the number of real-valued features), *I* (the number of integer-valued features), and *C* (the number of outcomes).

TABLE II
THE PROPERTIES OF TESTING DATASETS

Problem	# Inst	# Fs	# R	# I	# C
Breast-w	699	9	0	9	2
Balance-scale	625	4	4	0	3
Diabetes	768	8	8	0	2
Glass	214	9	0	0	6
Iris	150	4	4	0	3
Segment	2310	19	19	0	7
Tao	1888	2	2	0	2
Vehicle	946	17	17	0	4

We use a population size of 6400 for the single UCS, 3200 for each UCS in GUCS-2, and 1600 for each UCS in GUCS-4. Table III shows the accuracy rate of single UCS and the GUCS environment. We can see that GUCS-2 is significantly better on 1 problem, equivalent on 6 problems and perform worse in 1 problem. GUCS-4 was significantly better on 2 out of 8 problems, comparable results were achieved on 5 problems and perform worse in 1 problem. However in terms of computational time complexity, it is obvious that GUCS environments is much faster since an instance that get passed from the GNG to one of UCS would need to scan equivalent of 0.5 and 0.25 - for GUCS-2 and GUCS-4 respectively - of the population size of the single UCS. This is a significant reduction in time complexity.

TABLE III
THE MEAN AND DEVIATION OF THE ACCURACY OF UCS AND GUCS ON UCI DATASETS.

	UCS	GUCS-2	GUCS-4
Breast-w	0.966±0.015	0.962±0.022	0.964±0.016
Balance-scale	0.815±0.027	0.817±0.044	0.855±0.005 •
Diabetes	0.766±0.056	0.753±0.001	0.768±0.022
Glass	0.704±0.055	0.706±0.056	0.692±0.074
Iris	0.967±0.012	0.953±0.010	0.953±0.010
Segment	0.955±0.011	0.943±0.011 ○	0.951±0.020
Tao	0.880±0.012	0.940±0.021 •	0.956±0.016 •
Vehicle	0.715±0.031	0.701±0.025	0.681±0.026 ○

[a] Differences that are statistically significant at a significance level of .05 are denoted by a • if the GUCS environment is better, and by ○ if the GUCS environment is worse. N=6400 for the single UCS, N=3200 for each UCS in GUCS-2, N=1600 for each UCS in GUCS-4.

TABLE IV
THE MEAN AND DEVIATION OF THE ACCURACY OF UCS AND GUCS ON UCI DATASETS

	UCS	UCS$^+$	GUCS-2$^+$	GUCS-4$^+$
Segment	0.955±0.011	0.968±0.014	0.967±0.009 ▲	0.970±0.011 ▲
Vehicle	0.715±0.031	0.719±0.021	0.713±0.023	0.697±0.030

[a] The statistical difference between GUCS and UCS is denoted by ▲ if the other system are better, and by ▽ if the other system are worse. The statistical difference between the other config and UCS$^+$ is denoted by • if the other system are better, and by ○ if the other system are worse.

GUCS-4 perform better in balance-scale and tao problem because GNG helps system divide problems into several easier sub-problems which can be use the number of rules less than using single UCS. However, in segment and vehicle problem both have many numbers of class and features, GUCS environment perform worse than single UCS. After use GNG divide problem each sub-problems might be still large. We hypothesize that because of the population size of each UCS in GUCS environments is too small.

Therefore, we undertook extra experiment where we increased the population size in each UCS to N = 6400 in GUCS-2, called GUCS-2$^+$ and N = 3200 in GUCS-4, GUCS-4$^+$. We also compared this against a population size of N = 12800 for the single UCS, called UCS$^+$. Table IV shows the result of this experiment. In the segment problem, both GUCS-2$^+$ and GUCS-4$^+$ achieve better accuracy than UCS and comparable accuracy with UCS$^+$. In the vehicle problem, GUCS-2$^+$ and GUCS-4$^+$ are able to achieve similar accuracy as both UCS and UCS$^+$.

Next, we overlook at the number of macro classifiers which holds a distinct rule within the population. Figure 6 shows the number of macro classifiers of glass and diabetes problem, respectively. The number of macro classifier of both GUCS environments less than single UCS. Then both GUCS environments use less searching time because of GUCS environments always have search space less than single UCS.

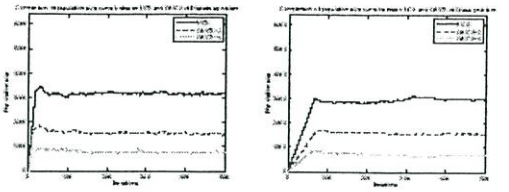


Fig. 6. The pop. size of UCS, GUCS-2 and GUCS-4 on the diabetes and glass problem, respectively).

V. Conclusion and Future Work

In this paper, we introduced a new adaptive rule-induction system. The system uses a growing neural gas and an evolutionary learning classifier system known as UCS. The advantages of the combine architecture are shown in terms of time complexity and accuracy. The new architecture, called GUCS, was tested on synthetic, real-world datasets and large problem dataset. It is shown that the performance is equivalence or better in terms of accuracy and always improved in terms of speed. In some case, after we divide problem into smaller subproblems, each subproblems might be still large or has the same complexity as the original problem. Therefore, we must carefully select the population size of each UCS. For future work, we will look into the issue of adaptive population of each UCS in GUCS that makes the interaction between the pre-gate and the post-gate in distributed environment.

References

[1] H. A. Abbass. Pareto neuro-ensembles. In T. Gedeon and L. Fung, editors, *Advances in Artificial Intelligence*, pages 554–566. Springer Berlin / Heidelberg, 2003.

[2] A. Albert Orriols-Puig and E. Bernadó-Mansilla. A further look at UCS classifier system. In *Proceedings of the 9th International Workshop on Learning Classifier Systems (IWLCS'06)*, 2006.

[3] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[4] E. Bernadó-Mansilla and J. M. Garrell-Guiu. Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. *Evolutionary Computation*, 11(3):209–238, 2003.

[5] L. Bull and T. Kovacs. Foundations of learning classifier systems: An introduction. In *Foundations of Learning Classifier Systems*, volume 183, pages 1–17. Springer-Verlag, June 2005.

[6] M. Butz. *Rule-Based Evolutionary Online Learning Systems: A Principled Approach to LCS Analysis and Design*, volume 191 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, 2006.

[7] H. H. Dam, P. Rojanavasu, H. A. Abbass, and C. Lokan. Distributed learning classifier systems. In E. B. Larry Bull and J. Holmes, editors, *Learning Classifier Systems in Data Mining*.

[8] M. Dittenbach, D. Merkl, and A. Rauber. The growing hierarchical self-organizing map. In *Proceedings of the International Joint Conference on Neural Networks 2000 (IJCNN2000)*, Como, Italy, 2000.

[9] P. W. Dixon, D. Corne, and M. J. Oates. A preliminary investigation of modified XCS as a generic data mining tool. In *IWLCS 4th*, pages 133–150. Springer-Verlag Berlin Heidelberg, 2001.

[10] X. Llorà. *Genetics-based machine learning using fine-grained parallelism for data mining*. PhD thesis, Enginyeria i Arquitectura La Salle, Ramon Llull University, Barcelona, Catalonia, European Union, 2002.

[11] O. Maimon and L. Rokach. *Decomposition Methodology for Knowledge Discovery and Data Mining: Theory and Applications*. World Scientific Publishing, 2005.

[12] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

ประวัติผู้เขียน

เด็กชายพรเทพ โรจนวสุ เกิดเมื่อวันที่ 19 พฤศจิกายน 2521 ที่จังหวัดราชบุรี เติบโตท่ามกลางสวนลำไยที่อำเภอเกาะคา จังหวัดลำปาง สำเร็จการศึกษาชั้นอนุบาลและประถมศึกษาจากโรงเรียนอัสสัมชัญลำปางในปีการศึกษา 2533 สำเร็จการศึกษาชั้นมัธยมศึกษาตอนต้นจากโรงเรียนบุญวาทย์วิทยาลัยจังหวัดลำปางในปีการศึกษา 2536 ในระหว่างศึกษาชั้นมัธยมศึกษาปีที่ 5 ณ.โรงเรียนบุญวาทย์วิทยาลัยนั้นได้สำเร็จการศึกษาชั้นมัธยมศึกษาตอนปลายจากศูนย์การศึกษานอกโรงเรียนจังหวัดลำปางในปีการศึกษา 2538 พร้อมกับก้าวเข้าสู่รั่วมหาวิทยาลัยในปีเดียวกันจนกระทั่งสำเร็จการศึกษาในระดับปริญญาตรีวิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์) จากมหาวิทยาลัยเชียงใหม่ในปีการศึกษา 2542 และสำเร็จปริญญาโทวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังในปีการศึกษา 2547

ปี พ.ศ. 2542-2544 เป็นลูกจ้างชั่วคราวตำแหน่งอาจารย์สังกัดภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

ปี พ.ศ. 2547-ปัจจุบัน เป็นพนักงานมหาวิทยาลัยตำแหน่งอาจารย์สังกัดสำนักวิชาเทคโนโลยีสารสนเทศและการสื่อสาร สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยนเรศวรพะเยา (http://www.ict.pyo.nu.ac.th/pornthep)