รายงานวิจัยฉบับสมบูรณ์

โครงการ อัลกอริทึมเรียนรู้สำหรับโดเมนหลายกลุ่มและโดเมนมีสัญญาณรบกวน

Learning Algorithms for Multiclass and Noisy Domains

โดย บุญเสริม กิจศิริกุล

สิงหาคม 2549

สัญญาเลขที่ RSA4680024

รายงานวิจัยฉบับสมบูรณ์

โครงการ อัลกอริทึมเรียนรู้สำหรับโดเมนหลายกลุ่มและโดเมนมีสัญญาณรบกวน
Learning Algorithms for Multiclass and Noisy Domains

บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณสำนักงานกองทุนสนับสนุนการวิจัยที่ได้ให้ทุนตลอดระยะเวลา 3 ปี (15 สค. 2546 — 14 สค. 2549) สำหรับโครงการ "อัลกอริทึมเรียนรู้สำหรับโดเมนหลายกลุ่มและโดเมนมีสัญญาณรบกวน" และขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ได้ให้โอกาส สถานที่ตลอดจนทรัพยากรต่างๆ ที่ใช้ในงานวิจัย ขอขอบคุณผู้ช่วยวิจัยและผู้มีส่วนร่วมในโครงการนี้ทุกท่าน

บุญเสริม กิจศิริกุล

สิงหาคม 2549

บทคัดย่อ

รหัสโครงการ: RSA4680024

ชื่อโครงการ: อัลกอริทึมเรียนรู้สำหรับโดเมนหลายกลุ่มและโดเมนมีสัญญาณรบกวน

ชื่อนักวิจัย: นายบุญเสริม กิจศิริกุล

E-mail Address: boonserm.k@chula.ac.th

ระยะเวลาโครงการ: 15 สค. 2546 — 14 สค. 2549

ในงานวิจัยนี้เราเสนอวิธีการสำหรับ (1) ปรับปรุงซัพพอร์ตเวกเตอร์แมชชีนสำหรับจัดการกับปัญหาข้อมูลหลายกลุ่ม และ (2) ทำให้การโปรแกรมตรรกะเชิงอุปนัย (ไอแอลพี) จัดการกับข้อมูลมีสัญญาณรบกวน เอสวีเอ็มถูกพัฒนาขึ้นเพื่อปัญหาการจำแนกประเภทข้อมูลที่มีสองกลุ่มได้อย่างมีประสิทธิภาพในการใช้งานจริง วิธีการดั้งเดิมสำหรับแก้ปัญหาของข้อมูลหลายกลุ่มทำได้โดยการนำฟังก์ชันตัดสินใจแบบสองกลุ่มมารวมกัน กราฟไม่มีวงแบบมีทิศทางสำหรับการตัดสินใจ (ดีดีเอจี) เป็นวิธีการที่รู้จักกันดีสำหรับเอสวีเอ็มแบบหลายกลุ่มซึ่งมีข้อดีในเรื่องของการใช้งานที่รวดเร็วและให้ความถูกต้องในการจำแนกประเภทที่ใกล้เคียงกับวิธีการอื่นๆ เราได้ปรับปรุงดีดีเอจีโดยนำเสนอกราฟไม่มีวงมีทิศทางแบบปรับได้ (เอดีเอจี) ซึ่งมีโครงสร้างที่ปรับแก้มาจากดีดีเอจีและมีจำนวนระดับตัดสินใจที่น้อยลง เอดีเอจีให้ความถูกต้องสูงกว่าดีดีเอจีในขณะที่ยังคงเวลาในการคำนวณต่ำ นอกจากนั้นเราได้นำเสนอเวอร์ชันปรับปรุงของเอดีเอจีที่เรียกว่า กราฟไม่มีวงมีทิศทางและปรับได้แบบจัดเรียงใหม่ (อาร์เอดีเอจี) เพื่อให้หาเอดีเอจีที่ดีสุดจากเอดีเอจีที่เป็นไปได้ทั้งหมดโดยใช้อัลกอริทึมจัดเรียงใหม่ร่วมกับการจับคู่สมบูรณ์น้ำหนักน้อยสุด ผลการทดลองกับชุดข้อมูลหลายชุดแสดงให้เห็นว่าวิธีการของเราให้ความถูกต้องที่สูงกว่าวิธีการดั้งเดิม

ไอแอลพีเป็นเทคนิคที่มีประสิทธิภาพสำหรับการทำเหมืองข้อมูลเชิงสัมพันธ์ แต่เมื่อประยุกต์ใช้ไอแอลพีในโดเมนมีสัญญาณรบกวน กฎที่ได้จากไอแอลพีมักประสบปัญหาเรื่องการปรับเหมาะเกินไป เรานำเสนอวิธีการสำหรับทำให้ไอแอลพีสามารถจัดการกับปัญหานี้ เริ่มจากการนำเสนอวิธีการเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งซึ่งสามารถจัดการกับข้อมูลมีสัญญาณรบกวนได้อย่างมีประสิทธิภาพ เนื่องจากต้นทุนในการคำนวณที่สูงสำหรับการเรียนข่ายงานเบส์ลำดับที่หนึ่งโดยตรง ทำให้เราปรับใช้ระบบไอแอลพีร่วมกับตัวเรียนรู้ข่ายงานเบส์เพื่อสร้างข่ายงานเบส์ลำดับที่หนึ่ง เราเสนออัลกอริทึมดึงลักษณะสำคัญเพื่อสร้างลักษณะสำคัญสำหรับกฎไอแอลพีและใช้ลักษณะสำคัญเหล่านี้เป็นโครงสร้างหลักของข่ายงานเบส์ลำดับที่หนึ่ง ผลการทดลองแสดงให้เห็นว่าข่ายงานเบส์ลำดับที่หนึ่งทำงานได้ดีกว่าระบบไอแอลพีดั้งเดิม เรายังได้นำเสนอวิธีการเรียนรู้แบบผสมเพื่อทำให้นิวรอลเน็ตเวิร์กสามารถจัดการกับโปรแกรมตรรกะลำดับที่หนึ่งได้โดยตรง วิธีการนี้เรียกว่านิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่ง วิธีการนี้ใช้นิวรอลเน็ตเวิร์กป้อนไปข้างหน้าแบบมาตรฐานและได้ผสมผสานการเรียนรู้เชิงอุปนัยจากตัวอย่างและความรู้ภูมิหลังร่วมด้วย เรานำเสนอวิธีการสำหรับกำหนดการแทนค่าตัวแปรที่เหมาะสมในการเรียนรู้นิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่งโดยใช้การเรียนรู้แบบหลายตัวอย่างย่อย ผลการทดลองกับปัญหาการเรียนรู้แบบตรรกะลำดับที่หนึ่งทั้งหมดสองปัญหาแสดงให้เห็นว่าวิธีการที่นำเสนอทำงานได้ดีกว่า PROGOL ซึ่งเป็นระบบไอแอลพีที่ทันสมัย

คำหลัก: ซัพพอร์ตเวกเตอร์แมชชีน, การจำแนกประเภทแบบหลายกลุ่ม, กราฟไม่มีวงวนมีทิศทางแบบปรับได้, กราฟไม่มีวงวนมีทิศทางและปรับได้แบบจัดเรียงใหม่, ข่ายงานเบส์ลำดับที่หนึ่ง, นิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่ง

Abstract

Project Code: RSA4680024
Project Title: Learning Algorithms for Multiclass and Noisy Domains
Investigator: Boonserm Kijsirikul
E-mail Address: boonserm.k@chula.ac.th
Project Period: August 15, 2003 – August 14, 2006

In this research, we propose methods for (1) extending Support Vector Machines (SVMs) for dealing with multiclass problems, and (2) enabling Inductive Logic Programming (ILP) for dealing with noisy data. SVMs were primarily designed for two-class classification problems with their outstanding performance in real world applications. Previous methods for solving the multiclass problem of SVMs are typically to consider the problem as the combination of two-class decision functions. The Decision Directed Acyclic Graph (DDAG) is a well-known method for multiclass SVMs that has advantage of fast evaluation time and provides classification accuracy comparable to other methods. Motivated by DDAG, we propose the Adaptive DAG (ADAG): a modified structure of DDAG that has a lower number of decision levels. ADAG improves the accuracy of DDAG while it maintains low computational requirement. Next, we propose an enhancement version of ADAG, called Reordering Adaptive Directed Acyclic Graph (RADAG), to find one best ADAG from all possible ADAGs by using the reordering algorithm with minimum-weight perfect matching. Experiment results on several datasets denote that our methods give higher accuracies than those of the previous methods.

ILP is an efficient technique for relational data mining, but when ILP is applied in noisy domains, the rules induced by ILP often struggle with the overfitting problem. We propose methods for enabling ILP to deal with this problem. We first propose a method for learning first-order Bayesian network (FOBN) which can handle noisy data powerfully. Due to a high computation cost for directly learning FOBN, we adapt an ILP system and a Bayesian network learner to construct FOBN. We propose a feature extraction algorithm to generate features from ILP rules, and use these features as the main structure of the FOBN. The experimental results show that FOBN performs better than a traditional ILP system. We also propose a novel hybrid learning method to enable neural networks to handle first-order logic programs directly. The proposed method, called First-Order Logical Neural Network (FOLNN), employs the standard feed-forward neural network and integrates inductive learning from examples and background knowledge. We propose a method for determining the appropriate variable substitution in FOLNN learning by using multiple-instance learning. The experimental results on two first-order learning problems show that the proposed method performs better than PROGOL, the state-of-the-art ILP system.

Keywords: Support Vector Machines, Multiclass Classification, Adaptive Directed Acyclic Graphs, Reordering Adaptive Directed Acyclic Graphs, First-Order Bayesian Networks, First-Order Logical Neural Networks.

I. เนื้อหางานวิจัย

อัลกอริทึมการเรียนรู้ของเครื่อง (Machine Learning algorithms) ส่วนใหญ่ถูกพัฒนาขึ้นมาเพื่อใช้กับปัญหาที่มีข้อมูลตัวอย่าง (training data) จากกลุ่มเพียง 2 กลุ่ม (class) และมักไม่มีความทนทานต่อข้อมูลมีสัญญาณรบกวน (noisy data) แต่เมื่อเรานำอัลกอริทึมการเรียนรู้ของเครื่องไปใช้งานจริง จะพบว่าปัญหาในโลกจริง (real-world problems) มักจะมีลักษณะที่เป็นแบบหลายกลุ่มและมักมีสัญญาณรบกวน ทำให้อัลกอริทึมใช้งานไม่ได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น งานวิจัยของ [Kijsirikul et al., 2001] แสดงให้เห็นว่าเมื่อนำระบบเรียนรู้ของเครื่อง Progol [Muggleton, 1995] มาใช้งานกับปัญหาการรู้จำตัวอักษรไทยจากข้อมูลภาพที่สแกน ซึ่งเป็นปัญหาของข้อมูลหลายกลุ่ม (กลุ่ม 'ก', กลุ่ม 'ข', กลุ่ม 'ฃ', กลุ่ม 'ค', ...) และเป็นข้อมูลที่มีสัญญาณรบกวน (จากเครื่องสแกนเนอร์และการสแกนภาพ) พบว่า Progol ให้ความถูกต้องเพียง 72.00% ซึ่งสาเหตุที่ได้ความถูกต้องไม่สูงก็เนื่องมาจาก กฎที่ได้จากการเรียนรู้ไม่ยืดหยุ่นเพียงพอ และเมื่อนำเทคนิคการประมาณกฎ [Kijsirikul et al., 2001] มาปรับปรุงกฎที่ได้ ก็ช่วยให้ความถูกต้องที่ได้เพิ่มขึ้นสูงถึง 94.40%

อัลกอริทึมการเรียนรู้ของเครื่องสองชนิดที่มีประสิทธิภาพสูง และได้รับความสนใจอย่างมากในปัจจุบันคือ (1) ซัพพอร์ตเวกเตอร์แมชชีน – เอสวีเอ็ม (Support Vector Machines – SVMs) และ (2) การโปรแกรมตรรกะเชิงอุปนัย – ไอแอลพี (Inductive Logic Programming – ILP) ซัพพอร์ตเวกเตอร์แมชชีนเป็นการเรียนรู้ของเครื่องที่พัฒนาขึ้นโดย Vapnik [Vapnik, 1998] โดยมีแนวคิดหลักอยู่ที่การสร้างระนาบหลายมิติแบ่งแยกดีสุด (optimal separating hyperplane) เพื่อแบ่งตัวอย่างสอนในสองกลุ่มโดยให้แต่ละกลุ่มอยู่คนละด้านของระนาบหลายมิติ และมีระยะห่างจากข้อมูลสอนมายังระนาบหลายมิติกว้างที่สุด จากการศึกษาพบว่าเอสวีเอ็มมีประสิทธิภาพสูงมากและทนทานต่อข้อมูลมีสัญญาณรบกวนได้ดี แต่มีข้อจำกัดที่เอสวีเอ็มสามารถใช้ได้โดยตรงกับข้อมูลแค่สองกลุ่มเท่านั้น ถ้าต้องการนำไปประยุกต์ใช้กับข้อมูลหลายกลุ่มก็ต้องมีการแก้ไขปรับปรุง

การโปรแกรมตรรกะเชิงอุปนัยหรือไอแอลพี [Muggleton, 1991] มีจุดหมายเพื่อจำแนกตัวอย่างออกเป็นสองกลุ่มคือ กลุ่มตัวอย่างบวก (positive training data class) และกลุ่มตัวอย่างลบ (negative training data class) โดยการสร้างกฎเพื่ออธิบายกลุ่มตัวอย่างบวก ไอแอลพีมีจุดเด่นที่ผู้สอนสามารถให้ความรู้ภูมิหลัง (background knowledge) กับระบบเรียนรู้ ทำให้การเรียนรู้มีประสิทธิภาพสูงมาก และผลการเรียนรู้จะได้กฎตรรกะอันดับที่หนึ่ง (first-order logic rules) ซึ่งถือได้ว่าเป็นการแทนความรู้ที่มีประสิทธิภาพมาก อย่างไรก็ดีไอแอลพีก็ประสบปัญหา ในกรณีที่เราต้องการนำกฎที่สร้างได้ไปจำแนกกลุ่มของตัวอย่างใหม่ ซึ่งกฎจะจำแนกตัวอย่างที่ตรงกับกฎเป็นตัวอย่างบวก ส่วนตัวอย่างที่ไม่ตรงกับกฎจะถูกจำแนกเป็นตัวอย่างลบ ทำให้เมื่อเราต้องการนำระบบไอแอลพีไปใช้จำแนกตัวอย่างที่เป็นตัวอย่างแบบหลายกลุ่ม หรือกรณีที่ตัวอย่างบางตัวโดยเฉพาะตัวอย่างที่มีสัญญาณรบกวน (noisy data) ที่ไม่ตรงกับกฎข้อใดข้อหนึ่งในเซตของกฎทั้งหมด ระบบไอแอลพีแต่เพียงอย่างเดียวไม่สามารถจำแนกกลุ่มของตัวอย่างในลักษณะนี้ได้

งานวิจัยนี้จะศึกษาอัลกอริทึมการเรียนรู้ของเครื่อง โดยมุ่งเน้นที่จะวิจัยเอสวีเอ็มแบบหลายกลุ่ม และวิจัยการทำให้ไอแอลพีมีความทนทานต่อสัญญาณรบกวนและใช้ได้ในโดเมนหลายกลุ่มให้ได้อย่างมีประสิทธิภาพ ซึ่งผลที่คาดว่าจะได้คือองค์ความรู้ใหม่ของอัลกอริทึมเรียนรู้ที่มีประสิทธิภาพสูงใช้งานได้ในปัญหาจริง ซึ่งคาดว่าจะเป็นความก้าวหน้าในเชิงวิชาการของสาขาการเรียนรู้ของเครื่องเอง และก่อให้เกิดความก้าวหน้าในสาขาที่นำอัลกอริทึมเรียนรู้ไปใช้อื่นๆ เช่น การทำเหมืองข้อมูล ชีวสารสนเทศศาสตร์ เป็นต้น

วัตถุประสงค์

วัตถุประสงค์ของการวิจัยในโครงการนี้มีดังต่อไปนี้

(1) เพื่อพัฒนาอัลกอริทึมการเรียนรู้ของเครื่องให้สามารถใช้กับโดเมนหลายกลุ่มและโดเมนที่มีสัญญาณรบกวน

(2) เพื่อพัฒนาเอสวีเอ็มให้เป็นตัวจำแนกประเภทแบบหลายกลุ่ม

(3) เพื่อพัฒนาวิธีการที่ทำให้ไอแอลพีมีความยืดหยุ่นมากขึ้นและทนทานต่อสัญญาณรบกวน สามารถใช้ได้ในโดเมนหลายกลุ่ม

ระเบียบวิธีวิจัยและผลที่ได้

1. การวิจัยอัลกอริทึมเอสวีเอ็มแบบหลายกลุ่ม

ซัพพอร์ตเวกเตอร์แมชชีนพัฒนาขึ้นเพื่อจำแนกข้อมูลสองกลุ่มได้อย่างมีประสิทธิภาพในการใช้งานจริง อย่างไรก็ตามปัญหาการพัฒนาซัพพอร์ตเวกเตอร์แมชชีนให้สามารถจำแนกข้อมูลได้หลายกลุ่มยังคงอยู่ในขั้นตอนการทำวิจัย วิธีการจำแนกข้อมูลแบบหลายกลุ่มสำหรับซัพพอร์ตเวกเตอร์แมชชีน มักเป็นการนำฟังก์ชันที่จำแนกข้อมูลแบบสองกลุ่มมารวมกัน เช่น การจำแนกแบบหนึ่งต่อหนึ่ง (one-against-one) โดยการเปรียบเทียบกลุ่มแต่ละกลุ่มกับกลุ่มอื่นทีละกลุ่ม และการจำแนกแบบหนึ่งต่อที่เหลือ (one-against-the-rest) โดยการเปรียบเทียบกลุ่มแต่ละกลุ่มกับกลุ่มอื่นทั้งหมด เป็นต้น

อัลกอริทึมแมกซ์วิน (Max Wins) เป็นหนึ่งในวิธีการจำแนกแบบหนึ่งต่อหนึ่งใช้เวลาในการเรียนรู้เร็วกว่าเมื่อเปรียบเทียบกับวิธีหนึ่งต่อที่เหลือ [Friedman, 1996] โดยให้ตัวจำแนกแต่ละตัวโหวตค่ากลุ่มที่เป็นคำตอบ ส่วนผลการจำแนกคือกลุ่มที่มีค่าโหวตมากที่สุด วิธี*กราฟไม่มีวงแบบมีทิศทางสำหรับการตัดสินใจ – ดีดีเอจี (Decision Directed Acyclic Graph - DDAG)* ที่เสนอโดย Platt และคณะ [Platt et al., 1999] สามารถลดเวลาในการเรียนรู้และเวลาในการจำแนกได้ในขณะที่ให้ความถูกต้องเทียบได้กับวิธีแมกซ์วิน [Hsu & Lin, 2002] การทดลองเปรียบเทียบวิธีจำแนกข้อมูลแบบหลายกลุ่มวิธีต่างๆ ใน [Hsu & Lin, 2002] แสดงให้เห็นว่าอัลกอริทึมแมกซ์วินและวิธีดีดีเอจีเหมาะที่จะนำไปใช้งานจริง

ในงานวิจัยนี้เราแสดงให้เห็นว่าโครงสร้างกราฟของดีดีเอจีก่อให้เกิดปัญหาในแง่ของความถูกต้องของอัลกอริทึมอันเนื่องมาจากการใช้จำนวนครั้งในการเปรียบเทียบกลุ่มที่ถูกต้องกับกลุ่มอื่นๆ มากเกินไป และเนื่องจากเอสวีเอ็มแบบสองกลุ่มแต่ละตัวมีโอกาสจะจำแนกกลุ่มของข้อมูลผิด เพราะว่าเราไม่สามารถที่จะสร้างเอสวีเอ็มที่สมบูรณ์ที่มีความถูกต้องในการจำแนกกลุ่มของข้อมูล 100 เปอร์เซนต์ได้ ด้วยเหตุนี้จึงทำให้จำนวนครั้งของการเปรียบเทียบมีผลต่อความถูกต้องโดยรวมของดีดีเอจี เราจึงนำเสนอแนวคิดในการสร้างโครงสร้างกราฟแบบใหม่เพื่อลดจำนวนครั้งในการเปรียบเทียบ โดยการปรับเปลี่ยนโครงสร้างของกราฟของดีดีเอจี โครงสร้างใหม่ที่นำเสนอนี้เรียกว่า *กราฟไม่มีวงมีทิศทางแบบปรับได้ - เอดีเอจี (Adaptive Directed Acyclic Graph - ADAG)* ซึ่งลดจำนวนครั้งของการเปรียบเทียบลงได้ อย่างไรก็ตามความถูกต้องของการจำแนกข้อมูลของเอดีเอจียังคงขึ้นอยู่กับลำดับของโนด (เอสวีเอ็มแบบสองกลุ่ม) ที่อยู่ในโครงสร้างกราฟ ลำดับของโนดที่ต่างกันจะให้ความถูกต้องแตกต่างกันและนำไปสู่ความแปรปรวนของอัลกอริทึม เราจึงได้เสนอวิธีสำหรับเลือกลำดับที่เหมาะสมซึ่งจะทำให้โอกาสที่จะจำแนกประเภทข้อมูลผิดพลาดน้อยที่สุด โดยนำเสนอการจัดเรียงลำดับใหม่ในระหว่างกระบวนการจำแนกข้อมูลให้เป็นไปตามข้อมูลทดสอบแต่ละตัว และได้แสดงให้เห็นว่าปัญหาการเลือกลำดับที่เหมาะสมสามารถทำได้โดยอัลกอริทึมการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุด (minimum-weight perfect matching)

1.1 การจำแนกข้อมูลของซัพพอร์ตเวกเตอร์แมชชีน

ในหัวข้อนี้จะอธิบายถึงแนวคิดพื้นฐานของซัพพอร์ตเวกเตอร์แมชชีนและวิธีการแก้ปัญหาการจำแนกข้อมูลแบบหลายกลุ่ม

1) ซัพพอร์ตเวกเตอร์แมชชีนเชิงเส้น (Linear support vector machine)

แนวคิดหลักๆ ในการจำแนกข้อมูลของซัพพอร์ตเวกเตอร์แมชชีนคือ การสร้างระนาบหลายมิติที่ใช้แบ่งข้อมูลออกเป็นสองกลุ่ม สมมติมีเซตของข้อมูล D ที่ประกอบด้วยตัวอย่างจำนวน l ตัวในปริภูมิอันดับ n ที่มีสองกลุ่ม (+1 และ –1)

$$D = \left\{ \left(\mathbf{x}_k, y_k \right) \middle| \, k \in \{1,..,l\}, \mathbf{x}_k \in \Re^n, y_k \in \{+1,-1\} \right\} \tag{1}$$

ระนาบหลายมิติในปริภูมิอันดับ n ถูกกำหนดโดย $(\mathbf{w},b)$ เมื่อ $\mathbf{w}$ คือเวกเตอร์ในปริภูมิอันดับ n ที่ตั้งฉากกับระนาบหลายมิติ และ b คือค่าคงที่ ระนาบหลายมิติ $(\mathbf{w}\cdot\mathbf{x})+b$ จะแบ่งข้อมูลได้ก็ต่อเมื่อ

$$\begin{aligned} (\mathbf{w}\cdot\mathbf{x}_i)+b > 0 \quad &\text{if} \quad y_i = +1 \\ (\mathbf{w}\cdot\mathbf{x}_i)+b < 0 \quad &\text{if} \quad y_i = -1 \end{aligned} \tag{2}$$

ถ้าเราต้องการค่า $\mathbf{w}$ และ b ที่ทำให้จุดที่อยู่ใกล้กับระนาบหลายมิติมากที่สุดมีระยะห่าง $1/|\mathbf{w}|$ แล้ว จะได้

$$\begin{aligned} (\mathbf{w}\cdot\mathbf{x}_i)+b \geq 1 \quad &\text{if} \quad y_i = +1 \\ (\mathbf{w}\cdot\mathbf{x}_i)+b \leq -1 \quad &\text{if} \quad y_i = -1 \end{aligned} \tag{3}$$

ซึ่งเท่ากับ

$$y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1 \quad \forall i \tag{4}$$

ในการค้นหาระนาบหลายมิติแบ่งแยกดีสุด (optimal separating hyperplane) จะต้องค้นหาระนาบหลายมิติที่มีระยะห่างระหว่างข้อมูลที่ใช้สอนกับระนาบหลายมิติที่น้อยที่สุดมีค่ามากที่สุด ระยะห่างหรือมาร์จินระหว่างข้อมูลตัวอย่างสองตัวจากกลุ่มที่แตกต่างกันมีค่าเท่ากับ

$$d(\mathbf{w},b) = \min_{\{\mathbf{x}_i | y_i = 1\}} \frac{(\mathbf{w}\cdot\mathbf{x}_i)+b}{|\mathbf{w}|} - \max_{\{\mathbf{x}_i | y_i = -1\}} \frac{(\mathbf{w}\cdot\mathbf{x}_i)+b}{|\mathbf{w}|} \tag{5}$$

จากสมการที่ (3) ค่าที่น้อยที่สุดและมากที่สุดที่เหมาะสมคือ ± 1 ดังนั้นจะได้ว่าเราต้องค้นหาระนาบที่ทำให้ค่าของฟังก์ชัน

$$d(\mathbf{w},b) = \frac{1}{|\mathbf{w}|} - \frac{-1}{|\mathbf{w}|} = \frac{2}{|\mathbf{w}|} \tag{6}$$

สูงที่สุด ดังนั้นปัญหานี้จึงเท่ากับ

- ลดค่าของ $|\mathbf{w}|^2/2$ ให้ต่ำที่สุด
- โดยขึ้นกับเงื่อนไขต่อไปนี้

(1) $y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1 \quad \forall i$

สำหรับกรณีที่ไม่สามารถแบ่งข้อมูลด้วยระนาบหลายมิติโดยไม่มีข้อผิดพลาดเกิดขึ้นนั้น เราจำเป็นต้องปรับเงื่อนไขโดยเพิ่มพจน์ค่าปรับ (penalty term) ซึ่งประกอบด้วยผลรวมของความคลาดเคลื่อน ξ_i จากขอบเขตเข้าไปในเงื่อนไข ดังนั้นปัญหาตอนนี้คือ

- ลดค่าของ $\frac{|\mathbf{w}|^2}{2} + C\sum_{i=1}^{l}\xi_i$ ให้ต่ำที่สุด
- โดยขึ้นกับเงื่อนไขต่อไปนี้

  (1) $y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1-\xi_i$,

  (2) $\xi_i \geq 0 \ \ \forall i$

  นำลากรองเกียนมาใช้กับปัญหานี้ ดังนั้นสามารถแปลงปัญหาเป็น
- ลดค่าของฟังก์ชันด่านล่างนี้ให้ต่ำที่สุด

$$L(\mathbf{w},b,\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j) \tag{7}$$

- โดยขึ้นกับเงื่อนไขต่อไปนี้

  (1) $0 \leq \alpha_i \leq C$, $\forall i$

  (2) $\sum_{i=1}^{l}\alpha_i\, y_i = 0$

เมื่อ α_i เรียกว่าตัวคูณลากรอง ตัวอย่างข้อมูลสอนแต่ละตัวจะมีตัวคูณลากรองหนึ่งตัว ตัวอย่างข้อมูลสอนที่มีค่า $\alpha_i > 0$ เรียกว่าซัพพอร์ตเวกเตอร์ (support vector) และเป็นซัพพอร์ตเวกเตอร์ที่ทำให้ค่าทางขวามือของสมการที่ (4) เท่ากับ 1 ส่วนตัวอย่างข้อมูลสอนตัวอื่นๆ ที่มี $\alpha_i = 0$ สามารถตัดออกไปจากเซตของตัวอย่างข้อมูลสอนได้โดยไม่มีผลกระทบใดๆ ต่อผลลัพธ์ระนาบหลายมิติที่จะเป็นระนาบหลายมิติแบ่งแยกดีสุด

ให้ α^0 ซึ่งเป็นเวกเตอร์ในปริภูมิอันดับ l แทนค่าต่ำสุดของ $L(\mathbf{w},b,\alpha)$ ถ้า $\alpha_i^0 > 0$ แล้ว $\mathbf{x}_i$ คือซัพพอร์ตเวกเตอร์ ระนาบหลายมิติแบ่งแยกดีสุด $(\mathbf{w}^0, b^0)$ สามารถเขียนในพจน์ของ α^0 และข้อมูลสอน โดยเฉพาะอย่างยิ่งในพจน์ของซัพพอร์ตเวกเตอร์ได้ดังนี้

$$\mathbf{w}^0 = \sum_{i=1}^{l}\alpha_i^0 y_i\mathbf{x}_i = \sum_{\text{support vectors}}\alpha_i^0 y_i\mathbf{x}_i \tag{8}$$

$$b^0 = 1 - \mathbf{w}^0\cdot\mathbf{x}_i \text{ for } \mathbf{x}_i \text{ with } y_i = 1 \text{ and } 0 < \alpha_i < C \tag{9}$$

ระนาบหลายมิติแบ่งแยกดีสุดจะจำแนกจุดต่างๆ ตามเครื่องหมายของผลลัพธ์ของฟังก์ชัน $f(x)$

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}\left(\mathbf{w}^0\cdot\mathbf{x}+b^0\right) \\ &= \text{sign}\left(\sum_{\text{suport vec tors}}\alpha_i^0\, y_i(\mathbf{x}_i\cdot\mathbf{x}) + b^0\right) \end{aligned} \tag{10}$$

2) ซัพพอร์ตเวกเตอร์แมชชีนไม่เชิงเส้น (Non-linear support vector machine)

อัลกอริทึมที่ได้กล่าวมาแล้วใช้ได้กับข้อมูลที่สามารถแบ่งด้วยระนาบหลายมิติเชิงเส้นได้เท่านั้น ซัพพอร์ตเวกเตอร์แมชชีนแก้ปัญหากรณีที่ข้อมูลแบ่งไม่ได้ด้วยระนาบเชิงเส้นโดยแมปข้อมูลตัวอย่างไปยังปริภูมิอันดับสูง (higher dimensional space) โดยเลือกใช้ฟังก์ชันการแมปที่ไม่เป็นเชิงเส้น นั่นคือ เราเลือกฟังก์ชันในการแมป $\Phi : \Re^n \mapsto \mathrm{H}$ เมื่อปริภูมิของ $\mathbf{H}$ มีอันดับสูงกว่าปริภูมิอันดับ n เราสามารถค้นหาระนาบหลายมิติแบ่งแยกดีสุดในปริภูมิอันดับสูงที่เทียบเท่ากับการแยกที่ไม่เป็นเชิงเส้นใน $\Re^n$

ข้อมูลสอนที่ใช้ในการเรียนรู้ในสมการที่ (7) อยู่ในรูปของผลคูณเชิงสเกลาร์ระหว่างเวกเตอร์ ดังนั้นในปริภูมิอันดับสูงเราสามารถจัดการกับข้อมูลในรูปของ $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$ ถ้าปริภูมิของ $\mathbf{H}$ มีอันดับสูงจะทำให้ยุ่งยากหรือใช้การคำนวณมาก อย่างไรก็ตามถ้าเรามีฟังก์ชันเคอร์เนลที่สามารถคำนวณ $k(\mathbf{x}_i,\mathbf{x}_j)=\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$ แล้วเรา

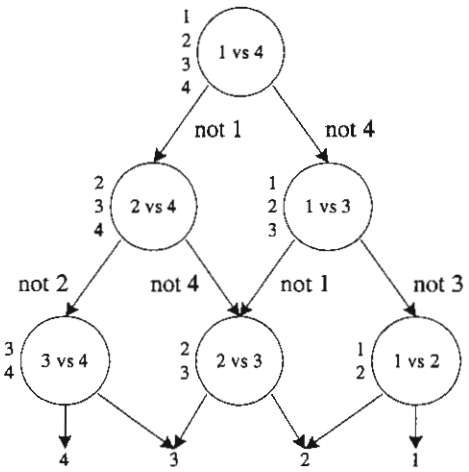
สามารถใช้ฟังก์ชันนี้แทนที่ $\mathbf{x}_i \cdot \mathbf{x}_j$ ทุกๆ ที่ในการคำนวณ และไม่จำเป็นต้องรู้ฟังก์ชันที่ใช้แมป Φ จริงๆ ฟังก์ชันเคอร์เนลที่นิยมใช้ ได้แก่

ฟังก์ชันพหุนาม (Polynomial degree d): $k(\mathbf{x},\mathbf{y}) = \left|\mathbf{x}\cdot\mathbf{y}+1\right|^d$ (11)

ฟังก์ชันอาร์บีเอฟ (Radial basis function): $k(\mathbf{x},\mathbf{y}) = e^{-\left|\mathbf{x}-\mathbf{y}\right|^2/c}$ (12)

1.2 ดีดีเอจี

Platt และคณะ [Platt et al., 1999] ได้เสนอวิธีดีดีเอจี (Decision Directed Acyclic Graph - DDAG) ซึ่งนำตัวจำแนกแบบสองกลุ่มหลายตัวมารวมกันเป็นตัวจำแนกแบบหลายกลุ่มหนึ่งตัว สำหรับปัญหาที่มี k กลุ่ม ในขั้นตอนการเรียนรู้จะสร้างตัวจำแนกแบบสองกลุ่มจำนวน $k(k-1)/2$ ตัวเช่นเดียวกับวิธีการจำแนกแบบหนึ่งต่อหนึ่ง ตัวจำแนกแต่ละตัวใช้สำหรับคู่ของกลุ่มแต่ละคู่ อย่างไรก็ตามในขั้นตอนการจำแนก จะใช้กราฟไม่มีวงแบบมีทิศทาง แต่ละโนดจะมีฟังก์ชันแบบทวิภาค ซึ่งมีโนดทั้งหมด $k(k-1)/2$ โนดและมี k ใบ (ดูรูปที่ 1) แต่ละโนดคือตัวจำแนกข้อมูลของกลุ่มที่ i และ j เมื่อต้องการจำแนกข้อมูล $\mathbf{x}$ ข้อมูลเข้าจะเริ่มจากโนดราก แล้วฟังก์ชันแบบทวิภาคที่โนดจะถูกนำมาคำนวณ ถ้าได้ค่าของฟังก์ชันน้อยกว่าศูนย์ก็ออกที่ขอบซ้าย ถ้าได้ค่ามากกว่าหรือเท่ากับศูนย์ก็ออกที่ขอบขวา หลังจากนั้นฟังก์ชันทวิภาคที่โนดต่อมาจะถูกคำนวณ จนถึงโนดใบซึ่งจะบ่งบอกถึงกลุ่มของข้อมูลที่ดีดีเอจีทำนาย



รูปที่ 1 โครงสร้างของดีดีเอจีสำหรับปัญหา 4 กลุ่ม

ปัญหาที่สำคัญของดีดีเอจีคือในการเปรียบเทียบหากลุ่มถูกต้องดีดีเอจีใช้จำนวนครั้งในการเปรียบเทียบมากเกินไป ซึ่งทำให้เกิดความผิดพลาดสะสมสูงและส่งผลให้กลุ่มที่ถูกต้องถูกกำจัดออกอย่างผิดพลาดได้ เส้นทางการจำแนกของดีดีเอจีมีความลึกเท่ากับ $k-1$ นั่นคือจำนวนครั้งที่กลุ่มที่ถูกต้องถูกจำแนกกับกลุ่มอื่นจะแปรตามจำนวนกลุ่มของข้อมูล

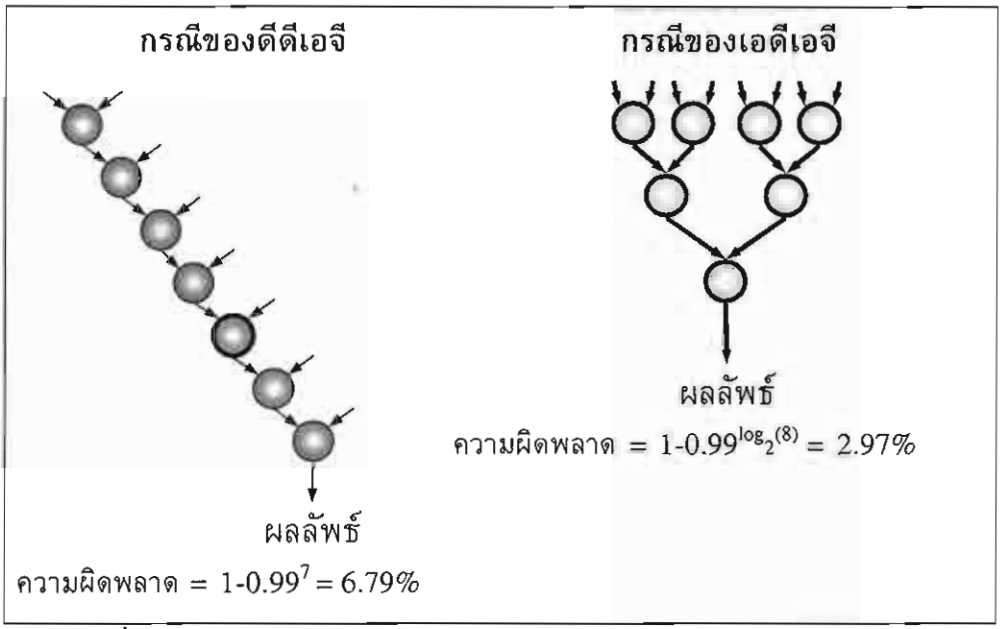
1.3 วิธีการที่นำเสนอ

หัวข้อนี้นำเสนอวิธีการปรับปรุงดีดีเอจีโดยการปรับเปลี่ยนโครงสร้างให้ลดจำนวนครั้งในการเปรียบเทียบเพื่อหากลุ่มที่ถูกต้อง เราเรียกโครงสร้างใหม่ที่ได้ว่า *กราฟไม่มีวงมีทิศทางแบบปรับได้ – ดีดีเอจี (Adaptive Directed Acyclic Graph – ADAG)* ต่อจากนั้นจะนำเสนอวิธีการกำหนดลำดับของตัวแปรในโครงสร้างกราฟซึ่งเรียกวิธีการที่ได้ว่า *กราฟไม่มีวงมีทิศทางและปรับได้แบบจัดเรียงใหม่ - อาร์เอดีเอจี (Reordering Adaptive Directed*

Acyclic Graph – RADAG) นอกจากนี้เรายังได้พัฒนาเอสวีเอ็มแบบหลายกลุ่มให้สามารถใช้จำนวนครั้งในการเปรียบเทียบเหลือเพียงประมาณ $\log_2 k$ ได้ ซึ่งเรียกวิธีนี้ว่า *การแตกครึ่งตามสารสนเทศ* (*Information-Based Dichotomization*) ดังมีรายละเอียดต่อไปนี้

1.3.1 เอดีเอจี

รูปที่ 2 ด้านล่างนี้ (ด้านซ้าย) แสดงให้เห็นถึงค่าความผิดพลาดของดีดีเอจีในการจำแนกข้อมูลในปัญหา 8 กลุ่ม โดยสมมติว่าเอสวีเอ็มแบบสองกลุ่มแต่ละตัว (แทนด้วยโนดแต่ละโนดในรูป) มีความผิดพลาด 1% ถ้ากลุ่มที่ถูกต้องถูกเปรียบเทียบที่โนดแรกของดีดีเอจี จำนวนครั้งในการเปรียบเทียบกลุ่มที่ถูกต้องกับกลุ่มอื่นจะเท่ากับ 7 ครั้ง ซึ่งเราจะคำนวณความผิดพลาดสะสมในกรณีนี้ได้เท่ากับ 6.79%



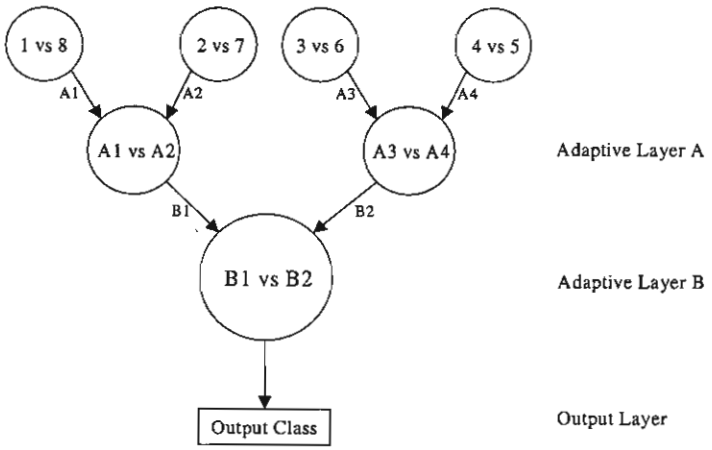
รูปที่ 2 เปรียบเทียบความถูกต้องของดีดีเอจีกับกราฟจากแนวคิดใหม่ในปัญหา 8 กลุ่ม

แนวคิดในการสร้างวิธีการใหม่ก็คือการลดจำนวนครั้งในการเปรียบเทียบ ซึ่งสามารถทำได้โดยเปลี่ยนโครงสร้างของกราฟของดีดีเอจี โดยเริ่มต้นจากระดับแรกของกราฟประกอบด้วยจำนวนบัพเท่ากับ $\lceil k/2 \rceil$[1] ตัว เมื่อ k คือจำนวนกลุ่ม จำนวนโนดลดลงทีละครึ่งในแต่ละระดับ แต่ละโนดจะเลือกกลุ่มที่ตนต้องการส่งต่อไปยังระดับถัดไป ด้วยวิธีการเช่นนี้จะทำให้จำนวนครั้งในการเปรียบเทียบกลุ่มที่ถูกต้องกับกลุ่มอื่นลดลง ซึ่งจะส่งผลให้ความผิดพลาดสะสมลดลง เราสามารถคำนวณจำนวนครั้งที่ใช้เปรียบเทียบกลุ่มที่ถูกต้องกับกลุ่มอื่นได้เท่ากับ $\lceil \log_2 k \rceil$ ในกรณีของปัญหา 8 กลุ่มในตัวอย่างรูปที่ 2 จะได้ว่าจำนวนครั้งที่ใช้เปรียบเทียบกลุ่มที่ถูกต้องเท่ากับ 3 (น้อยกว่ากรณีของดีดีเอจีที่เท่ากับ 7) และจะได้ว่าความผิดพลาดสะสมเท่ากับ 2.97% ซึ่งน้อยกว่ามากเมื่อเทียบกับกรณีของดีดีเอจี (6.79%)

ด้วยแนวคิดนี้เราเรียกวิธีการใหม่ที่เสนอนี้ว่าเอดีเอจี (Adaptive Directed Acyclic Graph - ADAG) ซึ่งเป็นกราฟไม่มีวงแบบมีทิศทางที่ปรับได้ มีโครงสร้างคล้ายสามเหลี่ยมคว่ำ สำหรับปัญหาที่มี k กลุ่ม จะสร้างตัวจำแนกแบบสองกลุ่มจำนวน $k(k-1)/2$ ตัว แต่ละตัวเป็นฟังก์ชันแบบทวิภาค โดยจะมีเพียง $k-1$ ตัวที่เอดีเอจีเลือกมาใช้ในการจำแนกข้อมูล ในขั้นตอนการจำแนก โนดของเอดีเอจีจะถูกจัดเรียงเป็นรูปสามเหลี่ยมคว่ำ โดยมี

[1] โดยที่ $\lceil x \rceil$ แสดงเลขจำนวนเต็มที่น้อยที่สุดที่มากกว่าหรือเท่ากับ x

โนดจำนวน $k/2$ โนด (ปัดขึ้น) ที่ระดับบนสุด และจำนวนโนดจะลดลงครึ่งหนึ่งสำหรับระดับถัดมา จนเหลือโนดเดียวในระดับล่างสุด (ดูรูปที่ 3) เมื่อต้องการจำแนกข้อมูล $\mathbf{x}$ เริ่มจากระดับบนสุด โนดแต่ละโนดจะกำจัดกลุ่มหนึ่งกลุ่มออกไปและส่งข้อความที่บ่งถึงกลุ่มที่เลือกมายังโนดระดับล่าง ในแต่ละรอบจำนวนกลุ่มที่มีโอกาสถูกเลือกเป็นกลุ่มที่ถูกต้องจะลดลงครึ่งหนึ่ง การเลือกตัวจำแนกแบบสองกลุ่มในโนดแต่ละโนดของระดับถัดมา จะเลือกจากข้อความที่ส่งมาจากโนดระดับบนสองโนด กระบวนการนี้จะดำเนินต่อไปจนเหลือโนดเดียวในระดับล่างสุด ผลการจำแนกจะมาจากข้อความที่ส่งมาจากโนดล่างสุด วิธีเอดีเอจีใช้โนดทั้งหมด $k-1$ โนดในการจำแนก เช่นเดียวกับวิธีดีดีเอจี กลุ่มที่ถูกต้องจะถูกจำแนกทั้งหมดอย่างมาก $log_2 k$ ครั้ง ซึ่งต่ำกว่าจำนวนที่ต้องการในโครงสร้างดีดีเอจี ซึ่งจำนวนครั้งจะแปรตามค่า k



รูปที่ 3 โครงสร้างของเอดีเอจีสำหรับปัญหา 8 กลุ่ม

จากการใช้โครงสร้างแบบสามเหลี่ยมคว่ำ วิธีเอดีเอจีสามารถลดจำนวนครั้งที่กลุ่มที่ถูกต้องถูกจำแนกกับกลุ่มอื่น จึงลดความผิดพลาดสะสมลงได้ อย่างไรก็ตามความถูกต้องของวิธีนี้ยังคงขึ้นอยู่กับลำดับของโนด ในหัวข้อถัดไปจะกล่าวถึงวิธีปรับปรุงประสิทธิภาพของเอดีเอจี โดยการเลือกลำดับของโนดที่เหมาะสมซึ่งมีโอกาสจำแนกผิดพลาดน้อย

1.3.2 อาร์เอดีเอจี

ในหัวข้อนี้จะนำเสนอการปรับปรุงวิธีเอดีเอจีเพื่อเพิ่มประสิทธิภาพให้สูงขึ้น วิธีการนี้จะเลือกลำดับของโนดในโครงสร้างเอดีเอจีที่เหมาะสม โดยพิจารณาจากค่าขอบเขตของความผิดพลาดของซัพพอร์ตเวกเตอร์ มีการจัดเรียงลำดับของโนดใหม่ในระหว่างกระบวนการจำแนกข้อมูลให้เป็นไปตามข้อมูลทดสอบแต่ละตัว เราเรียกวิธีการใหม่นี้ว่าอาร์เอดีเอจี

ประสิทธิภาพโดยนัยทั่วไปของซัพพอร์ตเวกเตอร์แมชชีน

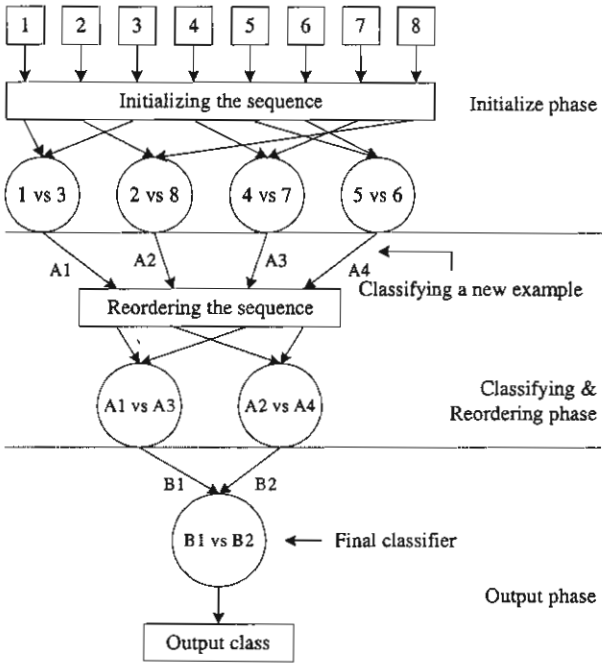
ประสิทธิภาพโดยนัยทั่วไป (generalization performance) ของซัพพอร์ตเวกเตอร์แมชชีนสามารถประมาณได้โดยกำหนดขอบเขตของความผิดพลาดของซัพพอร์ตเวกเตอร์แมชชีน [Bartlett et al., 1999] เรากำหนดชนิดของฟังก์ชันเป็นค่าตัวเลขจำนวนจริงภายในลูกบอลที่มีรัศมี R ค่ารัศมีไม่เกิน $\Re^n$ เป็นฟังก์ชันดังนี้ $F = \{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} : \|\mathbf{w}\| \le 1, \|\mathbf{x}\| \le R\}$ จะมีค่าคงที่ c ที่สำหรับการกระจายทุกประเภท ด้วยความน่าจะเป็นอย่างน้อย $1-\delta$ บนตัวอย่าง $\mathbf{z}$ ที่เลือกมาอย่างอิสระจากกัน m ตัว ถ้าตัวจำแนก $h = \mathrm{sgn}(f) \in \mathrm{sgn}(F)$ มีมาร์จินอย่างน้อย γ สำหรับตัวอย่างทุกตัวใน $\mathbf{z}$ แล้ว ความผิดพลาดของตัวจำแนก h จะไม่เกิน

$$\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m + \log\left(\frac{1}{\delta}\right)\right) \qquad (13)$$

นอกเหนือจากนี้แล้ว ด้วยความน่าจะเป็นอย่างน้อย 1-δ ทุกตัวจำแนก $h \in \mathrm{sgn}(F)$ จะมีความผิดพลาดไม่เกิน

$$\frac{k}{m} + \sqrt{\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m + \log\left(\frac{1}{\gamma}\right)\right)} \qquad (14)$$

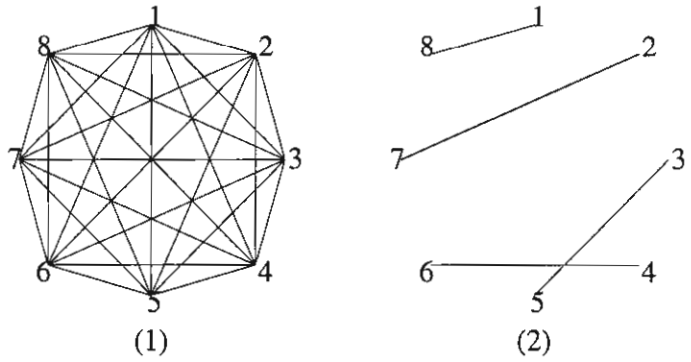
เมื่อ k คือจำนวนตัวอย่างข้อมูลใน **z** ที่มีมาร์จินน้อยกว่า γ



รูปที่ 4 การจำแนกข้อมูลของอาร์เอดีเอจี

อัลกอริทึมอาร์เอดีเอจี

ส่วนนี้อธิบายอัลกอริทึมอาร์เอดีเอจีเพื่อปรับปรุงความถูกต้องของการจำแนกของวิธีเอดีเอจีเดิม สำหรับปัญหาที่มี k กลุ่ม ในขั้นตอนการเรียนรู้ของอาร์เอดีเอจีจะเหมือนกับเอดีเอจี คือสร้างตัวจำแนกแบบสองกลุ่มจำนวน $k(k-1)/2$ ตัว แต่ละตัวเป็นฟังก์ชันแบบทวิภาค ในขั้นตอนการจำแนกจะมีกระบวนการคล้ายกันกับของเอดีเอจี แต่มีส่วนที่แตกต่างจากเอดีเอจีคือ ลำดับเริ่มต้นของตัวจำแนกในโนดในระดับบนสุดและลำดับของตัวจำแนกในโนดในระดับล่าง (ดูรูปที่ 4) ขั้นตอนแรกเราจะใช้อัลกอริทึมจัดเรียงใหม่ (reordering algorithm) ที่ทำงานร่วมกับอัลกอริทึมการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุดในการเลือกลำดับของตัวจำแนก ซึ่งจะอธิบายในหัวข้อต่อไป และจะใช้ลำดับนี้เป็นลำดับเริ่มต้นในการจำแนกข้อมูลทดสอบทุกตัว ขั้นตอนที่สองจะเหมือนกับของเอดีเอจี กล่าวคือแต่ละโนดจะกำจัดกลุ่มหนึ่งออกไปและส่งข้อความที่บ่งถึงกลุ่มที่เลือกมายังโนดระดับล่าง ขั้นตอนที่สามจะแตกต่างจากเอดีเอจี อาร์เอดีเอจีจะจัดเรียงลำดับของตัวจำแนกในโนดใหม่ก่อนจะประมวลผลในระดับถัดไป โดยใช้อัลกอริทึมจัดเรียงใหม่ที่ทำงานร่วมกับอัลกอริทึมการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุด ซึ่งลำดับของตัวจำแนกในแต่ละระดับสำหรับแต่ละตัวอย่างทดสอบจะแตกต่างกัน และขึ้นอยู่กับผลลัพธ์ของโนดในระดับบน ขั้นตอนที่สองและสามจะทำซ้ำจนกระทั่งเหลือโนดเดียวในระดับล่างสุด ผลการจำแนกจะมาจากข้อความที่ส่งมาจากโนดล่างสุด



รูปที่ 5 (1) กราฟสำหรับปัญหา 8 กลุ่ม
(2) ตัวอย่างของผลลัพธ์จากอัลกอริทึมจัดเรียงใหม่

อัลกอริทึมจัดเรียงใหม่ (Reordering algorithm)

จากขอบเขตของความผิดพลาดของซัพพอร์ตเวกเตอร์แมชชีนที่ได้กล่าวไปแล้ว เราจะพิจารณาขอบเขตของความผิดพลาดของตัวจำแนกแบบสองกลุ่มแต่ละตัว เพื่อเลือกตัวจำแนกที่มีขอบเขตของความผิดพลาดต่ำ $k/2$ ตัวไปใช้ในการจำแนกข้อมูล

ให้ $G = (V,E)$ แทนกราฟที่มีเซตของโนด V และเซตของเส้นเชื่อม E แต่ละโนดใน G แทนหนึ่งกลุ่มของข้อมูลและแต่ละเส้นเชื่อมแทนตัวจำแนกแบบสองกลุ่มซึ่งมีค่าน้ำหนักเป็นค่าขอบเขตของความผิดพลาดตามสมการที่ (14) (ดูรูปที่ 5(1)) ผลลัพธ์ของอัลกอริทึมจัดเรียงใหม่สำหรับกราฟ G คือเซตย่อยของเส้นเชื่อมที่มีผลรวมของค่าขอบเขตของความผิดพลาดของทุกเส้นต่ำที่สุด และแต่ละโนดใน G จะถูกเชื่อมด้วยเส้นเชื่อมที่อยู่ในเซตย่อยเพียงหนึ่งเส้นเท่านั้น (ดูรูปที่ 5(2)) กำหนดค่าน้ำหนักของเส้นเชื่อม c_e เท่ากับค่าขอบเขตความผิดพลาดของตัวจำแนกแบบสองกลุ่มซึ่งแทนด้วยเส้นเชื่อม e ของกราฟ G ปัญหาของอัลกอริทึมจัดเรียงใหม่สามารถแก้ปัญหาได้โดยใช้อัลกอริทึมการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุด ซึ่งจะหาค่าการจับคู่สมบูรณ์แบบ M ที่มีค่าน้ำหนักของเส้นเชื่อม $\Sigma(c_e : e \in M)$ ต่ำที่สุด

สำหรับ $U \subseteq V$ ให้ $E(U) = \{(i,j):(i,j)\in E, i\in U, j\in U\}$ โดย $E(U)$ คือเซตของตัวจำแนกที่ถูกเลือก ให้ $\delta(i)$ แทนเซตของเส้นเชื่อมที่เชื่อมกับโนด i หรือเซตของตัวจำแนกที่มีหนึ่งกลุ่มตรงกับกลุ่ม i แล้วการจับคู่สมบูรณ์แบบของกราฟ $G = (V,E)$ ที่มี $|V|$ เป็นเลขคู่หาได้โดย

$$(1)\quad x \in R_+^m \tag{15}$$

$$(2)\quad \sum_{e\in\delta(v)} x_e = 1 \quad \text{for } v \in V \tag{16}$$

$$(3)\quad \sum_{e\in E(U)} x_e \le \left\lfloor \frac{|U|}{2} \right\rfloor \text{ for all odd sets } U \subseteq V \text{ with } |U| \ge 3 \tag{17}$$

หรือโดยเงื่อนไข (1), (2) และ

$$(4)\quad \sum_{e\in\delta(U)} x_e \ge 1 \text{ for all odd sets } U \subseteq V \text{ with } |U| \ge 3 \tag{18}$$

เมื่อ $|E| = m$ โดย m คือจำนวนตัวจำแนกแบบสองกลุ่ม และ $x_e = 1$ หมายถึงตัวจำแนก e ถูกเลือกนำไปใส่ในลำดับ

ดังนั้นปัญหาของอัลกอริทึมจัดเรียงใหม่คือการแก้ปัญหาของโปรแกรมเชิงเส้นดังนี้

$$\min \sum_{e \in E} c_e x_e \qquad (19)$$

เมื่อ x เป็นไปตามเงื่อนไข "(1), (2) และ (3)" หรือ "(1), (2) และ (4)"

ปัจจุบันอัลกอริทึมการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุดสามารถหาคำตอบได้ในขอบเขตเวลา $O(n(m+n \log n))$ เมื่อ n คือจำนวนโนดในกราฟ (จำนวนกลุ่มของข้อมูล) และ m คือจำนวนเส้นเชื่อม (จำนวนตัวจำแนกข้อมูลแบบสองกลุ่ม) [Cook & Rohe, 1997] ดังนั้นอัลกอริทึมจัดเรียงใหม่สามารถจัดเรียงลำดับของตัวจำแนกได้ในเวลาพหุนาม (polynomial time)

ผลการทดลอง

ในหัวข้อนี้จะแสดงผลการทดลอง โดยใช้ชุดข้อมูลจำนวน 8 ชุดจาก the UCI Repository of Machine Learning Databases [Bartlett et al., 1999] ประกอบด้วยชุดข้อมูล Glass, Satimage, Segment, Shuttle, Vowel, Soybean, Letter และ Isolet (ดูตารางที่ 1) ชุดข้อมูลเหล่านี้มีจำนวนกลุ่มและขนาดของข้อมูลแตกต่างกันไป สำหรับชุดข้อมูล Glass และ Segment ไม่มีข้อมูลทดสอบ ดังนั้นจึงใช้วิธี 5-fold cross validation [Dietterich, 1997] ในการทดลอง สำหรับชุดข้อมูล Soybean จะตัดกลุ่มข้อมูล 4 กลุ่มสุดท้ายออกไปเนื่องจากมีข้อมูลบางส่วนไม่ครบ

ตารางที่ 1 รายละเอียดของชุดข้อมูลที่ใช้ในการทดลอง

ชุดข้อมูล	ข้อมูลสอน	ข้อมูลทดสอบ	กลุ่ม	จำนวนคุณลักษณะ
Glass	214	5-fold	6	9
Satimage	4,435	2,000	6	36
Segment	2,310	5-fold	7	18
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35
Letter	15,963	4,037	26	16
Isolet	6,238	1,559	26	617

ในการทดลองนี้จะปรับข้อมูลทั้งข้อมูลสอนและข้อมูลทดสอบให้อยู่ในช่วง [-1,1] และใช้เคอร์เนลแบบพหุนาม (Polynomial kernel) และแบบอาร์บีเอฟ (RBF kernel) แล้วเปรียบเทียบผลการทดลองของวิธีดีดีเอจี เอดีเอจี อาร์เอดีเอจีและแมกซ์วิน สำหรับวิธีดีดีเอจีและเอดีเอจีจะทำการทดลองกับทุกลำดับของตัวจำแนกแบบสองกลุ่มที่เป็นไปได้สำหรับชุดข้อมูลที่มีจำนวนกลุ่มไม่เกิน 7 กลุ่ม และจะทำการสุ่มลำดับของตัวจำแนกแบบสองกลุ่ม 50,000 ลำดับสำหรับชุดข้อมูลที่มีจำนวนกลุ่มมากกว่า 7 กลุ่ม ตารางที่ 2 และตารางที่ 3 เปรียบเทียบความถูกต้องระหว่างอัลกอริทึมทั้งสามสำหรับเคอร์เนลแบบพหุนามและแบบอาร์บีเอฟตามลำดับ

จากผลการทดลองพบว่าโดยส่วนใหญ่วิธีเอดีเอจีให้ความถูกต้องสูงกว่าของวิธีดีดีเอจี ซึ่งแสดงให้เห็นว่าโครงสร้างกราฟของเอดีเอจีช่วยให้ความถูกต้องในการจำแนกข้อมูลสูงขึ้น และพบว่าวิธีอาร์เอดีเอจีให้ความถูกต้องในการจำแนกข้อมูลสูงกว่าวิธีอื่นๆ ทั้งหมดที่นำมาทดลองซึ่งแสดงให้เห็นถึงประสิทธิภาพของวิธีอาร์เอดีเอจี นอกจากนั้นเรายังพบว่าในชุดข้อมูลส่วนใหญ่เมื่อใช้เคอร์เนลแบบอาร์บีเอฟจะให้ความถูกต้องในการจำแนกข้อมูลสูงกว่าเมื่อใช้เคอร์เนลแบบพหุนาม

ในการทดลองการวัดความเร็วของอัลกอริทึม เราพบว่าวิธีอาร์เอดีเอจีใช้เวลาในการประมวลผลน้อยกว่าแมกซ์วินประมาณ 63.75% โดยเฉลี่ย ในที่นี้จะขอยกตัวอย่างชุดข้อมูล Letter ที่ใช้เคอร์เนลแบบพหุนามดีกรี 4

ในการเรียนรู้ เมื่อประมวลผลด้วยโปรเซสเซอร์ Pentium II 400 MHz อัลกอริทึมแมกซ์วินใช้เวลาในการจำแนก *ข้อมูลทดสอบ 125.58 วินาที ส่วนวิธีอาร์เอดีเอจีใช้เวลาในการจำแนกและจัดเรียงลำดับรวมทั้งหมดเพียง 12.68* วินาที

ตารางที่ 2 เปรียบเทียบความถูกต้องเมื่อใช้เคอร์เนลแบบพหุนาม

ชุดข้อมูล	ดีดีเอจี	เอดีเอจี	แมกซ์วิน	อาร์เอดีเอจี
Glass	71.069	**71.135**	71.078	71.063
Satimage	89.599	89.622	89.615	**89.681**
Segment	97.360	97.383	97.351	**97.533**
Shuttle	99.919	99.922	99.923	**99.930**
Vowel	98.872	98.894	98.901	**98.990**
Soybean	92.202	92.281	92.470	**92.698**
Letter	95.994	96.379	96.512	**96.674**
Isolet	97.484	97.485	97.488	**97.499**

ตารางที่ 3 เปรียบเทียบความถูกต้องเมื่อใช้เคอร์เนลแบบอาร์บีเอฟ

ชุดข้อมูล	ดีดีเอจี	เอดีเอจี	แมกซ์วิน	อาร์เอดีเอจี
Glass	72.850	72.759	73.238	**74.319**
Satimage	92.129	92.141	92.148	**92.152**
Segment	97.652	97.650	**97.656**	97.576
Shuttle	99.926	99.927	99.928	**99.931**
Vowel	98.965	98.975	98.980	**99.091**
Soybean	91.739	92.570	92.533	**93.016**
Letter	95.994	96.379	96.512	**96.634**
Isolet	97.517	97.523	97.527	**97.589**

1.3.3 การประยุกต์ใช้ทฤษฎีสารสนเทศกับซัพพอร์ตเวกเตอร์แมชชีน

เทคนิคที่จะนำเสนอในหัวข้อนี้มีชื่อว่า *การแตกครึ่งตามสารสนเทศ (Information Based Dichotomization – IBD)* ซึ่งใช้ทฤษฎีสารสนเทศเพื่อสร้างซัพพอร์ตเวกเตอร์แมชชีนแบบหลายกลุ่ม ดังมีรายละเอียดต่อไปนี้

ทฤษฎีสารสนเทศ (Information Theory)

ค่าสารสนเทศของข้อมูลขึ้นอยู่กับความน่าจะเป็นของข้อมูล ซึ่งสามารถวัดอยู่ในรูปของบิตจากสูตร

$$\text{ค่าสารสนเทศของข้อมูล} = -\log_2 P \qquad (20)$$

โดยที่ P คือความน่าจะเป็นของข้อมูล

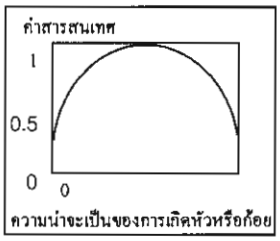
ถ้าให้ชุดของข้อมูล M ประกอบด้วยค่าที่เป็นไปได้คือ $\{m_1, m_2,\ldots,m_n\}$ และให้ความน่าจะเป็นที่จะเกิดค่า m_i มีค่าเท่ากับ $P(m_i)$ จะได้ว่าค่าสารสนเทศของ M หรือค่าเอนโทรปีของ M เขียนแทนด้วย $I(M)$ คำนวณได้จากสูตร

$$I(M) = \sum_{i=1}^{n} -P(m_i)\log_2 P(m_i) \quad (21)$$

กรณีที่ค่าสารสนเทศมีค่าน้อยก็จะหมายถึงข้อมูลในชุดนั้นมีความแตกต่างกันน้อย หรือเกือบจะเป็นกลุ่มเดียวกันทั้งหมด แต่กรณีค่าสารสนเทศสูงจะบ่งบอกว่าข้อมูลในชุดนั้นมีความแตกต่างกันมากหรือประกอบด้วยตัวอย่างหลายกลุ่มที่มีจำนวนใกล้เคียงกัน ตัวอย่างในการโยนหัวโยนก้อย ชุดข้อมูล M จะประกอบด้วยค่าที่เป็นไปได้ {หัว, ก้อย} จะได้ว่าค่าสารสนเทศของการโยนหัวก้อยเท่ากับ

$$-P(\text{หัว})\ log_2(P(\text{หัว})) - P(\text{ก้อย})\ log_2(P(\text{ก้อย})) \quad (22)$$

เมื่อพิจารณากรณีที่โยนออกหัวหมดหรือก้อยหมด ค่าสารสนเทศจะเป็น 0 และค่าสารสนเทศจะค่อยๆ เพิ่มขึ้นจนสูงที่สุด เมื่อความน่าจะเป็นของการเกิดหัวเท่ากับความน่าจะเป็นของการเกิดก้อยดังแสดงในรูปที่ 6



รูปที่ 6 ความสัมพันธ์ระหว่างความน่าจะเป็นและค่าสารสนเทศ

เทคนิคการจำแนกข้อมูลแบบแตกครึ่งตามสารสนเทศ

เทคนิคที่จะนำเสนอนี้จะทำให้การจำแนกข้อมูลแต่ละครั้งสามารถตัดกลุ่มข้อมูลที่ไม่ถูกต้องออกไปได้มากที่สุด โดยพิจารณาความน่าจะเป็นในการเกิดข้อมูลแต่ละกลุ่มประกอบเพื่อให้ได้จำนวนครั้งเฉลี่ยของการจำแนกลดลง

กรณีตัวอย่างจากปัญหาการเข้ารหัสข้อมูลของอักขระ 4 ตัวคือ A, B, C และ D โดยสมมติให้ความน่าจะเป็นของการเกิดข้อมูลแต่ละกลุ่มมีเท่ากัน จะได้ว่าจำนวนบิตที่น้อยที่สุดในการแทนอักขระเหล่านี้คือ 2 บิตโดยแทนแต่ละอักขระดังนี้

A	แทนด้วย	00
B	แทนด้วย	01
C	แทนด้วย	10
D	แทนด้วย	11

ตัวอย่างเช่น หากต้องการเข้ารหัสข้อมูลของ ABACADAB จะได้ 0001001000110001 จำนวนบิตที่ใช้คือ 16 เมื่อพิจารณาข้อมูลจากตัวอย่าง ตลอดจนข้อมูลที่เป็นข้อความทั่วไปในชีวิตประจำวันจะมีความน่าจะเป็นในการเกิดข้อมูลของแต่ละอักขระไม่เท่ากัน เช่น อักขระ A, E, I, O และ U ซึ่งเป็นสระในภาษาอังกฤษจะมีความน่าจะเป็นในการเกิดมากกว่าอักขระอื่นๆ

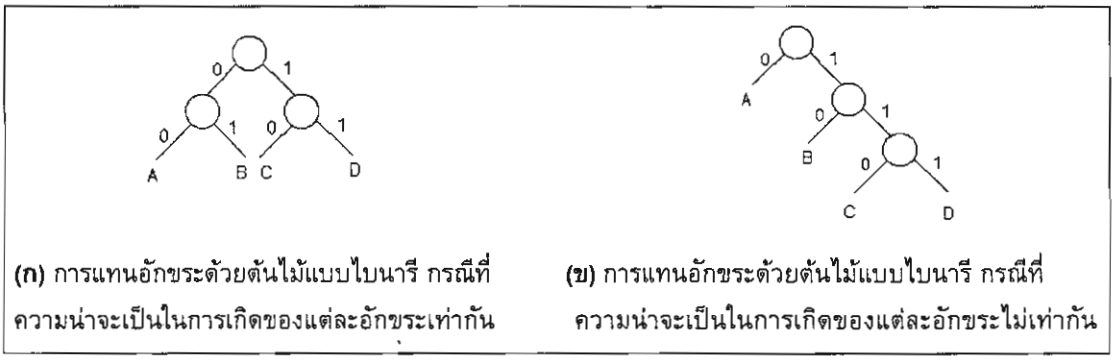
จากตัวอย่างเดิมสมมติว่าความน่าจะเป็นในการเกิดแต่ละอักขระเป็นดังนี้

A	มีความน่าจะเป็นคือ 1/2	แทนรหัสเดิมด้วย 0
B	มีความน่าจะเป็นคือ 1/4	แทนรหัสเดิมด้วย 10
C	มีความน่าจะเป็นคือ 1/8	แทนรหัสเดิมด้วย 110
D	มีความน่าจะเป็นคือ 1/8	แทนรหัสเดิมด้วย 111

หากต้องการเข้ารหัสข้อมูลชุดเดิมคือ ABACADAB จะได้ 01001100111010 จำนวนบิตที่ใช้ คือ 14 จะเห็นว่าเมื่อนำความน่าจะเป็นในการเกิดของแต่ละอักขระมาพิจารณาด้วย จะทำให้จำนวนบิตเฉลี่ยในการ

เข้ารหัสข้อมูลลดลง โดยอาศัยหลักพื้นฐานที่ว่าหากอักขระใดใช้บ่อยก็ให้แทนด้วยจำนวนบิตน้อยๆ ส่วนอักขระใดไม่ค่อยใช้ก็ยอมให้แทนด้วยจำนวนบิตที่ยาวกว่า

เมื่อพิจารณาในกรณีแรกหากความน่าจะเป็นในการเกิดข้อมูลของแต่ละอักขระมีค่าเท่ากันก็เหมาะสมแล้วที่จะแทนอักขระแต่ละตัวด้วยรหัสไบนารีจำนวน 2 บิต เพื่อให้ง่ายแก่การเข้าใจจึงแสดงวิธีการแทนอักขระกรณีที่ความน่าจะเป็นในการเกิดข้อมูลของแต่ละอักขระเท่ากันและไม่เท่ากันด้วยรูปที่ 7 (ก) และ (ข) ตามลำดับ



(ก) การแทนอักขระด้วยต้นไม้แบบไบนารี กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระเท่ากัน

(ข) การแทนอักขระด้วยต้นไม้แบบไบนารี กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระไม่เท่ากัน

รูปที่ 7 การแทนอักขระด้วยต้นไม้แบบไบนารี

รูปที่ 7 (ก) และ (ข) แสดงการแทนอักขระด้วยต้นไม้แบบไบนารี จะเห็นว่าในรูปที่ 7 (ก) กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระเท่ากัน ต้นไม้แบบไบนารีที่สมดุลจะให้จำนวนเส้นเชื่อม (edge) จากรากไปถึงแต่ละใบนั้นเท่ากัน แต่ในรูปที่ 7 (ข) กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระไม่เท่ากันจะให้จำนวนเส้นเชื่อมจากรากไปถึงใบแต่ละใบด้วยจำนวนเส้นเชื่อมมากน้อยต่างกันขึ้นกับความน่าจะเป็นในการเกิดของอักขระนั้น เช่น อักขระ A มีความน่าจะเป็นสูงสุดจะให้จำนวนเส้นเชื่อมจากรากไปถึงใบต่ำที่สุด ส่วนอักขระ C และ D จะมีจำนวนเส้นเชื่อมจากรากไปถึงใบมากที่สุดเนื่องจากมีความน่าจะเป็นในการเกิดต่ำสุด

หากพิจารณาในแง่ของการค้นหาข้อมูลของรูปที่ 7 (ข) อักขระ A เกิดบ่อย จึงแทนด้วยจำนวนบิตที่น้อย หรือกล่าวในเชิงการค้นหาข้อมูลคือ จำนวนครั้งในการค้นหาน้อย ส่วนอักขระ C และ D มีโอกาสเกิดน้อย จึงแทนด้วยจำนวนบิตที่ยาวได้หรือจำนวนครั้งในการค้นหาสูงได้ เมื่อพิจารณาการค้นหาโดยรวมจึงทำให้เวลาการค้นหาเฉลี่ยต่ำสุด

จากข้างต้นการสร้างต้นไม้สำหรับการจำแนกแบบหลายกลุ่มเทียบได้กับการเข้ารหัสข้อมูลโดยแต่ละโนดจะเป็นตัวจำแนกแบบสองกลุ่มที่เลือกมา ซึ่งจำนวนครั้งในการจำแนกข้อมูลเฉลี่ยย่อมขึ้นกับความน่าจะเป็นของการเกิดข้อมูลในแต่ละกลุ่มเช่นเดียวกับความน่าจะเป็นของการเกิดอักขระ

เทคนิคการจำแนกข้อมูลแบบแตกครึ่งตามสารสนเทศ ซึ่งมีแนวคิดในการจำแนกข้อมูลด้วยระนาบหลายมิติที่แบ่งข้อมูลออกเป็นสองส่วนได้ดีที่สุดโดยพิจารณาจากความน่าจะเป็นในการเกิดข้อมูลแต่ละกลุ่ม ซึ่งมีขั้นตอนในการจำแนกข้อมูลดังนี้

(1) การค้นหาตำแหน่งของระนาบหลายมิติ

การสร้างฟังก์ชันของระนาบหลายมิติในการจำแนกแบบสองกลุ่มตามปกตินั้นจะได้ตำแหน่งของระนาบอยู่กึ่งกลางระหว่างข้อมูลสองกลุ่มที่เราใช้สอนโดยที่ตำแหน่งของข้อมูลทั้งสองกลุ่มจะอยู่คนละด้านของระนาบแต่จะไม่รู้ตำแหน่งเทียบกับกลุ่มข้อมูลอื่น จึงมีการค้นหาตำแหน่งของระนาบเทียบกับกลุ่มข้อมูลอื่นเพิ่มโดยนำฟังก์ชันของระนาบแบบสองกลุ่มที่มีอยู่แล้ว ไปคำนวณหาตำแหน่งของกลุ่มข้อมูลอื่นเพิ่มเติมก็จะได้ตำแหน่งของระนาบเมื่อเทียบกับข้อมูลทั้งหมด เช่นเดียวกับในวิธีแตกครึ่งแบบสมดุล

(2) การค้นหาระนาบแตกครึ่งตามสารสนเทศ

เมื่อทำการคำนวณเพื่อค้นหาตำแหน่งของระนาบหลายมิติเทียบกับกลุ่มข้อมูลอื่นตามขั้นตอนข้างต้นแล้วจะทำการค้นหาระนาบที่สามารถแบ่งข้อมูลได้ดีที่สุดโดยพิจารณาจากค่าเอนโทรปีของข้อมูลโดยเลือกระนาบที่ให้ค่าเอนโทรปีต่ำสุด ซึ่งจะเป็นตัวบ่งชี้ว่าหากนำระนาบนั้นไปใช้ในการจำแนกข้อมูลแล้ว ข้อมูลที่ได้ในแต่ละด้านของระนาบจะมีจำนวนกลุ่มที่ปนกันอยู่น้อยที่สุดตามหลักของทฤษฎีสารสนเทศที่กล่าวข้างต้น เมื่อแต่ละด้านมีจำนวนกลุ่มของข้อมูลปนกันอยู่น้อยจึงสามารถตัดกลุ่มที่ไม่ใช่ออกไปได้ดีด้วย

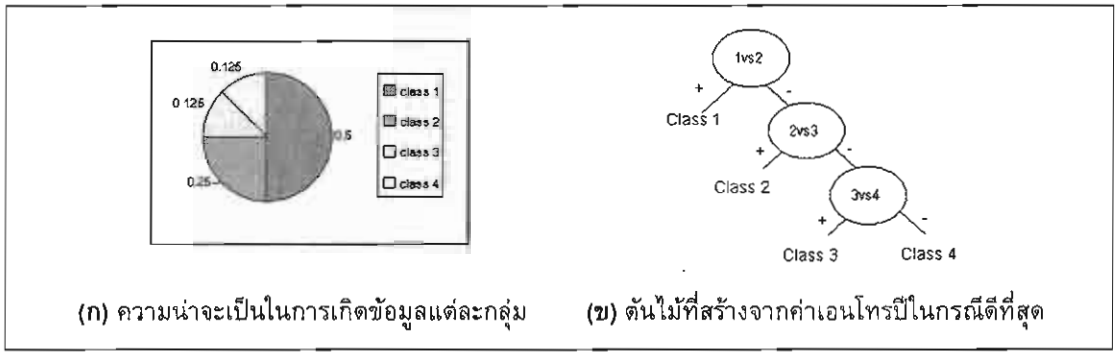
สูตรที่ใช้ในการคำนวณค่าเอนโทรปีเป็นดังนี้

$$\text{ค่าเอนโทรปีของตัวจำแนก} = \sum_{i=1}^{n} \frac{|t_i|}{|T|} I(t_i) \tag{23}$$

$$I(t_i) = \sum_{i=1}^{k} -P(m_i)\log_2 P(m_i) \tag{24}$$

โดยที่ n เป็นจำนวนกิ่งของต้นไม้ในที่นี้จะมีค่าเป็น 2 ซึ่งได้แก่ กิ่งบวกและกิ่งลบ, k เป็นจำนวนกลุ่มของข้อมูลทั้งหมด, $P(m_i)$ คือความน่าจะเป็นในการเกิดข้อมูลของกลุ่มที่ m_i, $|T|$ คือจำนวนข้อมูลทั้งหมด และ $|t_i|$ คือจำนวนข้อมูลที่อยู่ในแต่ละกิ่งของต้นไม้

ตัวอย่างเช่น กำหนดให้ชุดข้อมูลหนึ่งมีจำนวนกลุ่มเป็น 4 กลุ่มและมีความน่าจะเป็นในการเกิดข้อมูลของแต่ละกลุ่มเป็นดังรูปที่ 8 (ก) และต้นไม้สำหรับการจำแนกแบบหลายกลุ่มที่สร้างจากค่าเอนโทรปีของข้อมูลชุดดังกล่าวกรณีที่ดีที่สุดแสดงดังรูปที่ 8 (ข)



(ก) ความน่าจะเป็นในการเกิดข้อมูลแต่ละกลุ่ม **(ข)** ต้นไม้ที่สร้างจากค่าเอนโทรปีในกรณีดีที่สุด

รูปที่ 8 ตัวอย่างต้นไม้ที่สร้างจากค่าเอนโทรปีสำหรับข้อมูลที่มีความน่าจะเป็นที่จะเกิดต่างกัน

การคำนวณค่าจำนวนครั้งในการจำแนกที่คาดหวังสามารถคำนวณได้จากสูตร

$$\sum_{i=1}^{k} -P(m_i)\log_2 P(m_i) \tag{25}$$

สำหรับปัญหาการจำแนก k กลุ่ม และ $P(m_i)$ คือความน่าจะเป็นในการเกิดข้อมูลของกลุ่มที่ m_i กรณีตัวอย่างในรูปที่ 8 สามารถคำนวณค่าจำนวนครั้งในการจำแนกที่คาดหวังได้ดังนี้

$$\begin{aligned}
&\sum_{i=1}^{4} -P(m_i)\log_2 P(m_i) \\
&= [-(0.5) \times \log_2(0.5)] + [-(0.25) \times \log_2(0.25)] + \\
&\quad [-(0.125) \times \log_2(0.125)] + [-(0.125) \times \log_2(0.125)] \\
&= 1.75
\end{aligned} \tag{26}$$

ผลการทดลอง

เราทำการทดลองโดยใช้ข้อมูลทดสอบจาก UCI Machine Learning Repository [Blake et al., 1998] จำนวน 4 ชุด ดังตารางที่ 4 ข้อมูลแต่ละชุดจะประกอบด้วย ข้อมูลสอนและข้อมูลทดสอบ ในขั้นตอนของการทดลองจะทำการแบ่งชุดข้อมูลสอนออกเป็น ข้อมูลสอนจริงและข้อมูลทดสอบความถูกต้อง (Validation data) เพื่อใช้ในการหาค่าพารามิเตอร์ที่เหมาะสม [Kijsirikul et al., 2004] ซึ่งประกอบด้วยค่า P คือค่าร้อยละในการตัดเล็ม ($0 \le P \le 10$) และ R คือ ค่าขอบเขตความผิดพลาด ($x_{\min} \le R \le x_{mean+sd}$ เมื่อ x_{min} คือ ค่าต่ำสุดของขอบเขตความผิดพลาดของตัวจำแนกทั้งหมด และ $x_{mean+sd}$ คือ ผลรวมของค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของขอบเขตความผิดพลาดของตัวจำแนกทั้งหมด) หลังจากนั้นจึงใช้ค่า P และ R ที่ได้เพื่อสร้างตัวจำแนกแบบหลายกลุ่มจากชุดข้อมูลสอนเดิมและใช้ชุดข้อมูลทดสอบเพื่อหาค่าความถูกต้อง และค่าจำนวนครั้งเฉลี่ยในการจำแนกข้อมูล ผลการทดลองแสดงในตารางที่ 5 และตารางที่ 6 ในตารางทั้งสองนั้น IBD คือวิธีการที่นำเสนอ BD คือ Balanced Dichotomization [Kijsirikul et al., 2004], 'Expected Value' และ 'Output' คือ ค่าจำนวนครั้งในการจำแนกที่คาดหวังและจำนวนครั้งในการจำแนกที่ใช้จริง ส่วนตัวอักษร c, d คือพารามิเตอร์ของฟังก์ชันเคอร์เนล และตัวอักษรเข้มในตารางแสดงค่าที่ดีที่สุดในชุดข้อมูลแต่ละชุด

ตารางที่ 4 ลักษณะของข้อมูลที่นำมาทดสอบ

ชุดข้อมูล	จำนวนตัวอย่างสอน	จำนวนตัวอย่างทดสอบ	จำนวนกลุ่ม	จำนวนคุณสมบัติ
Satimage	4,435	2,000	6	36
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35

ตารางที่ 5 เปรียบเทียบจำนวนครั้งของการจำแนก

ชุดข้อมูล	Polynomial Kernel					
	Expected Value of IBD	Output of IBD	Expected Value of BD ($\log_2 k$)	Output of BD	RADAG (k-1)	Max Wins (k(k-1)/2)
Satimage	2.474	4.999	2.585	**4.847**	5	15
Shuttle	0.965	**5.359**	2.807	5.378	6	21
Vowel	3.459	**5.385**	3.459	5.665	10	55
Soybean	3.617	6.971	3.907	**6.783**	14	105
ชุดข้อมูล	RBF Kernel					
	Expected Value of IBD	Output of IBD	Expected Value of BD ($\log_2 k$)	Output of BD	RADAG (k-1)	Max Wins (k(k-1)/2)
Satimage	2.474	4.734	2.585	**4.577**	5	15
Shuttle	0.965	**4.982**	2.807	5.758	6	21
Vowel	3.459	**5.628**	3.459	5.819	10	55
Soybean	3.617	**5.400**	3.907	6.591	14	105

ตารางที่ 6 เปรียบเทียบเปอร์เซ็นต์ความถูกต้อง

ชุดข้อมูล	Polynomial Kernel							
	d	IBD	d	BD	d	RADAG	d	Max Wins
Satimage	6	88.850	6	87.840	6	**88.900**	6	88.453
Shuttle	8	**99.924**	8	**99.924**	8	**99.924**	8	**99.924**
Vowel	3	64.935	2	**65.022**	3	64.502	3	64.329
Soybean	3	90.882	4	89.529	3	**91.176**	3	90.471
ชุดข้อมูล	RBF Kernel							
	c	IBD	c	BD	c	RADAG	c	Max Wins
Satimage	0.5	91.950	1.0	91.350	0.5	91.950	0.5	**91.984**
Shuttle	3.0	99.890	3.0	99.886	3.0	**99.897**	3.0	**99.897**
Vowel	0.3	66.450	0.2	62.900	0.2	**67.100**	0.2	65.340
Soybean	0.07	**91.471**	0.04	89.118	0.07	90.882	0.08	90.468

จากผลการทดลองการแตกครึ่งตามสารสนเทศให้ค่าความถูกต้องใกล้เคียงกับวิธีอื่น ส่วนจำนวนครั้งในการจำแนกเมื่อเปรียบเทียบกับอาร์เอดีเอจีและแมกซ์วิน พบว่าการแตกครึ่งตามสารสนเทศให้ค่าจำนวนครั้งในการจำแนกต่ำกว่าทุกกรณี และเมื่อเปรียบเทียบกับการแตกครึ่งแบบสมดุล พบว่าการแตกครึ่งตามสารสนเทศให้ค่าจำนวนครั้งในการจำแนกต่ำกว่าถึง 5 ใน 8 กรณี ส่วนอีก 3 ใน 8 กรณี แม้การแตกครึ่งตามสารสนเทศจะให้จำนวนครั้งในการจำแนกที่สูงกว่าแต่ก็ให้ค่าความถูกต้องที่สูงกว่าด้วย

ความสามารถในการลดจำนวนครั้งของการจำแนกจะขึ้นกับระนาบของตัวจำแนกแบบสองกลุ่มที่สร้างได้จากค่า P และ R ที่ดีที่สุดด้วย โดยหากระนาบที่ได้สามารถแบ่งกลุ่มของข้อมูลได้ดีนั่นคือ แต่ละด้านของระนาบมีกลุ่มของข้อมูลปนกันน้อยหรือถ้าหากด้านใดด้านหนึ่งของระนาบมีเพียงกลุ่มเดียว ค่าจำนวนครั้งเฉลี่ยของการจำแนกที่ได้จริงยิ่งให้ค่าใกล้เคียงกับค่าจำนวนครั้งในการจำแนกที่คาดหวัง ในทางตรงกันข้ามหากระนาบที่ได้ไม่มีระนาบใดที่แบ่งกลุ่มข้อมูลได้ดีพอก็ยิ่งส่งผลให้ตัวจำแนกแบบหลายกลุ่มที่ได้มีจำนวนครั้งในการจำแนกเฉลี่ยสูงและคลาดเคลื่อนจากค่าจำนวนครั้งในการจำแนกที่คาดหวังมากขึ้นเท่านั้น

1.4 สรุปผลการวิจัยอัลกอริทึมเอสวีเอ็มแบบหลายกลุ่ม

งานวิจัยนี้ได้เสนอวิธีใหม่สำหรับการจำแนกข้อมูลแบบหลายกลุ่มของซัพพอร์ตเวกเตอร์แมชชีน โดยนำเสนอเอดีเอจีซึ่งเป็นโครงสร้างกราฟปรับปรุงจากวิธีดีดีเอจี ผลที่ได้ทำให้สามารถลดจำนวนครั้งของการเปรียบเทียบเพื่อหากลุ่มที่ถูกต้องลดลงได้อย่างมาก และส่งผลให้ความถูกต้องสูงขึ้น อย่างไรก็ดีเอดีเอจียังคงขึ้นอยู่กับลำดับของโนด (ตัวจำแนกข้อมูลแบบสองกลุ่ม) แม้ว่าจะมีความขึ้นต่อกันของลำดับของตัวจำแนกข้อมูลแบบสองกลุ่มน้อยกว่าวิธีดีดีเอจี แต่ยังคงให้ความถูกต้องแตกต่างกันเมื่อมีลำดับของโนดแตกต่างกัน เราจึงทำการปรับปรุงเอดีเอจีเพิ่มเติม โดยพยายามขจัดความขึ้นต่อกันของลำดับของตัวจำแนกข้อมูลแบบสองกลุ่มในโนดในโครงสร้างของเอดีเอจีโดยเลือกลำดับที่เหมาะสม ซึ่งจะพิจารณาจากตัวจำแนกที่มีค่าขอบเขตของความผิดพลาดต่ำ และได้นำอัลกอริทึมการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุดมาใช้ ทำให้วิธีอาร์เอดีเอจีสามารถจัดเรียงลำดับของตัวจำแนกได้ในเวลาพหุนาม ผลการทดลองแสดงให้เห็นว่าโดยส่วนใหญ่วิธีอาร์เอดีเอจีให้ความถูกต้องสูงกว่าวิธีเอดีเอจี หรือแม้แต่แมกซ์วินซึ่งเป็นวิธีที่ให้ความถูกต้องสูงที่สุดสำหรับปัญหาการจำแนกข้อมูลแบบหลายกลุ่มของซัพพอร์ตเวกเตอร์แมชชีนในปัจจุบัน นอกจากนี้ยังแสดงให้เห็นว่าวิธีอาร์เอดีเอจีใช้เวลาในการประมวลผลต่ำกว่าแมกซ์วิน

นอกจากนี้แล้วเรายังได้นำเสนอเทคนิคการแตกครึ่งตามสารสนเทศสามารถเพิ่มประสิทธิภาพการจำแนกกลุ่มของซัพพอร์ตเวกเตอร์แมชชีนแบบหลายกลุ่มโดยลดจำนวนครั้งของการจำแนกลงได้มากขึ้นกว่าของอาร์เอดีเอจี โดยผลดังกล่าวจะยิ่งปรากฏชัดเจนในกรณีปัญหาที่มีจำนวนกลุ่มเป็นจำนวนมาก อีกทั้งยังคงให้ค่าความถูกต้องใกล้เคียงกับตัวจำแนกแบบหลายกลุ่มวิธีอื่น

2. การวิจัยการปรับปรุงประสิทธิภาพของไอแอลพีให้ทนทานต่อสัญญาณรบกวน

ไอแอลพีมีจุดเด่นที่ผู้สอนสามารถให้ความรู้ภูมิหลังกับระบบเรียนรู้ได้ ทำให้การเรียนรู้มีประสิทธิภาพสูงขึ้นและใช้การแทนความรู้โดยกฎตรรกะอันดับที่หนึ่งซึ่งเป็นการแทนความรู้ที่มีประสิทธิภาพมาก อยากไรก็ดีไอแอลพีประสบปัญหาในกรณีที่เราต้องการนำกฎที่ได้จากการเรียนรู้ไปจำแนกกลุ่มตัวอย่างใหม่โดยเฉพาะอย่างยิ่งในกรณีที่ข้อมูลมีสัญญาณรบกวน เนื่องจากกฎที่ได้จากไอแอลพีไม่มีความยืดหยุ่นเพียงพอทำให้การจำแนกประเภทตัวอย่างใหม่ผิดพลาดได้ หัวข้อนี้นำเสนอการประยุกต์ใช้ข่ายงานเบส์และนิวรอลเน็ตเวิร์กเพื่อทำให้ไอแอลพีมีความยืดหยุ่นมากขึ้นทำงานกับข้อมูลที่มีสัญญาณรบกวนได้ดียิ่งขึ้น

2.1 ข่ายงานเบส์ลำดับที่หนึ่ง

2.1.1 แนวคิด

ในช่วงหลายปีที่ผ่านมานี้ได้มีการวิจัยอย่างจริงจังในการพัฒนาตัวจำแนกประเภทข้อมูลที่สามารถเอาชนะข้อจำกัดทั้งสองข้อดังกล่าวซึ่งได้แก่ *ข่ายงานเบส์ลำดับที่หนึ่ง (First-Order Bayesian Network : FOBN)* [Getoor et al., 2001; Kersting & De Raedt, 2002] โดยข่ายงานเบส์ลำดับที่หนึ่งได้รวมข้อดีของตัวจำแนกประเภท 2 ชนิด คือ*ตรรกะลำดับที่หนึ่ง (First-order Logic : FOL)* และ*ข่ายงานเบส์ (Bayesian Network : BN)* [Mitchell, 1997; Pearl, 1991] โดยตรรกะลำดับที่หนึ่งมีความสามารถจัดการกลุ่มข้อมูลที่ข้อมูลแต่ละหน่วยอาจส่งผลกระทบต่อกัน ในขณะที่ข่ายงานเบส์มีความสามารถในการจัดการข้อมูลที่ไม่สามารถแบ่งกลุ่มได้ชัดเจนได้อย่างมีประสิทธิภาพด้วยทฤษฎีความน่าจะเป็น อย่างไรก็ตามเนื่องจากข่ายงานเบส์ลำดับที่หนึ่งมีความซับซ้อนมาก ระบบการเรียนรู้เพื่อสร้างข่ายงานเบส์ลำดับที่หนึ่งจากข้อมูลดิบจึงพัฒนาได้ยากยิ่ง

ก่อนที่จะกล่าวถึงระบบการเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งที่นำเสนอนั้นจะขอกล่าวถึงแนวคิดของแบ่งประเภทข้อมูลก่อน สมมติว่ามีข้อมูลดิบที่ใช้ในการเรียนรู้ดังตารางที่ 7

ตารางที่ 7 ตัวอย่างข้อมูลดิบ

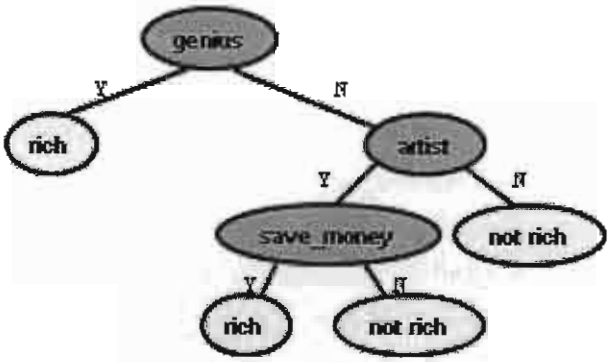
a. ความรวยและคุณสมบัติของบุคคล

ตัวแทน	rich	genius	artist	diligent	save_money
b	+	1	0	0	0
c	+	0	1	0	1
d	+	1	1	1	1
e	+	0	0	0	0
f	-	0	1	0	0
g	-	0	0	1	0

b. ผู้ปกครอง

parent	child
b	e
e	f
f	g

จากข้อมูลดิบในตารางข้างต้นระบบจำแนกประเภทข้อมูลชนิดต้นไม้ตัดสินใจ (Decision Tree) [Mitchell, 1997] สามารถสร้างต้นไม้ได้ดังรูปที่ 9



รูปที่ 9 ต้นไม้ตัดสินใจอย่างง่าย

เราสามารถเขียนกฎที่ได้จากต้นไม้ตัดสินใจให้อยู่ในรูปตรรกศาสตร์ประพจน์ (Propositional Logic) ได้สองข้อดังนี้

```
rich ← genius, diligent.
rich ← artist, save_money.
```

สังเกตว่ากฎที่ได้คนๆ หนึ่งจะรวยหรือไม่จะขึ้นอยู่กับคุณสมบัติของคนๆ นั้นเท่านั้น ซึ่งในความเป็นจริงแล้วอาจไม่เป็นเช่นนั้นก็ได้ บุคคลใดจะรวยหรือไม่อาจเกี่ยวข้องกับปัจจัยจากบุคคลอื่นเช่น มีผู้ปกครองที่รวย เป็นต้น อย่างไรก็ตามการใช้ตัวจำแนกประเภทข้อมูลชนิดต้นไม้ตัดสินใจหรือตรรกศาสตร์ประพจน์นี้ไม่สามารถเรียนรู้กฎในลักษณะที่บุคคลหนึ่งขึ้นกับบุคคลอื่นได้อย่างสะดวกเช่น พิจารณากฎ 'rich ← rich' จากกฎนี้เราไม่สามารถบอกได้ชัดเจนว่าความรวยของบุคคลใดส่งผลต่อบุคคลใด ในงานวิจัยนี้เราเรียกข้อมูลดิบประเภทที่ข้อมูลในแต่ละหน่วยสามารถมีผลกระทบต่อหน่วยอื่นได้ว่า*ข้อมูลที่มีความสัมพันธ์ต่อกัน (relational data)* การมีความสัมพันธ์กันระหว่างข้อมูลทำให้การเรียนรู้ทำได้ยากยิ่งขึ้นเช่น โดยปกติต้นไม้ตัดสินใจไม่สามารถสร้างกฎบางกฎเช่น ความรวยของลูกขึ้นกับผู้ปกครอง อย่างไรก็ตามตรรกะลำดับที่หนึ่งซึ่งถูกพัฒนาจากตรรกศาสตร์ประพจน์สามารถเขียนกฎสำหรับข้อมูลที่มีความสัมพันธ์ต่อกันได้อย่างชัดเจนโดยมีการนำตัวแปรทางตรรกศาสตร์มาใช้ดังนี้

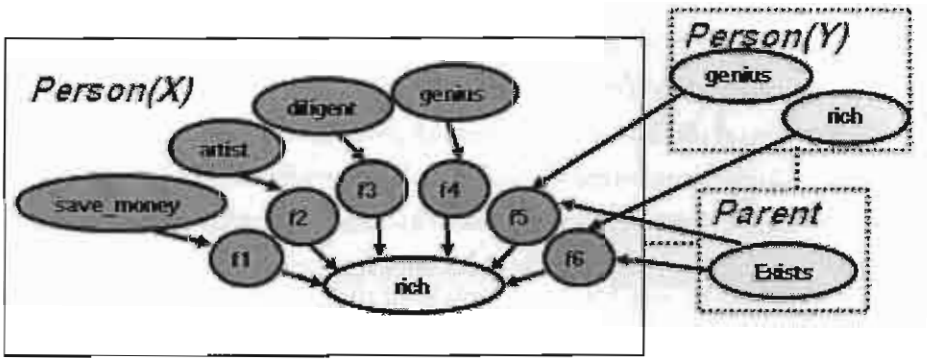
```
rich(X) ← genius(X), diligent(X).
rich(X) ← artist(X), save_money(X).
rich(X) ← parent(Y,X), rich(Y), genius(Y).
```

โดยในกรณีนี้ตัวแปรแต่ละตัวแสดงถึงบุคคล เช่น ในกฎข้อที่สามระบุว่าถ้า Y เป็นผู้ปกครองของ X และ Y เป็นคนรวยและเป็นอัจฉริยะแล้ว X จะเป็นคนรวย (Y ตรงกับนาย b และ X ตรงกับนาย e) ส่วนกฎข้อที่หนึ่งและข้อที่สองนั้นเทียบเท่ากับต้นไม้ตัดสินใจในรูปที่ 9 จะเห็นได้ว่าการใช้ตรรกะลำดับที่หนึ่งในการจำแนกประเภทข้อมูลแทนต้นไม้ตัดสินใจช่วยลดขีดจำกัดของกฎได้อย่างมาก สำหรับวิธีมาตรฐานในการเรียนรู้ตรรกะลำดับที่หนึ่งจากข้อมูลดิบนั้นก็คือ*การโปรแกรมตรรกะเชิงอุปนัย – ไอแอลพี (Inductive Logic Programming – ILP)* [Muggleton, 1991; Lavrac & Dzeroski, 1994] อย่างไรก็ตามระบบไอแอลพียังไม่สามารถจัดการข้อมูลไม่สามารถแบ่งกลุ่มได้แน่นอนหรือมีสัญญาณรบกวนได้ดีนัก เมื่อจำเป็นต้องเรียนรู้กฎจากข้อมูลดิบที่ไม่ดีเหล่านี้ทั้งระบบการเรียนรู้ต้นไม้ตัดสินใจและไอแอลพีจะเผชิญกับปัญหาที่เรียกว่าการปรับเหมาะเกินไป

(overfitting problem) [Mitchell, 1997] ซึ่งจะทำให้ต้นไม้ตัดสินใจหรือตรรกะลำดับที่หนึ่งที่เรียนรู้ได้มีลักษณะเฉพาะเจาะจง (specific) กับข้อมูลอินพุตมากเกินไป (ทำงานได้ถูกต้องเฉพาะข้อมูลอินพุตเท่านั้น) พิจารณาตรรกะลำดับที่หนึ่งทั้งสามข้อในตัวอย่างที่ผ่านมาพบว่าในบางกรณีกฎเหล่านี้มีความเฉพาะเจาะจงมากเกินไปเช่นกัน ตัวอย่างเช่นถ้าเพิ่มนาย a ลงไปในตารางที่ 7 โดยกำหนดคุณสมบัติให้นาย a ดังต่อไปนี้

```
genius(a). save_money(a). parent(e,a).
```

จะเห็นได้ว่าคุณสมบัติต่างๆ ของนาย a ไม่ตรงกับกฎข้อใดเลยทำให้เราสรุปจากกฎได้ว่านาย a เป็นบุคคลที่ไม่รวย อย่างไรก็ตามสังเกตว่าการสรุปแบบนี้เป็นการสรุปที่ไม่มีประสิทธิภาพนักเนื่องจากคุณสมบัติต่างๆ ของนาย a *ตรงบางส่วน (partially match)* กับกฎตรรกะลำดับที่หนึ่งในตัวอย่างที่แล้วทั้งสามกฎ คุณสมบัติการตรงบางส่วนนี้เองเป็นที่มาของการใช้ข่ายงานเบส์ลำดับที่หนึ่งเป็นตัวจำแนกประเภทข้อมูล โดยเมื่อกำหนดคุณสมบัติต่างๆ ของนาย a ให้กับข่ายงานเบส์ลำดับที่หนึ่งแล้ว ข่ายงานเบส์ลำดับที่หนึ่งจะสามารถคำนวณหาค่าความจะเป็นภายหลัง (posterior probability) ที่นาย a จะรวยว่ามากน้อยเพียงใดด้วยเทคนิคการอนุมาน (inference) [Pearl, 2001] ได้อย่างมีประสิทธิภาพ จะเห็นได้ว่าการใช้ข่ายงานเบส์ลำดับที่หนึ่งเป็นตัวจำแนกประเภทข้อมูลมีความยืดหยุ่นเป็นอย่างมากเนื่องจากรองรับทั้งคุณสมบัติความไม่แน่นอนของข้อมูลได้ด้วยการใช้ทฤษฎีความน่าจะเป็น (แทนที่จะสรุปว่า *ใช่* หรือ *ไม่ใช่* ซึ่งเป็นวิธีที่ต้นไม้ตัดสินใจและตรรกะลำดับที่หนึ่งใช้) นอกจากนี้ข่ายงานเบส์ลำดับที่หนึ่งยังมีคุณสมบัติรองรับความสัมพันธ์ระหว่างข้อมูลเช่นเดียวกับตรรกะอันดับที่หนึ่งอีกด้วย รูปที่ 10 แสดงข่ายงานเบส์ลำดับที่หนึ่งที่ใช้แก้ปัญหาความรวยในตัวอย่างนี้



รูปที่ 10 ข่ายงานเบส์ลำดับที่หนึ่งที่เพิ่มความสามารถจากต้นไม้ตัดสินใจและตรรกะอันดับที่หนึ่ง

ข่ายงานเบส์ลำดับที่หนึ่งประกอบด้วยส่วนสำคัญสองส่วนเช่นเดียวกับข่ายงานเบส์คือ*โครงสร้าง (structure)* ซึ่งหมายถึงโนดต่างๆ และเส้นเชื่อมแบบมีทิศทาง (directed link) ทั้งหมดที่เชื่อมโนดแต่ละโนดเข้าไว้ด้วยกัน โนดแต่ละโนดแทนคุณสมบัติหรือคุณลักษณะ (attribute) เช่นเดียวกับต้นไม้ตัดสินใจ การมีเส้นเชื่อมจากโนด A ไปยังโนด B หมายถึงโนด A ส่งผลกระทบโดยตรงต่อโนด B ส่วนประกอบส่วนที่สองของข่ายงานเบส์ลำดับที่หนึ่งคือ*ตารางของความน่าจะเป็นแบบมีเงื่อนไข – ซีพีที (conditional probability table – CPT)* โดยเราคำนวณความน่าจะเป็นภายหลังเพื่อทำนายคุณสมบัติจากซีพีทีนี้เอง โดยทั่วไปการเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งจึงประกอบไปด้วยสองส่วนหลักคือเรียนรู้โครงสร้างและซีพีที อย่างไรก็ตามการเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งไม่สามารถทำได้โดยง่ายนักเนื่องจากได้มีการพิสูจน์ว่าเพียงแค่การเรียนรู้ตรรกะลำดับที่หนึ่งหรือข่ายงานเบส์เพียงลำพังก็ยังมีความซับซ้อนของอัลกอริทึมที่ใช้ในการเรียนรู้เป็นแบบ NP [Botta et al., 2000; Chickering, 1996] ดังนั้นการเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งซึ่งเสมือนเป็นการเรียนรู้ทั้งตรรกะลำดับที่หนึ่งและข่ายงานเบส์พร้อมกันอย่างถูกต้องและรวดเร็วจึงทำได้ยากยิ่ง งานวิจัยนี้จึงได้เสนอแนวคิดใหม่โดย

การนำระบบไอแอลพีและ*ตัวเรียนรู้ข่ายงานเบส์ (Bayesian Network Learner : BNL)* [Chickering, 2002; Pearl, 2001] สองระบบมาทำงานต่อเนื่องกันเพื่อลดความซับซ้อนของระบบเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่ง

ในการใช้ตัวเรียนรู้ข่ายงานเบส์เพื่อเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งนั้น ตัวเรียนรู้ข่ายงานเบส์ต้องการข้อมูลอินพุตประเภทฐานข้อมูลตารางเดี่ยวเช่นเดียวกับการเรียนรู้ต้นไม้ตัดสินใจดังเช่น ตารางที่ 7 a. อินพุตประกอบไปด้วยตัวอย่าง n ตัว ตัวอย่างแต่ละตัวประกอบไปด้วยค่าของคุณสมบัติ m ค่า ตัวอย่างหนึ่งตัวแทนด้วยข้อมูลหนึ่งแถวในตารางอินพุต สดมภ์หนึ่งๆ แทนคุณสมบัติหนึ่งอย่าง สังเกตว่าเราต้องกำหนดคุณสมบัติหรือโนดทั้งหมดของตัวอย่างที่ต้องการเรียนให้ตัวเรียนรู้ข่ายงานเบส์ จากนั้นในขั้นตอนการเรียนรู้ ตัวเรียนรู้ข่ายงานเบส์จะสร้างเส้นเชื่อมที่เหมาะสมรวมทั้งซีพีทีของโนดเหล่านั้น ดังนั้นถ้าเราต้องการใช้ตัวเรียนรู้ข่ายงานเบส์เรียนข่ายงานเบส์ลำดับที่หนึ่งในลักษณะเดียวกับการเรียนข่ายงานเบส์ เราจึงจำเป็นต้องสร้างฐานข้อมูลตารางเดี่ยวจากข้อมูลอินพุตประเภทข้อมูลมีความสัมพันธ์ต่อกัน (ซึ่งอาจอยู่ในรูปฐานข้อมูลเชิงสัมพันธ์ (relational database) เช่นตารางที่ 7 a. และตารางที่ 7 b.) แต่ว่าการสร้างฐานข้อมูลตารางเดี่ยวจากข้อมูลลักษณะนี้มีปัญหาตรงที่ไม่สามารถกำหนดหลักหรือคุณสมบัติที่ต้องการให้แน่ชัดได้เพราะคุณสมบัติที่เป็นไปได้ทั้งหมดในอินพุตประเภทนี้สามารถมีได้มากมายมหาศาล [Kramer et al., 2001] ตัวอย่างเช่น อาจกำหนดให้ความรวยของพ่อแม่เป็นคุณสมบัติหนึ่งอย่าง ให้ความรวยของปู่ย่าเป็นคุณสมบัติอีกหนึ่งอย่าง (ด้วยการเพิ่มตัวแปรเช่น `parent(Z,Y)`, `parent(Y,X)` หมายถึง Z เป็นปู่ของ X) ความรวยของทวดเป็นคุณสมบัติอีกหนึ่งอย่าง จะเห็นได้ว่าวิธีนี้สามารถเพิ่มคุณสมบัติของฐานข้อมูลตารางเดี่ยวที่ต้องการได้อย่างไม่มีที่สิ้นสุด อย่างไรก็ตาม*การแปลงหรือลดรูปปัญหาจากข้อมูลที่มีความสัมพันธ์ต่อกันไปเป็นฐานข้อมูลตารางเดี่ยว (propositionalization)* นั้นสามารถทำได้โดยการเลือกเฉพาะคุณสมบัติที่สำคัญหรือ*ลักษณะสำคัญ (feature)* ของข้อมูลมาเป็นสดมภ์ของฐานข้อมูลตารางเดี่ยวก็เพียงพอ ไม่จำเป็นต้องนำคุณสมบัติทุกอย่างที่เป็นไปได้มาสร้างเป็นสดมภ์ทั้งหมด อย่างไรก็ตามปัญหาก็คือเราไม่อาจทราบล่วงหน้าว่าต้องกำหนดความสัมพันธ์ลงไปลึกกี่ขั้นถึงจะได้ลักษณะที่สำคัญที่ต้องการ (ดังเช่นความสัมพันธ์แบบบรรพบุรุษในตัวอย่างที่แล้ว) ซึ่งปัญหานี้เป็นปัญหาเดียวกับการเรียนรู้ตรรกะลำดับที่หนึ่งของไอแอลพีนั่นเอง ดังนั้นงานวิจัยนี้จึงเสนอวิธีการสร้างลักษณะสำคัญที่จำเป็น โดยสร้างลักษณะสำคัญเหล่านั้นจากตรรกะลำดับที่หนึ่งที่ได้จากระบบไอแอลพี ดังเช่นจากตัวอย่างที่ผ่านมาเราสามารถบอกได้ว่าลักษณะสำคัญของข้อมูลในแง่ความสัมพันธ์ทางบรรพบุรุษจะจำกัดไม่เกินรุ่นพ่อแม่ (กฎข้อ 3)

ระบบต้นแบบในการเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งที่เสนอแสดงในรูปที่ 11 ซึ่งอธิบายได้ดังนี้ ในขั้นแรกใช้ระบบไอแอลพีสร้างตรรกะลำดับที่หนึ่งตามปกติโดยมีอินพุตคือความรู้ภูมิหลัง (background knowledge) และข้อมูลดิบซึ่งอาจมาจากฐานข้อมูลเชิงสัมพันธ์ หลังจากนั้นนำตรรกะลำดับที่หนึ่งที่ได้มาสร้างลักษณะสำคัญและนำลักษณะสำคัญเหล่านั้นมาสร้างข่ายงานเบส์ลำดับที่หนึ่งโดยใช้ตัวเรียนรู้ข่ายงานเบส์ต่อไป โดยงานวิจัยนี้เสนออัลกอริทึมสองชิ้นคือ *MCAFEE* (จะกล่าวในหัวข้อต่อไป) เพื่อใช้ในการค้นหาลักษณะสำคัญต่างๆ จากตรรกะลำดับที่หนึ่ง เพื่อนำมาเป็นสดมภ์ของฐานข้อมูลตารางเดี่ยวที่ต้องการ และอัลกอริทึมชิ้นที่สองที่เสนอคือ *GRASP* (ในหัวข้อที่ 2.1.3) เป็นอัลกอริทึมที่จะสร้างแถวหรือตัวอย่าง (training examples) ในฐานข้อมูลตารางเดี่ยวที่ต้องการ

อย่างไรก็ตามการเรียนรู้ข่ายงานเบส์หรือข่ายงานเบส์ลำดับที่หนึ่งจากตัวเรียนรู้ข่ายงานเบส์โดยทั่วไปจะเป็นการเรียนรู้ในเชิงอธิบายลักษณะของข้อมูล (descriptive learning) มากกว่าการเรียนรู้เพื่อใช้ข่ายงานเบส์ลำดับที่หนึ่งไปใช้เชิงทำนายข้อมูล (predictive learning) [Mitchell, 1997; Pearl, 2001] (โดยการสร้างตัวจำแนกประเภทข้อมูลจากข้อมูลดิบนับเป็นการเรียนรู้เชิงทำนายข้อมูลเช่นกัน) เราเรียกข่ายงานเบส์ลำดับที่หนึ่งที่ถูกเรียนรู้เพื่องานทำนายข้อมูลโดยเฉพาะว่าตัวจำแนกประเภทข่ายงานเบส์ลำดับที่หนึ่ง *(FOBN classifier)* ซึ่งมีนิยามดังนี้

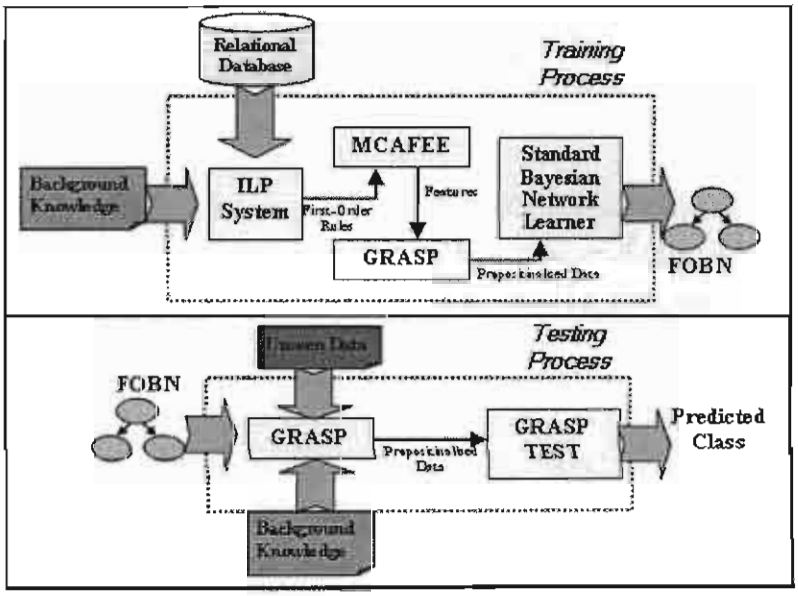
นิยามที่ 1 (ตัวจำแนกประเภทข่ายงานเบส์ลำดับที่หนึ่ง). ข่ายงานเบส์ลำดับที่หนึ่งใดๆ จะถูกเรียกว่าเป็นตัวจำแนกประเภท ข่ายงานเบส์ลำดับที่หนึ่งเมื่อมีคุณสมบัติสองข้อดังต่อไปนี้

(1) โนดของข่ายงานเบส์ลำดับที่หนึ่งที่เราสนใจจะทำนายคุณสมบัติ (target หรือ class node) จะต้องไม่มีโนดลูกใดๆ เลย

(2) โนดพ่อแม่ (parent node) ของโนดที่เราจะทำนายคุณสมบัติจะต้องเป็นลักษณะสำคัญต่างๆ เท่านั้น □

คุณสมบัติข้อที่ (1) มีจุดมุ่งหมายเพื่อป้องกันไม่ให้โนดที่เราต้องการทำนายไปทำนายคุณสมบัติของโนดอื่นซึ่งจะทำให้เราไม่ได้ตัวจำแนกประเภทข่ายงานเบส์ลำดับที่หนึ่งที่จะทำนายโนดที่ต้องการ ส่วนคุณสมบัติข้อที่ (2) มีจุดประสงค์ในการนำเทคนิคการตรงบางส่วนกับกฎมาใช้ทำนายโนดที่ต้องการ สังเกตว่านิยามที่ 1 ได้กำหนดเงื่อนไข (constraint) ของโครงสร้างของข่ายงานเบส์ลำดับที่หนึ่งที่เรียนรู้ได้ไว้ส่วนหนึ่งแล้วซึ่งเราจะเรียกโครงสร้างส่วนนี้ว่า*โครงสร้างหลัก* (*main structure*) และเรียกโครงสร้างส่วนที่เหลืออื่นๆ ที่เรียนได้จากตัวเรียนรู้ข่ายงานเบส์ว่า *โครงสร้างที่เหลือ* (*remaining structure*) รูปที่ 10 แสดงตัวจำแนกประเภทข่ายงานเบส์ลำดับที่หนึ่งที่ตรงตามนิยามที่ 1 โดยโนด `f1-f6` หมายถึงลักษณะสำคัญทั้ง 6 อย่าง (ดูรายละเอียดในหัวข้อที่ 2.1.2)

การคำนวณหาค่าความน่าจะเป็นภายหลังเพื่อทำนายข้อมูล เราใช้อัลกอริทึม GRASP (รูปที่ 11 ตอนล่าง) ซึ่งจะกล่าวละเอียดในหัวข้อที่ 2.1.3

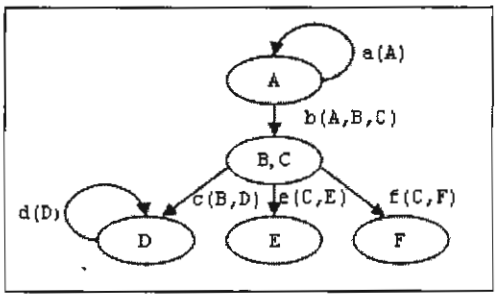


รูปที่ 11 ระบบต้นแบบการเรียนรู้และทำนายของตัวจำแนกประเภทข่ายงานเบส์ลำดับที่หนึ่ง

2.1.2 MCAFEE

หัวข้อนี้เสนออัลกอริทึม MCAFEE (Minimal ChAin FEature Extraction) เพื่อใช้ค้นหาลักษณะสำคัญจากตรรกะลำดับที่หนึ่งที่ได้จากไอแอลพี ในที่นี้เราจะเรียกส่วนหนึ่งของตรรกะลำดับที่หนึ่งว่าสายสัญพจน์ *(chain)* อาจเรียกลักษณะสำคัญที่ต้องการได้อีกอย่างหนึ่งว่า*สายสัญพจน์สำคัญ (significant chain)* โดยคุณสมบัติแรกของสายสัญพจน์สำคัญในงานวิจัยนี้คือต้องไม่ใช่สายสัญพจน์*ที่ไม่มีความหมาย (meaningless chain)* ในงานวิจัยนี้สายสัญพจน์ที่ไม่มีความหมายคือการมีตัวแปรบางตัวในสายสัญพจน์ถูกสร้างขึ้นมาโดยไม่ได้สัมพันธ์กับตัวแปรอื่นในสายสัญพจน์นั้น (ดู [Kijsirikul et al., 2001] เพิ่มเติม) เช่น พิจารณากฎ `'rich(X) :-`

`dad(Y,X), rich(Y), good(W).'` สังเกตว่าตัวแปร Y ถูกสร้างขึ้นมาโดยสัมพันธ์กับตัวแปร X ทำให้สามารถทำความเข้าใจ `rich(Y)` ได้ ขณะที่ตัวแปร W ถูกสร้างขึ้นมาอย่างไม่สัมพันธ์กับตัวแปรอื่นทำให้ไม่สามารถตีความหมายได้ชัดเจน ดังนั้นจากกฎข้อนี้สายสัญพจน์ใดๆ ที่มี `good(W)` อยู่จะไม่ถูกเลือกให้เป็นลักษณะสำคัญ วิธีการสร้างสายสัญพจน์ที่สำคัญจากตรรกะลำดับที่หนึ่งของ MCAFEE ใช้แนวคิดจาก [Kijsirikul et al., 2001] โดยจะมองกฎตรรกะลำดับที่หนึ่งเสมือนเป็นกราฟมีทิศทาง (directed graph) เช่นจากกฎ `'r(A) :- a(A), b(A,B,C), c(B,D), d(D), e(C,E), f(C,F).'` สามารถสร้างกราฟมีทิศทางได้ดังรูปที่ 12



รูปที่ 12 กราฟมีทิศทาง

โนดราก (root node) คือตัวแปรทั้งหมดที่อยู่ในส่วนหัวของกฎ (head of rule) ส่วนโนดอื่นๆ คือตัวแปรอื่นๆ ที่ถูกสร้างขึ้นในสัญพจน์ (literal) แต่ละตัว เส้นเชื่อม (edge) ในแต่ละโนดแทนสัญพจน์ในส่วนตัวของกฎ (body of rule) งานวิจัยนี้เรียกสายสัญพจน์ที่มีความหมายอีกชื่อหนึ่งว่า*สายสัญพจน์สมบูรณ์ (valid chain)* โดยนิยามได้ดังนี้

นิยามที่ 2 (สายสัญพจน์สมบูรณ์). เมื่อมองกฎตรรกะลำดับที่หนึ่งหนึ่งๆ เสมือนเป็นกราฟมีทิศทางแล้วเส้นทาง (path) จากโนดรากไปยังโนดใดๆ คือสายสัญพจน์สมบูรณ์ก็ต่อเมื่อเส้นทางนั้นมีคุณสมบัติใดคุณสมบัติหนึ่งต่อไปนี้ (1) เส้นทางนั้นมีวงวน (loop) เกิดขึ้น หรือ (2) เส้นทางนั้นมีโนดที่เป็นโนดใบ (leaf node) อยู่ในเส้นทาง □

โดยโนดใบในงานวิจัยนี้หมายถึงโนดที่ไม่มีเส้นเชื่อมออกจากตัวมันเอง เส้นทางใดๆ ที่สามารถสร้างได้จากรากไปยังโนดหนึ่งๆ จะเป็นสายสัญพจน์สมบูรณ์หรือมีความหมายเสมอ ทั้งนี้เนื่องจากถ้ามีตัวแปรที่ไม่มีความสัมพันธ์กับตัวแปรอื่น โนดที่แทนตัวแปรนั้นย่อมไม่มีเส้นเชื่อมไปยังโนดอื่นๆ นั่นเอง โดยเงื่อนไข (1) และ (2) ถูกสร้างขึ้นเพื่อต้องการให้สายสัญพจน์แต่ละสายที่สมบรูณ์มีข้อมูลที่ครบถ้วนจากกฎตรรกะลำดับที่หนึ่งเดิมให้มากที่สุดเท่าที่เป็นไปได้นั่นเอง (ดู [Kijsirikul et al., 2001] เพิ่มเติม)

อย่างไรก็ตามในนิยามที่ 1 ได้กำหนดว่าลักษณะสำคัญทั้งหมดจะเป็นโนดพ่อแม่ของโนดที่ต้องการทำนาย ซึ่งหมายความว่าต้องมีการคำนวณค่าความน่าจะเป็นภายหลังทั้งหมดทุกรูปแบบที่เป็นไปได้ของลักษณะสำคัญหรือโนดพ่อแม่ทั้งหมดในซีพีทีของโนดที่ต้องการทำนาย ดังนั้นถ้าเลือกสายสัญพจน์ทุกๆ สายที่สมบูรณ์เป็นลักษณะสำคัญของโนดที่ต้องการทำนายอาจทำให้เกิดความซ้ำซ้อน (redundant) ได้โดยไม่จำเป็น ตัวอย่างเช่นจากรูปที่ 12 `'b(A,B,C), e(C,E), f(C,F)'` เป็นสายสัญพจน์ที่สมบูรณ์แต่ซ้ำซ้อน เนื่องจากสามารถเกิดจากการรวมกัน (combination) ของสายสัญพจน์ที่สมบูรณ์สองสายคือ `'b(A,B,C), e(C,E)'` กับ `'b(A,B,C),f(C,F)'` ดังนั้นในการสร้างลักษณะสำคัญ MCAFEE จะสร้างเฉพาะสายสัญพจน์สมบูรณ์*และไม่ซ้ำซ้อน*เท่านั้นโดยจะเรียกสายสัญพจน์ประเภทนี้ว่า*สายสัญพจน์สมบูรณ์เล็กสุด (minimal valid chain)* และนิยามดังนี้

นิยามที่ 3 (สายสัญพจน์สมบูรณ์เล็กสุด). สายสัญพจน์สมบูรณ์ r ใดๆ เล็กที่สุดก็ต่อเมื่อไม่มีสายสัญพจน์สมบูรณ์อื่นๆ r' ที่มีคุณสมบัติ $r' \subseteq r$ □

ตัวอย่างของสายสัญพจน์สมบูรณ์เล็กสุดหรือลักษณะสำคัญจากรูปที่ 12 ที่สร้างโดย MCAFEE แสดงได้ในรูปที่ 13 และสายสัญพจน์ที่ถูกต้องที่เล็กที่สุดทั้งหมดของตรรกะลำดับที่หนึ่งทั้งสามกฎในตัวอย่างของหัวข้อที่ 2.1.1 แสดงในรูปที่ 14

```
f1(A) :- a(A).
f2(A,B,C,D) :- b(A,B,C),c(B,D),d(D).
f3(A,B,C,E) :- b(A,B,C),e(C,E).
f4(A,B,C,F) :- b(A,B,C),f(C,F).
```

รูปที่ 13 ลักษณะสำคัญจากรูปที่ 12 ที่สร้างโดย MCAFEE

```
f1(A) :- genius(A).
f2(A) :- diligent(A).
f3(A) :- artist(A).
f4(A) :- save_money(A).
f5(A,B) :- parent(B,A),rich(B).
f6(A,B) :- parent(B,A),genius(B).
```

รูปที่ 14 ลักษณะสำคัญจากตัวอย่างในหัวข้อที่ 2.1.1

2.1.3 GRASP

หัวข้อนี้นำเสนอ GRASP (GRound substitution AS example Propositionalization) ซึ่งเป็นอัลกอริทึมในการสร้างแถวหรือตัวอย่างหนึ่งตัวของฐานข้อมูลตารางเดี่ยว ทั้งนี้เนื่องจากบางกรณีตัวอย่างของข้อมูลที่มีความสัมพันธ์ต่อกันเมื่อแปลงรูปไปเป็นฐานข้อมูลตารางเดี่ยวตามสดมภ์ที่กำหนดแล้วสามารถเปลี่ยนเป็นแถวใหม่มากกว่าหนึ่งแถวได้ ตัวอย่างเช่น พิจารณาตัวอย่างในหัวข้อที่ 2.1.1 อีกครั้ง สมมุติให้นาย a เป็นตัวอย่างหนึ่งในข้อมูลอินพุตที่มีความสัมพันธ์ต่อกันและให้นาย a มีพ่อบุญธรรมอยู่อีกหนึ่งคน (นาย h) ซึ่งมีคุณสมบัติดังนี้ `parent(h,a).` `genius(h).` จากตัวอย่างนี้เมื่อมีการแปลงข้อมูลให้ไปอยู่ในรูปฐานข้อมูลตารางเดี่ยวแล้วจะได้ตัวอย่างใหม่สองตัวดังแสดงในตารางที่ 8 (ดู `f1-f6` ในรูปที่ 6)

ตารางที่ 8 ข้อมูลของนาย a (หัวข้อที่ 2.1.1) เมื่อแปลงให้อยู่ในรูปฐานข้อมูลตารางเดี่ยว

ตัวอย่าง	การแทนค่า	f1	f2	f3	f4	f5	f6
a	A/a, B/e	1	0	0	1	1	0
	A/a, B/h	1	0	0	1	0	1

ในกรณีแรกนาย a มีพ่อที่รวย (นาย e) ส่วนในกรณีที่สองนาย a มีพ่อที่อัจฉริยะ (นาย h) แต่ว่าไม่ใช่คนเดียวกัน [Fensel et al., 1995] เสนอให้*ใช้การแทนค่าตัวแปรด้วยข้อมูลจริง (ground substitution : GS หรือ variable binding)* ทั้งหมดทุกกรณีเป็นตัวอย่างใหม่ทั้งหมดในฐานข้อมูลตารางเดี่ยว (โดยถือว่าตัวอย่างใหม่ทั้งหมดที่เกิดจากตัวอย่างเดิมนั้นไม่มีความข้องเกี่ยวต่อกันและกันเลย) ดังเช่นการแทนค่าทั้งสองกรณีของนาย a ในตารางที่ 8 เกิดเป็นตัวอย่างใหม่สองตัว วิธีนี้มีข้อดีสองข้อ ข้อแรกคือการแปลงข้อมูลประเภทนี้ทำให้เกิดข้อมูลใหม่เป็นฐานข้อมูลตารางเดี่ยวที่แท้จริงทำให้สามารถใช้ระบบเรียนรู้ต้นไม้ตัดสินใจหรือตัวเรียนรู้ข่ายงานเบส์ได้ทันทีไม่ต้องปรับแต่งข้อมูลอีกซึ่งตรงข้ามกับงานวิจัยบางงานที่เสนอให้ตัวอย่างใหม่ทั้งหมดที่เกิดจากตัวอย่างเก่าเดียวกันมีความสัมพันธ์ต่อกันที่เรียกว่า*มัลติอินแสตนซ์ (multi-instance propositionalization)* [Kramer et al., 2001] ซึ่งทำให้ต้องพัฒนาตัวจำแนกประเภทข้อมูลใหม่ทั้งหมด (ไม่สามารถใช้ตัวเรียนรู้ข่ายงาน

เบส์หรือระบบเรียนรู้ต้นไม้ตัดสินใจที่มีอยู่แล้วได้) และข้อดีข้อที่สองคือความสัมพันธ์ระหว่างโนดที่สร้างได้จากวิธีนี้จะมีความสมบูรณ์เข้มมากกว่า (strong completeness) พิจารณาตารางที่ 9 ด้านล่างนี้

ตารางที่ 9 ตัวอย่างของความสมบูรณ์เข้มมากกว่า

ตัวอย่าง	class	การแทนค่า	f1	f2	f3	f4
E1	+	θ_{E11}	1	0	0	1
		θ_{E12}	0	1	0	0
E2	+	θ_{E21}	0	1	0	0
		θ_{E22}	1	0	0	1
E3	+	θ_{E31}	0	0	1	1
		θ_{E32}	1	0	1	0
E4	–	θ_{E41}	0	0	1	0
		θ_{E42}	1	0	0	0
E5	–	θ_{E51}	1	0	1	0
		θ_{E52}	1	0	0	0

เนื่องจากความน่าจะเป็นภายหลัง p(+|f2) และ p(+|f4) มีค่า 1.0 ในขณะที่ p(+) คำนวณได้ 0.6 ฉะนั้นโนด `class` จึงขึ้นกับ (depend on) โนด `f2` และ `f4` แต่ถ้าเลือกเพียงการแทนค่าเดียว ในตัวอย่างเดิมเป็นตัวอย่างใหม่จะทำให้คำนวณได้ว่าโนด `class` ขึ้นกับโนด `f2` หรือ `f4` โนดใดโนดหนึ่งเท่านั้นซึ่งไม่ครบถ้วน

อย่างไรก็ตามการเลือกการแทนค่าทุกๆ แบบเป็นตัวอย่างใหม่อาจทำให้เกิดปัญหาสองอย่าง ปัญหาข้อแรกคือจำนวนตัวอย่างใหม่ที่ได้อาจมีจำนวนมหาศาลทำให้หน่วยความจำในการเก็บฐานข้อมูลตารางเดี่ยวไม่เพียงพอ ปัญหาข้อที่สองคือวิธีนี้อาจทำให้เกิดความผิดพลาดในการเรียนรู้ได้เช่นกรณีในตารางที่ 10

ตารางที่ 10 กรณีตัวอย่างเดิม 7 ตัวถูกเปลี่ยนเป็นตัวอย่างใหม่ 10 ตัว

ตัวอย่าง	class	การแทนค่า	f1	f2	f3	f4
E1	+	θ_{E11}	1	0	0	1
E2	+	θ_{E21}	1	0	0	1
E3	+	θ_{E31}	0	0	1	1
E4	+	θ_{E41}	0	0	0	1
E5	–	θ_{E51}	1	0	1	0
E6	–	θ_{E61}	1	0	1	0
E7	–	θ_{E71}	0	0	0	1
		θ_{E72}	0	0	0	1
		θ_{E73}	0	0	0	1
		θ_{E74}	0	0	0	1

จากตารางที่ 10 จะเห็นได้ว่าความน่าจะเป็นภายหลังของ p(+|f4) คำนวณได้ 0.5 ในขณะที่ p(+) คำนวณได้ 0.57 เนื่องจากค่าความน่าจะเป็นทั้งสองค่าใกล้เคียงกันมากทำให้ตัวเรียนรู้ข่ายงานเบส์สรุปว่าโนด `class` เป็นอิสระต่อโนด `f4` ซึ่งไม่ถูกต้องเนื่องจากในความเป็นจริงมีตัวอย่างลบเพียงตัวอย่างเดียว (`E7`) ที่ขัดแย้งกับตัวอย่างบวกสี่ตัวอย่าง (หรือ p(+|f4) = 0.8)

GRASP แก้ปัญหาทั้งสองข้อข้างต้นโดยการเลือกการแทนค่าเพียงบางแบบเป็นตัวอย่างใหม่เท่านั้นโดยจะเลือกเฉพาะ *GS ที่ไม่ซ้ำ (non-duplicate)* และ*มีการแทนค่าจำเพาะมากสุด (maximal specific binding – MSB)* โดยการแทนค่าจำเพาะมากสุดมีนิยามดังนี้

นิยามที่ 4 (การแทนค่าจำเพาะมากสุด). เมื่อกำหนด ω คือการแทนค่าใดๆ กำหนดให้ $f(\omega)$ คือเซตของลักษณะสำคัญทั้งหมดที่เป็นจริงของ ω พิจารณาการแทนค่า θ ของแต่ละตัวอย่างเดิม e θ เป็นการแทนค่าจำเพาะมากสุดก็ต่อเมื่อไม่มี α ซึ่งเป็นการแทนค่าอื่นของ e ที่ $f(\theta) \subseteq f(\alpha)$ □

พิจารณาตารางที่ 11 และจากนิยามที่ 4 จะได้ว่าการแทนค่าจำเพาะมากสุดที่ไม่ซ้ำจากตารางมีสี่กรณีคือ θ_{E11} θ_{E13} θ_{E21} และ θ_{E22} สังเกตว่า θ_{E14} ก็เป็นการแทนค่าจำเพาะมากสุดแต่ซ้ำกับ θ_{E11} เช่นเดียวกับ θ_{E23} ซึ่งซ้ำกับ θ_{E22} ฉะนั้นตัวอย่างใหม่ที่สร้างโดย GRASP มีเพียงการแทนค่าสี่กรณีข้างต้นเท่านั้น

ตารางที่ 11 กรณีตัวอย่างเดิมสองตัวเปลี่ยนเป็นตัวอย่างใหม่ 8 ตัว

ตัวอย่าง	class	การแทนค่า	f1	f2	f3	f4
E1	+	θ_{E11}	1	1	0	1
		θ_{E12}	0	1	0	1
		θ_{E13}	0	1	1	0
		θ_{E14}	1	1	0	1
E2	−	θ_{E21}	1	1	0	1
		θ_{E22}	0	0	1	0
		θ_{E23}	0	0	1	0
		θ_{E24}	1	1	0	0

เนื่องจากระบบเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งที่เสนอในงานวิจัยนี้ใช้การแทนค่าจำเพาะมากสุดที่ไม่ซ้ำเป็นตัวอย่างใหม่ ในการทำนายคุณสมบัติของตัวอย่างที่ไม่เคยพบมาก่อนด้วยข่ายงานเบส์ลำดับที่หนึ่ง จึงใช้การแทนค่าจำเพาะมากสุดที่ไม่ซ้ำของตัวอย่างนั้นๆ ในการทำนายด้วยเช่นกัน อย่างไรก็ตามการทำนายประเภทของข้อมูลที่มีความสัมพันธ์ต่อกันโดยใช้วิธีนี้ อาจมีปัญหาเนื่องจากการแทนค่าจำเพาะมากสุดที่ไม่ซ้ำหรือตัวอย่างใหม่ทั้งหมดที่สร้างจากตัวอย่างเดิมหนึ่งตัวอาจถูกจัดให้อยู่ในประเภทหรือกลุ่มที่ไม่เหมือนกัน โดยปกติวิธีการทำนายประเภทของข้อมูลในปัญหาแบบนี้ขึ้นอยู่กับ*ไบแอส (bias)* [Mitchell, 1997] ของปัญหาแต่ละปัญหาซึ่งมักไม่เหมือนกันเช่น ผู้ใช้อาจต้องการให้นำค่าความน่าจะเป็นภายหลังที่มากที่สุดของการแทนค่าจำเพาะมากสุดที่ไม่ซ้ำมาตัดสินประเภทหรือกลุ่ม หรือตัดสินโดยการใช้เสียงส่วนมาก (majority vote) ของการแทนค่าจำเพาะมากสุดที่ไม่ซ้ำทั้งหมด หรือเฉลี่ยค่าความน่าจะเป็นภายหลังของการแทนค่าจำเพาะมากสุดที่ไม่ซ้ำทั้งหมดก่อนแล้วค่อยตัดสินประเภทจากค่าความน่าจะเป็นเฉลี่ยที่ได้ ดังนั้นเพื่อประสิทธิภาพสูงสุดในระบบต้นแบบของงานนี้จึงอนุญาตให้ผู้ใช้กำหนดไบแอสที่เหมาะสมกับปัญหาได้เอง

2.1.4 ผลการทดลอง

การทดลองใช้ชุดข้อมูล (dataset) *ของความสามารถในการก่อกลายพันธุ์ของโมเลกุล (mutagenesis)* [Srinivasan & King, 1999] ซึ่งเป็นชุดข้อมูลแบบมีความสัมพันธ์ต่อกันที่เป็นมาตรฐานในการทดสอบความสามารถของระบบไอแอลพี โดยความรู้ภูมิหลังในการทดลองนี้ได้จากผู้เชี่ยวชาญทางด้านโมเลกุลโดยตรง และตัวอย่างแต่ละตัวประกอบไปด้วยคุณสมบัติต่างๆ ของโมเลกุลหนึ่งชนิด จุดมุ่งหมายของปัญหานี้คือการแบ่งประเภทของโมเลกุลที่ไม่เคยพบมาก่อนว่าสามารถก่อกลายพันธุ์ (mutagenic) ได้หรือไม่

ในการทดลองนี้ เราได้เลือกใช้ระบบ PROGOL (เวอร์ชัน CProgol4.2) [Muggleton, 1995] ซึ่งเป็นระบบไอแอลพีที่ได้รับการยอมรับกันอย่างแพร่หลาย เพื่อสร้างตรรกะลำดับที่หนึ่งและใช้โปรแกรม WinMine [Chickering, 2002] เป็นตัวเรียนรู้ข่ายงานเบส์เพื่อที่จะเรียนซีพีทีและโครงสร้างส่วนที่เหลือของข่ายงานเบส์ลำดับที่หนึ่ง ในขั้นตอนการทำนายการทดลองครั้งนี้ใช้เทคนิค 3-fold cross-validation และใช้ค่าความน่าจะเป็นภายหลังของการแทนค่าจำเพาะมากสุดที่ไม่ซ้ำที่มีค่าสูงที่สุดในบรรดาการแทนค่าจำเพาะมากสุดที่ไม่

ซ้ำทั้งหมดเพื่อทำนายประเภทข้อมูลของตัวอย่างเดิมดังที่ได้อธิบายไปแล้วในตอนท้ายของหัวข้อที่ 2.1.3 นอกจากนี้การทดลองนี้ยังได้เพิ่มสัญญาณรบกวนอย่างสุ่มจำนวน 10% และ 15% ในชุดข้อมูลนี้อีกด้วยเพื่อทดสอบประสิทธิภาพความทนทานต่อสัญญาณรบกวนของข่ายงานเบส์ลำดับที่หนึ่ง เนื่องจากระบบ PROGOL อนุญาตให้ผู้ใช้ปรับเปลี่ยนตัวเลือก (option) เพื่อจัดการกับสัญญาณรบกวนได้ ในการทดลองครั้งนี้จึงได้ลองปรับค่าตัวเลือกต่างๆ ของ PROGOL แล้วนำผลที่ได้มาเปรียบเทียบกับผลการทดลองจากข่ายงานเบส์ลำดับที่หนึ่ง (PROGOL+FOBN ในตาราง) ผลการทดลองที่ได้แสดงในตารางที่ 12

ตารางที่ 12 เปอร์เซ็นต์ความถูกต้องในปัญหาการก่อกลายพันธุ์

ระดับสัญญาณรบกวนในชุดข้อมูล	PROGOL 0% noise setting	PROGOL 5% noise setting	PROGOL 10% noise setting	PROGOL 15% noise setting	PROGOL +FOBN
0%	**84.58**	82.99	77.14	77.14	84.34
10%	64.23	65.42	69.72	71.29	**78.67**
15%	60.56	59.02	61.54	65.31	**74.33**

"x% noise setting" หมายถึงการปรับตัวเลือกให้ PROGOL ปรับระบบเพื่อเรียนรู้กฎที่สามารถมีสัญญาณรบกวนได้ x% จะเห็นได้ว่าเมื่อไม่มีสัญญาณรบกวนภายในชุดข้อมูล ผลการทำงานของข่ายงานเบส์ลำดับที่หนึ่งมีประสิทธิภาพใกล้เคียงกับ PROGOL แต่ว่าผลการทำงานของข่ายงานเบส์ลำดับที่หนึ่งในชุดข้อมูลที่มีสัญญาณรบกวนให้ความถูกต้องมากกว่า PROGOL ในทุกกรณีเป็นอย่างมาก ความถูกต้องที่สูงกว่านี้เมื่อวิเคราะห์แล้วเกิดจากสาเหตุใหญ่สองประการคือ (1) ลักษณะสำคัญที่ได้จาก MCAFEE ช่วยให้เทคนิคการตรงบางส่วนของกฎมีประสิทธิภาพอย่างมาก โดย Kijsirikul et al. [2001] ได้แสดงว่าในปัญหาที่มีสัญญาณรบกวนและปัญหาการจำแนกประเภทข้อมูลที่เป็นไปได้มีมากกว่าสองประเภท (multi-class) ด้วยลักษณะสำคัญเหล่านี้เพียงลำพัง (โดยไม่จำเป็นต้องใช้เทคนิคอื่นเพิ่มเติมเช่น ความน่าจะเป็นหรือข่ายงานประสาทเทียม) สามารถได้รับความถูกต้องมากกว่าการใช้ตรรกะลำดับที่หนึ่งตรงๆ และ (2) การใช้ทฤษฎีความน่าจะเป็นของข่ายงานเบส์ลำดับที่หนึ่ง ช่วยให้การใช้งานลักษณะสำคัญมีความยืดหยุ่นมากยิ่งขึ้นทำให้จำแนกประเภทข้อมูลที่มีสัญญาณรบกวนได้ดี

2.2 นิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่ง

เนื้อหาในส่วนนี้จะอธิบายถึงการนำนิวรอลเน็ตเวิร์กมาประยุกต์เข้ากับการโปรแกรมตรรกะเชิงอุปนัยโดยทำให้สามารถรับอินพุตในรูปแบบของตรรกะอันดับที่หนึ่งมาทำการเรียนรู้ได้โดยตรง ไม่ต้องใช้ระบบไอแอลพีมาช่วยในการเรียนรู้ วิธีการที่นำเสนอนี้สามารถจัดการกับข้อจำกัดของระบบไอแอลพีที่ไม่ทนทานต่อสัญญาณรบกวนได้ดียิ่งขึ้นและให้ผลที่แม่นยำมากขึ้น โดยเรียกวิธีการที่พัฒนาขึ้นนี้ว่า นิวรอลเน็ตเวิร์กตรรกะอันดับที่หนึ่ง (First-Order Neural Networks: FONNs) การเรียนรู้ของนิวรอลเน็ตเวิร์กตรรกะอันดับที่หนึ่งแบ่งออกเป็นส่วนๆ คือ การสร้างโครงสร้างของนิวรอลเน็ตเวิร์กตรรกะอันดับที่หนึ่ง การสร้างอินพุตของนิวรอลเน็ตเวิร์กตรรกะอันดับที่หนึ่งโดยหลักการเรียนรู้แบบหลายตัวอย่างย่อย (Multiple-Instance Learning: MIL) [Chevaleyre & Zucker, 2001; Huang et al., 2003] และการปรับค่าน้ำหนักของนิวรอลเน็ตเวิร์กอันดับที่หนึ่งดังจะกล่าวต่อไปนี้

2.2.1 โครงสร้างของนิวรอลเน็ตเวิร์กตรรกะอันดับที่หนึ่ง

เนื่องจากนิวรอลเน็ตเวิร์กตรรกะอันดับที่หนึ่งนั้นเป็นระบบการเรียนรู้ที่เราพัฒนามาจากนิวรอลเน็ตเวิร์ก ดังนั้นตัวโครงสร้างก็จะมีพื้นฐานมาจากโครงสร้างของนิวรอลเน็ตเวิร์ก แต่ประยุกต์ให้รับตัวอย่างและความรู้ภูมิหลังในรูปแบบของตรรกะได้ [Garcez et al., 2002] โครงสร้างของนิวรอลเน็ตเวิร์กอันดับที่หนึ่งจะแบ่งออกเป็น 3 ชั้นด้วยกัน คือ ชั้นนำเข้า (input layer) ชั้นซ่อน (hidden layer) และชั้นผลลัพธ์ (output layer) ในแต่ละชั้นก็จะมีความหมายและหน้าที่ดังต่อไปนี้

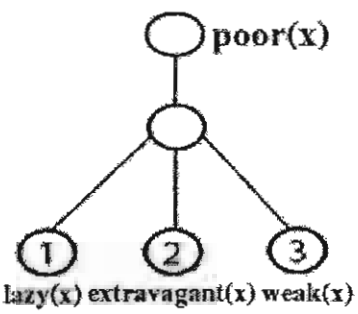
1) ชั้นผลลัพธ์ เป็นชั้นที่แสดงถึงแนวคิดที่ต้องการให้ระบบทำการเรียนรู้ โดยชั้นนี้จะคำนวณหาผลลัพธ์สุดท้ายของเน็ตเวิร์ก จำนวนนิวรอนในชั้นผลลัพธ์จะขึ้นอยู่กับจำนวนแนวคิดทั้งหมดที่ต้องการเรียน
2) ชั้นซ่อน เป็นชั้นที่เชื่อมต่อระหว่างชั้นนำเข้าและชั้นผลลัพธ์ ช่วยเพิ่มความสามารถในการเรียนกฎที่มีความซับซ้อนได้ จำนวนนิวรอนในชั้นนี้มักจะกำหนดโดยดูจากความซับซ้อนของแนวคิดที่ต้องการเรียนรู้
3) ชั้นนำเข้า เป็นชั้นที่แสดงถึงสัญพจน์ (predicate) ต่างๆที่จะนำมาใช้เป็นตัวแทนของกฎหรือแนวคิดในการเรียน โดยจะเป็นชั้นแรกในการรับข้อมูลก่อนที่จะผ่านไปยังชั้นต่อๆไป จำนวนนิวรอนในชั้นนี้จะขึ้นอยู่กับจำนวนสัญพจน์จากความรู้ภูมิหลัง

ตัวอย่างเช่น การเรียนแนวคิดของ `poor(x)` ซึ่งมีอินพุตที่ใช้ในการเรียนเป็นดังรูปที่ 15

- ความรู้ภูมิหลัง : `lazy(John), extravagant(John), lazy(Peter), weak(Peter), extravagant(Bob), weak(Bob)`
- ตัวอย่างบวก : `poor(John)`
- ตัวอย่างลบ : `poor(Peter), poor(Bob)`

รูปที่ 15 อินพุตสำหรับการเรียนแนวคิด `poor(x)`

อินพุตที่ได้รับประกอบไปด้วยตัวอย่างบวก 1 ตัว คือ `poor(John)` ซึ่ง `John` เป็นคนเกียจคร้าน (`lazy(John)`)และฟุ่มเฟือย (`extravagant(John)`) มีตัวอย่างลบ 2 ตัว คือ `poor(Peter)` และ `poor(Bob)` โดยที่ `Peter` เป็นคนเกียจคร้าน (`lazy(Peter)`) และไม่แข็งแรง (`weak(Peter)`) `Bob` เป็นคนฟุ่มเฟือย (`extravagant(Bob)`) และไม่แข็งแรง (`weak(Bob)`) อินพุตที่ได้รับมีจำนวนสัญพจน์ที่ปรากฏในความรู้ภูมิหลังอยู่ 3 ตัวด้วยกัน คือ `lazy(x)`, `extravagant(x)` และ `weak(x)` ดังนั้นในชั้นนำเข้าจะมีอยู่ 3 นิวรอน เพื่อแทนลักษณะทั้ง 3 สัญพจน์นั้น ส่วนจำนวนนิวรอนในชั้นผลลัพธ์จะมีเพียง 1 นิวรอนเท่านั้น เนื่องจากกฎที่ต้องการเรียนมีเพียง 1 กฎเท่านั้น คือ `poor(x)` ดังแสดงในรูปที่ 16



รูปที่ 16 ลักษณะโครงสร้างของเน็ตเวิร์กโดยกำหนดให้ในชั้นซ่อนมี 1 นิวรอน

2.2.2 อินพุตของนิวรอลเน็ตเวิร์กตรรกะอันดับที่หนึ่งโดยหลักการเรียนรู้แบบหลายตัวอย่างย่อย

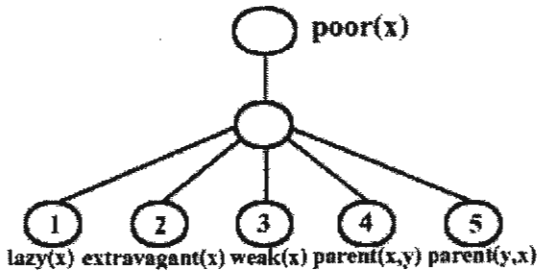
อินพุตของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

โดยปกติแล้วอินพุตของนิวรอลเน็ตเวิร์กจะอยู่ในรูปของจำนวนจริง แต่ว่าอินพุตของระบบไอแอลพีจะอยู่ในรูปของตรรกะ (ความรู้ภูมิหลังและตัวอย่าง) ดังนั้นจึงต้องเปลี่ยนอินพุตที่เป็นลักษณะของตรรกะไปเป็นอินพุตที่สามารถเรียนรู้ได้โดยนิวรอลเน็ตเวิร์กเสียก่อน การเปลี่ยนจะทำทีละ 1 ตัวอย่าง ทีละ 1 นิวรอน โดยตัวอย่าง 1 ตัวจะถูกเปลี่ยนให้เป็นค่า 1 เพื่อเป็นอินพุตของนิวรอน ถ้าแทนค่าตัวแปรในสัญพจน์ประจำนิวรอนด้วยค่าคงที่ที่ปรากฏในตัวอย่างแล้วมีค่าความจริงเป็นจริงในความรู้ภูมิหลัง แต่ถ้าไม่เป็นจริงในความรู้ภูมิหลังก็จะกำหนดให้อินพุตของนิวรอนมีค่าเป็น 0 ตัวอย่างเช่น จากตัวอย่างอินพุตในรูปที่ 15 เมื่อเปลี่ยนตัวอย่าง `poor(John)` ไปเป็นอินพุตเน็ตเวิร์กจะได้ค่าอินพุตของนิวรอน `lazy(x)`, `extravagant(x)` และ `weak(x)` เป็น 1, 1 และ 0 ตามลำดับ เนื่องจากเมื่อแทนตัวแปร `x` ในสัญพจน์ของแต่ละนิวรอนด้วย `John` ซึ่งเป็นค่าคงที่ของตัวอย่าง `poor(John)` แล้ว จะได้เป็น `lazy(John)`, `extravagant(John)` และ `weak(John)` ซึ่ง `lazy(John)` และ `extravagant(John)` เป็นจริง ในความรู้ภูมิหลังจึงให้ค่าอินพุตของนิวรอนเป็น 1 ในขณะที่ `weak(John)` ไม่เป็นจริงในความรู้ภูมิหลัง จึงมีค่าเป็น 0 สำหรับตัวอย่าง `poor(Peter)` และ `poor(Bob)` จะได้ค่าอินพุตของนิวรอนเป็น 1,0,1 และ 0,1,1 ตามลำดับ

อย่างไรก็ตามเนื่องจากแนวคิดที่ได้จากระบบไอแอลพีอาจมีการสร้างตัวแปรใหม่ขึ้นในสัญพจน์โดยที่ตัวแปรนั้นไม่ปรากฏอยู่ในส่วนหัวของกฎ ตัวแปรใหม่ที่สร้างขึ้นนั้นเพื่อนำมาใช้อธิบายความสัมพันธ์ที่เกี่ยวข้อง (relation) กันของแต่ละสัญพจน์ อย่างเช่น `poor(x) ← parent(y,x), poor(y), lazy(x)` ซึ่งหมายความว่าถ้า `y` เป็นผู้ปกครองของ `x` โดยที่ `y` เป็นคนจน และ `x` เป็นคนเกียจคร้าน แล้ว `x` จะเป็นคนจน ในกฎนี้มีการสร้างตัวแปร `y` ขึ้นมาเพื่อใช้อธิบายถึงบุคคลอื่นซึ่งไม่ใช่บุคคลที่ปรากฏอยู่ในส่วนหัวของกฎ (`poor(x)`) โดยใช้สัญพจน์ `parent(y,x)` เป็นตัวเชื่อมความสัมพันธ์จาก `x` ไปยัง `y` แต่ว่าการสร้างตัวแปรขึ้นใหม่ในลักษณะนี้จะทำให้การสร้างอินพุตสำหรับนิวรอนเกิดปัญหาความไม่แน่นอนขึ้น เช่น ถ้าความรู้ภูมิหลังและตัวอย่างที่รับเข้ามามีลักษณะเป็นดังรูปที่ 17

รูปที่ 17 อินพุตที่เพิ่มสัญพจน์ `parent(x,y)` ในความรู้ภูมิหลัง

ในกรณีนี้โครงสร้างของเน็ตเวิร์กจะมีความซับซ้อนมากขึ้นเนื่องจากในความรู้ภูมิหลังมีสัญพจน์ `parent(x,y)` เพิ่มเข้ามา และเนื่องจากสัญพจน์นี้มีอาร์กิวเมนต์ 2 ตัวและเราไม่สามารถรู้ได้ว่าอาร์กิวเมนต์ 2 ตัวนี้ควรมีการวางตำแหน่งอย่างไร จึงได้ทำการสร้างนิวรอนขึ้นมาสำหรับทั้ง 2 กรณี คือ `parent(x,y)` และ `parent(y,x)` ทำให้จำนวนนิวรอนในชั้นนำเข้าเพิ่มขึ้นอีก 2 นิวรอน จำนวนนิวรอนทั้งหมดในชั้นนำเข้าจึงเป็น 5 นิวรอน ดังรูปที่ 18



รูปที่ 18 เน็ตเวิร์กที่เพิ่มนิวรอน `parent(x,y)` และ `parent(y,x)` ขึ้นมาจากเน็ตเวิร์กในรูปที่ 16

แต่ว่านิวรอนที่เพิ่มขึ้นมานี้มีตัวแปรใหม่อยู่ด้วย ทำให้มีปัญหาในการกำหนดค่าอินพุตให้กับนิวรอน เช่น ตัวอย่าง `poor(John)` เมื่อจะกำหนดค่าอินพุตให้กับนิวรอน `parent(y,x)` ก็จะแทนที่ตัวแปร `x` ด้วยค่าคงที่ `John` ส่วนตัวแปร `y` ซึ่งไม่มีการกำหนดค่ามาจากตัวอย่าง ทำให้ไม่สามารถระบุได้ว่าต้องแทนตัวแปร `y` ด้วยค่าคงที่ใด และการแทนด้วยค่าคงที่แต่ละแบบนั้นก็จะให้ค่าอินพุตของนิวรอนแตกต่างกันดังแสดงในตารางที่ 13 ส่วนกรณีของ `parent(x,y)` ไม่เกิดปัญหาเนื่องจากเมื่อแทนค่าตัวแปร `x` ด้วย `John` แล้วไม่ว่าจะแทนค่าตัวแปร `y` ด้วยค่าคงที่ใด ก็ไม่เป็นจริงในความรู้ภูมิหลัง ทำให้ค่าอินพุตของนิวรอน `parent(x,y)` เป็น 0 ในทุกกรณี

ตารางที่ 13 ค่าอินพุตของนิวรอน `parent(y,x)` เมื่อแทนตัวแปร `y` ด้วยค่าคงที่ต่างๆ

parent(y,x)	**ค่าอินพุตของนิวรอน**
แทน `x` ด้วย `John` แทน `y` ด้วย `John`	0
แทน `x` ด้วย `John` แทน `y` ด้วย `Peter`	1
แทน `x` ด้วย `John` แทน `y` ด้วย `Bob`	0

จากตัวอย่างข้างต้นแสดงให้เห็นว่าการกำหนดค่าอินพุตให้กับนิวรอนในกรณีที่สัญพจน์ประจำนิวรอนนั้นมีตัวแปรที่ใช้แสดงความสัมพันธ์กับสัญพจน์อื่น โดยตัวแปรนั้นไม่ปรากฏอยู่ในส่วนหัวของกฎด้วย ในบางกรณีจะทำให้ไม่สามารถกำหนดค่าที่แน่นอนให้กับนิวรอนได้ เนื่องจากไม่ทราบว่าควรแทนค่าให้กับตัวแปรที่แสดงความสัมพันธ์ด้วยค่าใด จึงได้นำหลักการของการเรียนรู้แบบหลายตัวอย่างย่อย (Multiple-Instance Learning: MIL) [Chevaleyre & Zucker, 2001; Huang et al., 2003] มาประยุกต์เพื่อทำการสร้างอินพุตให้กับนิวรอลเน็ตเวิร์กอันดับที่หนึ่งดังจะกล่าวในหัวข้อต่อไปนี้

หลักการเรียนรู้แบบหลายตัวอย่างย่อย

ในปัจจุบันการเรียนรู้สามารถแบ่งได้เป็น 3 ประเภทใหญ่ๆด้วยกัน คือ การเรียนรู้แบบสอน (supervised learning) การเรียนรู้แบบไม่สอน (unsupervised learning) และการเรียนแบบเสริมความแกร่ง (reinforcement learning) แต่ว่าในบางปัญหานั้นก็ไม่สามารถที่จะเรียนได้อย่างมีประสิทธิภาพด้วยการเรียนรู้ที่มีอยู่ทั้ง 3 ประเภทนี้ ตัวอย่างของปัญหาในลักษณะนี้จะมีลักษณะที่มีความคลุมเครืออยู่ (ambiguous example) ไม่สามารถระบุได้ว่าแต่ละตัวอย่างนั้นเป็นตัวอย่างบวกหรือตัวอย่างลบ บอกได้แค่เพียงว่าในกลุ่มตัวอย่างนั้นมีตัวอย่างบวกอยู่หรือไม่ การเรียนรู้แบบหลายตัวอย่างย่อยได้ถูกนำเสนอเพื่อนำมาใช้แก้ปัญหาในลักษณะนี้

การเรียนรู้แบบหลายตัวอย่างย่อยจะนิยามตัวอย่างในการเรียนรู้เป็นถุง (bag) $\{B_1, B_2, \ldots, B_n\}$ โดยที่ n เป็นจำนวนของถุง แต่ละถุง (B_i) จะประกอบไปด้วยตัวอย่างย่อย (instance) m_i ตัว $\{B_{i1}, B_{i2}, \ldots, B_{imi}\}$ โดยที่

ถุงหนึ่งๆ จะถูกกำหนดให้เป็นถุงตัวอย่างบวก (positive bag) ก็ต่อเมื่อในถุงนั้นมีตัวอย่างย่อยอย่างน้อย 1 ตัวที่เป็นบวก และจะกำหนดให้เป็นถุงตัวอย่างลบ (negative bag) ก็ต่อเมื่อตัวอย่างย่อยทุกตัวในถุงนั้นเป็นตัวอย่างลบทั้งหมด โดยที่ถุงตัวอย่างบวกจะกำหนดให้มีค่าเป้าหมาย (label) เป็น 1 ส่วนถุงตัวอย่างลบจะมีค่าเป็น 0

เราจะนำหลักการเรียนรู้แบบหลายตัวอย่างย่อยมาใช้เพื่อแปลงตัวอย่างไปเป็นอินพุตให้กับเน็ตเวิร์ก โดยจะให้ตัวอย่างบวก 1 ตัว แทนด้วยถุงตัวอย่างบวก 1 ถุง และแทนตัวอย่างลบ 1 ตัวด้วยถุงตัวอย่างลบ 1 ถุง (ในกรณีที่เป็นการเรียนตัวอย่างแบบหลายประเภทจะถือว่าทุกถุงเป็นถุงตัวอย่างบวกของประเภทนั้นๆ) ในถุงจะประกอบไปด้วยตัวอย่างย่อยซึ่งแต่ละตัวมาจากการแทนค่าตัวแปรด้วยค่าคงที่แต่ละแบบ อย่างเช่นจากตัวอย่างอินพุตตามรูปที่ 3 ในการเรียนแนวคิดของ `poor(x)` จะมีถุงตัวอย่างบวกทั้งหมด 2 ถุง คือ `poor(John)` และ `poor(Peter)` ถุงตัวอย่างลบ 1 ถุง คือ `poor(Bob)` และในแต่ละถุงจะประกอบไปด้วยตัวอย่างย่อยทั้งหมด 3 ตัว จากการแทนค่าตัวแปร `y` ด้วยค่า `John`, `Peter` และ `Bob` ดังแสดงตัวอย่างในตารางที่ 14

ตารางที่ 14 การแปลงตัวอย่างของ `poor(John)` ไปเป็นอินพุตให้กับนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

ถุงตัวอย่าง ของ `poor(John)`	`lazy(x)`	`extravagant(x)`	`weak(x)`	`parent(x,y)`	`parent(y,x)`
แทน `x` ด้วย `John` แทน `y` ด้วย `John`	1	0	0	0	0
แทน `x` ด้วย `John` แทน `y` ด้วย `Peter`	1	0	0	0	1
แทน `x` ด้วย `John` แทน `y` ด้วย `Bob`	1	0	0	0	0

จากนั้นจึงนำถุงตัวอย่างที่ได้ไปใส่เป็นอินพุตให้กับเน็ตเวิร์กและทำการเรียนรู้โดยอัลกอริทึมแบ็คพรอพาเกชัน (Backpropagation Algorithm) สำหรับการเรียนรู้แบบหลายตัวอย่างย่อย [Wattuya et al., 2003; Zhou & Zhang, 2002]

2.2.3 การปรับค่าน้ำหนักของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

การเรียนรู้ของนิวรอลเน็ตเวิร์กนั้นเป็นการปรับค่าน้ำหนักของเส้นเชื่อมให้สามารถจำแนกตัวอย่างได้อย่างถูกต้อง การปรับค่าน้ำหนักจะทำทั้งส่วนที่เชื่อมระหว่างชั้นนำเข้ากับชั้นซ่อน และระหว่างชั้นซ่อนกับชั้นผลลัพธ์ การปรับค่าน้ำหนักจะนำหลักการของแบ็กพรอพาเกชัน [Wattuya et al., 2003; Zhou & Zhang, 2002] มาใช้ หลักการของแบ็กพรอพาเกชันเป็นการปรับค่าน้ำหนักโดยมีจุดมุ่งหมายที่จะทำให้ค่าความผิดพลาดรวมมีค่าน้อยที่สุด ค่าความผิดพลาดรวม (Global Error: E) นิยามดังนี้

$$E = \sum_{i=1}^{n} E_i \tag{27}$$

โดยที่ E_i เป็นค่าความผิดพลาดของถุงตัวอย่างแต่ละถุง ซึ่งนิยามโดยขึ้นอยู่กับประเภทของถุงตัวอย่างนั้นๆ คือ

$$E_i = \begin{cases} \min\limits_{1 \le j \le m_i} \sum\limits_{k=1}^{o} E_{ijk} & if\ B_i = + \\ \max\limits_{1 \le j \le m_i} \sum\limits_{k=1}^{o} E_{ijk} & if\ B_i = - \end{cases} \tag{28}$$

และค่าความผิดพลาดของตัวอย่างย่อยนิยามดังนี้

$$E_{ijk} = \begin{cases} 0 & if\,(B_i = +) \text{ and } (0.5 \le o_{ijk}), l_{ik} = 1 \\ 0 & if\,(B_i = -) \text{ and } (o_{ijk} < 0.5), for\ all\ k \\ \frac{1}{2}(l_{ik} - o_{ijk})^2 & otherwise \end{cases} \quad (29)$$

เมื่อ E_{ijk} หมายถึงค่าความผิดพลาดของนิวรอนในชั้นผลลัพธ์ตัวที่ k ของตัวอย่างย่อย j ในถุงตัวอย่าง i

B_i = + หมายถึง ถุงตัวอย่างบวก

B_i = - หมายถึง ถุงตัวอย่างลบ

o_{ijk} หมายถึงผลลัพธ์ของนิวรอนชั้นผลลัพธ์ตัวที่ k ของตัวอย่างย่อย j ในถุงตัวอย่าง i

l_{ik} หมายถึง ค่าเป้าหมายของนิวรอนชั้นผลลัพธ์ตัวที่ k ของถุงตัวอย่างที่ i

ในการเรียนแต่ละรอบ ตัวอย่างที่ใช้ในการสอนจะถูกป้อนให้กับเน็ตเวิร์กทีละถุงทีละตัวอย่างย่อย จากนั้นจะทำการคำนวณหาค่าความผิดพลาดของแต่ละตัวอย่างย่อยในถุงนั้นๆ ตามสมการ (29) คือถ้าเน็ตเวิร์กจำแนกตัวอย่างได้ถูกต้องแล้ว ก็จะกำหนดให้ค่าความผิดพลาดสำหรับตัวอย่างย่อยนั้นมีค่าเป็น 0 โดยในกรณีที่ถุงตัวอย่างที่ป้อนให้กับเน็ตเวิร์กเป็นถุงตัวอย่างบวกและพบว่ามีตัวอย่างย่อยที่ให้ค่าความผิดพลาดเป็น 0 แล้ว ตัวอย่างย่อยที่เหลือไม่จำเป็นต้องป้อนให้กับเน็ตเวิร์กและไม่ต้องทำการปรับค่าน้ำหนักเส้นเชื่อมใดๆ เพราะถือว่าตัวอย่างย่อยตัวนั้นเป็นตัวอย่างที่ทำให้ถุงตัวอย่างเป็นบวกและเน็ตเวิร์กจำแนกได้ถูกต้องแล้ว แต่ถ้าเน็ตเวิร์กจำแนกผิดพลาดก็จะทำการหาค่าความผิดพลาดสำหรับตัวอย่างย่อยนั้น เพื่อนำไปหาค่าความผิดพลาดของถุงตัวอย่างตามสมการที่ (28) จากนั้นเมื่อป้อนครบทุกตัวอย่างย่อยในถุงแล้วก็จะนำค่าความผิดพลาดของถุงที่ได้ไปทำการปรับค่าน้ำหนักของเส้นเชื่อมตามวิธีของแบ็กพรอพาเกชัน แล้วจึงทำการป้อนถุงตัวอย่างใหม่เข้าไปและทำตามลำดับขั้นตอนเดิม โดยจะหยุดกระบวนการเรียนรู้เมื่อค่าความผิดพลาดรวมในสมการ (27) มีค่าลดลงจนถึงจุดที่กำหนดไว้หรือเรียนรู้จนครบจำนวนรอบที่ได้กำหนดไว้

2.2.4 การทดลองและผลการทดลอง

ในการทดลองเรานำชุดข้อมูล (Dataset) การวิเคราะห์ไฟไนต์เอลิเมนต์ (Finite Element Mesh Design: FEM) [Dolsak & Muggleton, 1992] และความสามารถในการก่อกลายพันธุ์ของโมเลกุล (Mutagenesis: MUTA) มาใช้ทำการทดลอง ปัญหา FEM มีจุดมุ่งหมายเพื่อหากฎที่ใช้ในการวิเคราะห์ไฟไนต์เอลิเมนต์ในโครงสร้างสำหรับงานทางด้านวิศวกรรม มีกลุ่มความรู้ภูมิหลังเป็นลักษณะต่างๆของโครงสร้างและประกอบด้วยประเภทของตัวอย่าง 12 ประเภทด้วยกัน โดยจะแบ่งประเภทตามจำนวนองค์ประกอบ (Element) ที่เหมาะสมของโครงสร้างนั้น ตัวอย่างแต่ละตัวถูกจัดอยู่ในรูปแบบ `mesh(Edge,Element)` เมื่อ `Edge` คือชื่อโครงสร้างและ `Element` คือ จำนวนองค์ประกอบภายในโครงสร้างนั้น รวมจำนวนตัวอย่างทั้งหมด 278 ตัวอย่าง ส่วนชุดข้อมูล MUTA เหมือนกับในหัวข้อที่แล้ว ในการทดลองนั้นใช้วิธี 3-fold cross validation ทำโดยแบ่งข้อมูลทั้งหมดออกเป็น 3 ส่วนเท่าๆกัน ทำการทดลองทั้งหมด 3 ครั้ง ในแต่ละครั้งจะเลือกส่วนหนึ่งใดๆ เป็นชุดทดสอบและส่วนที่เหลือ 2 ส่วนจะใช้เป็นชุดสอน

ในชุดข้อมูล FEM นั้นโครงสร้างของเน็ตเวิร์กที่ใช้จะมีจำนวนนิวรอนในชั้นนำเข้าทั้งหมด 130 นิวรอน กำหนดจากความรู้ภูมิหลังที่ได้รับเข้ามา นิวรอนในชั้นผลลัพธ์ 12 นิวรอน กำหนดจากประเภทของตัวอย่าง และใช้จำนวนนิวรอนในชั้นซ่อน 80 นิวรอน(จากการทดลอง) มีค่าอัตราการเรียนรู้เป็น 0.0001 และค่าโมเมนตัมเป็น 0.97 โดยกระบวนการเรียนรู้จะหยุดเมื่อครบ 6000 รอบหรือค่าความผิดพลาดรวมมีค่าน้อยกว่า 0.05 ส่วนใน

กรณีของ MUTA นั้น โครงสร้างของเน็ตเวิร์กที่ใช้จะมีจำนวนนิวรอนในชั้นนำเข้าทั้งหมด 235 นิวรอน นิวรอนในชั้นผลลัพธ์ 1 นิวรอน และใช้จำนวนนิวรอนในชั้นซ่อน 100 นิวรอน กำหนดค่าอัตราการเรียนรู้เป็น 0.0001 และค่าโมเมนตัมเป็น 0.97

ตารางที่ 15 ผลเปรียบเทียบเปอร์เซ็นต์ความถูกต้องในการทดสอบกับชุดข้อมูล FEM

ชุดข้อมูล	PROGOL	FOLNN
FEM	57.80	59.18
MUTA	84.58	88.27

ตารางที่ 15 แสดงผลการทดลองที่ได้ในการเปรียบเทียบระหว่าง FOLNN กับ PROGOL จะเห็นว่า FOLNN สามารถเรียนแนวคิดจากชุดข้อมูลตรรกะอันดับที่หนึ่งได้และให้เปอร์เซ็นต์ความถูกต้องในการจำแนกสูงกว่า PROGOL

นอกจากนี้แล้วเรายังได้ทำการทดลองกับข้อมูลมีสัญญาณรบกวน โดยใช้ชุดข้อมูล MUTA ที่ใช้ในการทดลองที่แล้ว (แบ่งเป็น 3 ส่วนแล้วเพื่อทำ 3-fold cross validation) มาทำการเติมสัญญาณรบกวนลงไปอย่างสุ่มจำนวน 10% และ 15% ที่ประเภทของตัวอย่าง การเติมสัญญาณรบกวนจะทำเฉพาะชุดข้อมูลฝึกเท่านั้น ไม่มีการเติมสัญญาณรบกวนลงในชุดข้อมูลทดสอบ การเติมสัญญาณรบกวน x% อย่างสุ่มหมายความว่า ในตัวอย่าง 100 ตัวจะมีตัวอย่างอยู่ x ตัวที่ถูกเลือกมาอย่างสุ่ม และทำให้มีค่าของประเภทของตัวอย่างนั้นๆ ผิดไปจากเดิม ผลการทดลองแสดงในตารางที่ 16

ตารางที่ 16 ผลเปรียบเทียบเปอร์เซ็นต์ความถูกต้องในการทดสอบกับชุดข้อมูล MUTA ที่มีสัญญาณรบกวน

ระดับสัญญาณรบกวนในชุดข้อมูล	PROGOL 0% noise setting	PROGOL 10% noise setting	PROGOL 15% noise setting	FOLNN
10%	64.23	69.72	71.29	84.01
15%	60.56	61.54	65.31	81.28

ตารางที่ 16 แสดงให้เห็นว่าในกรณีที่ชุดข้อมูลมีสัญญาณรบกวนเป็นจำนวน 10% ระบบ PROGOL จะให้ค่าความถูกต้องสูงที่สุดเป็น 71.29% และระบบ FOLNN ให้ค่าความถูกต้องเป็น 84.01% และในกรณีที่เพิ่มปริมาณสัญญาณรบกวนเป็นจำนวน 15% ระบบ PROGOL จะให้ค่าความถูกต้องสูงที่สุดเป็น 65.31% และระบบ FOLNN ให้ค่าความถูกต้องเป็น 81.28% โดยทั้งสองกรณีระบบ PROGOL ให้ค่าความถูกต้องสูงที่สุดเมื่อปรับค่าการรับมือสัญญาณรบกวนเป็น 15% โดยในกรณีที่มีสัญญาณรบกวน 10% มีระดับความเชื่อมั่นสูงกว่า 99.0% และในกรณีที่มีสัญญาณรบกวน 15% มีระดับความเชื่อมั่นสูงกว่า 99.5%

2.3 สรุปงานวิจัยการปรับปรุงประสิทธิภาพของไอแอลพีให้ทนทานต่อสัญญาณรบกวน

หัวข้อนี้ได้นำเสนอข่ายงานเบส์ลำดับที่หนึ่งและนิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่งเพื่อปรับปรุงประสิทธิภาพของไอแอลพีให้มีความทนทานต่อสัญญาณรบกวน การเรียนรู้ข่ายงานเบส์ลำดับที่หนึ่งทำโดยการประยุกต์ใช้ระบบไอแอลพีร่วมกับระบบเรียนรู้ข่ายงานเบส์แบบดั้งเดิม ผลการทดลองแสดงให้เห็นว่าวิธีการที่นำเสนอมีประสิทธิภาพดีกว่าระบบไอแอลพีโดยให้ความถูกต้องสูงกว่าและทนทานต่อสัญญาณรบกวนได้ดีกว่าระบบไอแอลพีดั้งเดิม นอกจากนั้นเราได้นำเสนอนิวรอลเน็ตเวิร์กแบบใหม่ที่ประยุกต์เข้ากับแนวคิดของการเรียนรู้เชิงตรรกะเรียกว่านิวรอลเน็ตเวิร์กอันดับที่หนึ่ง เพื่อเป็นแนวทางหนึ่งในการนำมาช่วยจัดการกับข้อจำกัดของการ

โปรแกรมตรรกะเชิงอุปนัย โดยที่นิวรอลเน็ตเวิร์กอันดับที่หนึ่งมีข้อดีที่สามารถรับอินพุตในรูปแบบของตรรกะอันดับที่หนึ่งได้โดยตรงและมีข้อดีของนิวรอลเน็ตเวิร์กซึ่งทนทานต่อข้อมูลที่มีสัญญาณรบกวนได้ดีอีกด้วย

2.4 การเปรียบเทียบข่ายงานเบย์ลำดับที่หนึ่ง นิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่งกับเอสวีเอ็มแบบหลายกลุ่ม

เราสามารถเปรียบเทียบข่ายงานเบย์ลำดับที่หนึ่ง นิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่งกับเอสวีเอ็มแบบหลายกลุ่มได้ดังนี้คือ อาร์เอดีเอจีซึ่งเป็นวิธีการที่เรานำเสนอขึ้นสำหรับเอสวีเอ็มแบบหลายกลุ่มนั้นมีข้อดีของเอสวีเอ็มที่ทนทานต่อสัญญาณรบกวนได้ดีเหมาะกับข้อมูลเชิงตัวเลข และจากการที่เราทำให้สามารถจำแนกข้อมูลหลายกลุ่มได้อย่างรวดเร็วมีประสิทธิภาพจึงทำให้สามารถนำไปใช้งานในทางปฏิบัติกับปัญหาที่มีจำนวนประเภทมากๆ ได้อย่างดี หากปัญหาที่เราสนใจอยู่นั้นเป็นข้อมูลเชิงตัวเลขก็จะเหมาะกับการใช้อาร์เอดีเอจี ส่วนข่ายงานเบย์ลำดับที่หนึ่งและนิวรอลเน็ตเวิร์กลำดับที่หนึ่งมีข้อดีที่สามารถรับความรู้ภูมิหลังได้และเหมาะกับข้อมูลเชิงสัมพันธ์ที่แสดงในรูปตรรกะลำดับที่หนึ่งซึ่งไม่เหมาะกับข้อมูลเชิงตัวเลข หากปัญหาที่เราสนใจอยู่นั้น เรามีความรู้พื้นฐานเกี่ยวกับปัญหานั้นอยู่และเราสามารถใส่ความรู้นั้นในรูปความรู้ภูมิหลังได้แล้ว ข่ายงานเบย์ลำดับที่หนึ่งและนิวรอลเน็ตเวิร์กลำดับที่หนึ่งก็เป็นทางเลือกที่ดี เนื่องจากทั้งสองวิธีนี้เราได้ทำให้มีความยืดหยุ่นมากขึ้นกว่าระบบไอแอลพีดั้งเดิมทำให้ทนทานต่อสัญญาณรบกวนได้ดีและสามารถจัดการกับข้อมูลหลายกลุ่มได้ด้วย

3. สรุปงานวิจัย

ในงานวิจัยนี้เราได้นำเสนอวิธีการสำหรับ (1) ปรับปรุงเอสวีเอ็มเพื่อจัดการกับปัญหาข้อมูลหลายกลุ่ม และ (2) ทำให้ไอแอลพีสามารถทนทานต่อข้อมูลมีสัญญาณรบกวน สำหรับวิธีการสำหรับเอสวีเอ็มแบบหลายกลุ่มนั้นเราได้นำเสนออาร์เอดีเอจีซึ่งมีประสิทธิภาพดีกว่าวิธีการดั้งเดิมที่มีอยู่ในปัจจุบันทั้งในด้านความเร็วในการทำงานและความถูกต้องของการจำแนกประเภทข้อมูล ซึ่งเป็นผลมาจากโครงสร้างกราฟแบบใหม่ที่เรานำเสนอร่วมกับการใช้ประสิทธิภาพโดยนัยทั่วไปในการจับคู่เอสวีเอ็มแบบสองกลุ่ม สำหรับการทำให้ไอแอลพีมีความทนทานต่อสัญญาณรบกวนนั้น เราได้ใช้ข้อดีของข่ายงานเบส์และนิวรอลเน็ตเวิร์กที่มีความทนทานต่อสัญญาณรบกวนได้ดีอยู่แล้วนำมาขยายให้รองรับกับข้อมูลเชิงสัมพันธ์ที่แสดงในรูปตรรกะลำดับที่หนึ่งได้ และทำให้เราได้ข่ายงานเบส์ลำดับที่หนึ่งและนิวรอลเน็ตเวิร์กตรรกะลำดับที่หนึ่งซึ่งมีประสิทธิภาพดีกว่าระบบไอแอลพีดั้งเดิม

รายการอ้างอิง

Bartlett, P. L. and Shawe-Taylor, J. (1999) Generalization performance of support vector machines and other pattern classifiers. *Advances in Kernel Methods - Support Vector Learning*, Schoelkopf, B., Burges, C. J. and Smola, A. J. (eds), pp. 43-54. MIT Press.

Blake, C., Keogh, E. and Merz, C. (1998) UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

Botta, M., Giordana, A., Saitta, L. and Sebag, M. (2000) Relational learning: Hard problems and phase transition. Selected papers from AIIA'99, Springer-Verlag.

Chevaleyre, Y. and Zucker, J. D. (2001) A framework for learning rules from multiple instance data. *The 12th European Conference on Machine Learning*, Freiburg, Germany.

Chickering, D. M. (1996) Learning Bayesian networks is NP-complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics* V.

Chickering, D. M. (2002) The WinMine toolkit. *Technical Report MSR-TR-2002-103*, Microsoft.

Cook, W. and Rohe, A. (1997) Computing minimum-weight perfect matchings. Technical Report 97863, Forschungsinstitut für Diskrete Mathematik, Universität Bonn.

Dietterich,T.G. (1997) Machine learning research: four current directions, *AI Magazine*, 4, 97-136.

Dolsak, B. and Muggleton, S. (1992) The application of inductive logic programming to finite element mesh design. In *Inductive Logic Programming*, S. Muggleton, Ed.: Academic Press, pp. 453–472.

Fensel, D., Zickwolff, M. and Weise, M. (1995) Are substitutions the better examples? In L. De Raedt, editor, *The proceedings of the 5th International Workshop on Inductive Logic Programming*.

Friedman, J. H. (1996) Another approach to Polychotomous classification. Technical report, Department of Statistics, Stanford University.

Garcez, A. S., Broda, K. B. and Gabbay, D. M. (2002) *Neural-Symbolic Learning Systems*: Springer-Verlag.

Getoor, L., Friedman, N., Koller, D. and Pfeffer, A. (2001) Learning probabilistic relational models. *Relational Data Mining*, S. Dzeroski and N. Lavrac, editors.

Hsu, C. W. and Lin, C. J. (2002) A comparison of methods for multiclass support vector machines. *IEEE Trans.on Neural Networks*, Vol. 13, pp. 415-425.

Huang, X., Chen, S. C. and Shyu, M. L. (2003) An open multiple instance learning framework and its application in drug activity prediction problems. *The Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering (BIBE'03)*, Bethesda, Maryland.

Kersting, K. and De Raedt, L. (2000) Basic principles of learning Bayesian logic programs. *Technical Report No. 174*, Institute for Computer Science, University of Freiburg, Germany.

Kijsirikul, B., Boonsirisumpun, N. and Limpiyakorn, Y. (2004) Multiclass support vector machines using balanced dichotomization. The *8th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2004)*.

Kijsirikul, B., Sinthupinyo, S. and Chongkasemwongse, K. (2001) Approximate match of rules using backpropagation neural networks. *Machine Learning*, 44(3), 273–299.

Kramer, S., Lavrac, N. and Flach, P. (2001) Propositionalization approaches to relational data mining, in: Dzeroski S., Lavrac N, editors, *Relational Data Mining.*

Lavrac, N. and Dzeroski, S. (1994) *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York.

Mitchell, T. (1997) *Machine Learning*, McGraw-Hill. New York.

Muggleton, S. (1991). Inductive logic programming, *New Generation Computing*, 8(4), 295–318.

Muggleton, S. (1995). Inverse entailment and PROGOL. *New Generation Computing*, 13, 245–286.

Pearl, J. (1991) *Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd edition.

Pearl, J. (2001) *Causality*. Addison Wesley.

Platt, J., Cristianini, N. and Shawe-Taylor, J. (1999) Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, MIT Press, Vol. 12, pp. 547-553.

Srinivasan, A. and King, R. D. (1999) Feature construction with inductive logic programming: A study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3(1):37-57.

Vapnik, V. (1998) *Statistical Learning Theory*. Wiley.

Wattuya, P., Rungsawang, A. and Kijsirikul, B. (2003) Multiple-instance neural network with strong boundary criteria. *The 7th National Computer Science and Engineering Conference*, Chonburi, Thailand.

Zhou, Z. H. and Zhang, M. L. (2002) Neural network for multi-instance learning. *Proceedings of the International Conference on Intelligent Information Technology*, Beijing, China.

II. ผลที่ได้จากโครงการ

งานวิจัยที่ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ

1. Thimaporn Phetkaew, Wanchai Rivepiboon and Boonserm Kijsirikul, "Reordering adaptive directed acyclic graphs for multiclass support vector machine", *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 7, No. 3, October 2003.
2. Boonserm Kijsirikul and Thanupol Lerdlumnouchai, "First-order logical neural network", *International Journal of Hybrid Intelligent Systems*, Vol. 2, No. 4, December, 2005.
3. Boonserm Kijsirikul and Thimaporn Phetkaew, "Adaptive directed acyclic graphs: Algorithms for multiclass support vector machines", *Journal of Machine Learning Research*, 2006 (submitted).

การนำเสนอผลงานในการประชุมวิชาการ

4. Thimaporn Phetkaew, Boonserm Kijsirikul and Wanchai Rivepiboon, "Multiclass classification of support vector machines by reordering adaptive directed acyclic graph", *Proceedings of SANKEN International Workshop on Intelligent Systems*, December 17, ICHO KAIKAN, Osaka University, Japan. 2003.
5. Thimaporn Phetkaew, Wanchai Rivepiboon and Boonserm Kijsirikul, "Learning multiclass support vector machines by reordering adaptive directed acyclic graph", *The 7th National Computer Science and Engineering Conference (NCSEC 2003)*, Cholburi, Thailand, 2003. October 28-30, 2003.
6. Thanupol Lerdlumnouchai and Boonserm Kijsirikul, "A new framework for learning first-order representation", *Proceedings Of the 7th Annual National Symposium on Computational Science and*

Engineering (ANSCSE'04), Suranaree University of Technology, Nakhon Rachasima, July 21-23, 2004.

7. Boonserm Kijsirikul, Narong Boonsirisumpun and Yachai Limpiyakorn. "Multiclass support vector machines using balanced dichotomization", *The 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2004)*, August 9-13, 2004.
8. Thanupol Lerdlumnouchai and Boonserm Kijsirikul, "First-order logic neural networks", *The Fourth International Conference on Hybrid Intelligent Systems (HIS04)*, Japan, December 5-8, 2004.
9. รัฐฉัตร ฉัตรพัฒนศิริ และ บุญเสริม กิจศิริกุล, "การเรียนรู้เน็ตเวิร์กเบย์ลำดับที่หนึ่ง", การประชุมวิชาการวิทยาการคอมพิวเตอร์และวิศวกรรมคอมพิวเตอร์แห่งชาติฯ ครั้งที่ 7, ชลบุรี, 28-30 ตุลาคม 2546.
10. ธนุพล เลิศลำเนาชัย และ บุญเสริม กิจศิริกุล, "การเรียนรู้โปรแกรมตรรกะโดยนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง", การประชุมวิชาการวิทยาการคอมพิวเตอร์และวิศวกรรมคอมพิวเตอร์แห่งชาติฯ ครั้งที่ 8, 21-22 ตุลาคม 2547.
11. ปทุมศิริ สงศิริ และ บุญเสริม กิจศิริกุล, "เทคนิคการแตกครึ่งตามสารสนเทศสำหรับซัพพอร์ตเวกเตอร์แมชชีนแบบหลายประเภท", กจรประชุมวิชาการวิทยาการคอมพิวเตอร์และวิศวกรรมคอมพิวเตอร์แห่งชาติฯ ครั้งที่ 9, 27-28 ตุลาคม 2548.

การผลิตบัณฑิต

โครงการนี้ได้ผลิตบัณฑิตในระดับดุษฎีบัณฑิตและมหาบัณฑิตที่ทำวิทยานิพนธ์เกี่ยวข้องกับโครงการดังต่อไปนี้

1. นางสาวฐิมาพร เพชรแก้ว	ระดับดุษฎีบัณฑิต
2. นายณรงค์ บุญสิริสัมพันธ์	ระดับมหาบัณฑิต
3. นางสาวปทุมศิริ สงศิริ	ระดับมหาบัณฑิต
4. นายธนุพล เลิศลำเนาชัย	ระดับมหาบัณฑิต

ภาคผนวก

Paper:

Reordering Adaptive Directed Acyclic Graphs for Multiclass Support Vector Machines

Thimaporn Phetkaew, Wanchai Rivepiboon, and Boonserm Kijsirikul

Department of Computer Engineering
Chulalongkorn University
Phayathai, Patumwan, Bangkok, 10330 Thailand
E-mail: Thimaporn.P@student.chula.ac.th
E-mail: {Wanchai.R, Boonserm.K}@chula.ac.th
[Received June 20, 2003; accepted August 26, 2003]

The problem of extending binary support vector machines (SVMs) for multiclass classification is still an ongoing research issue. Ussivakul and Kijsirikul proposed the Adaptive Directed Acyclic Graph (ADAG) approach that provides accuracy comparable to that of the standard algorithm-Max Wins and requires low computation. However, different sequences of nodes in the ADAG may provide different accuracy. In this paper we present a new method for multiclass classification, Reordering ADAG, which is the modification of the original ADAG method. We show examples to exemplify that the margin (or 2/|w| value) between two classes of each binary SVM classifier affects the accuracy of classification, and this margin indicates the magnitude of confusion between the two classes. In this paper, we propose an algorithm to choose an optimal sequence of nodes in the ADAG by considering the |w| values of all classifiers to be used in data classification. We then compare our performance with previous methods including the ADAG and the Max Wins algorithm. Experimental results demonstrate that our method gives higher accuracy. Moreover it runs faster than Max Wins, especially when the number of classes and/or the number of dimensions are relatively large.

Keywords: support vector machines, multiclass classification, ADAG, Reordering ADAG

1. Introduction

Support vector machines (SVMs) were primarily designed for two-class classification problems with their outstanding performance in real world applications. However, extending SVMs for multiclass classification is still an ongoing research issue. Typically, previous methods for solving the multiclass problem of SVMs are to consider the problem as the combination of two-class decision functions, e.g. one-against-one and one-against-the-rest. The one-against-the-rest approach works by constructing a set of k binary classifiers for a k-class problem. The i^{th} classifier is trained with all of the examples in the i^{th} class with positive labels, and all other examples with negative labels. The final output is the class that corresponds to the classifier with the highest output value. Friedman [5] suggested the Max Wins algorithm in which each one-against-one classifier casts one vote for its preferred class, and the final result is the class with the most votes. The Max Wins algorithm offers faster training time compared to the one-against-the-rest method. The Decision Directed Acyclic Graph (DDAG) method proposed by Platt et al. reduces training and evaluation time, while maintaining the accuracy relative to the Max Wins [10]. The comparison experiments by several methods on large problems in [6] show that the Max Wins algorithm and the DDAG may be more suitable for practical use. Ussivakul and Kijsirikul proposed the Adaptive Directed Acyclic Graph (ADAG) method which is the modification of the DDAG. This method reduces the dependency of the sequence of nodes in the structure as well as lowering the number of tests required to evaluate for the correct class. Their approach yields higher accuracy and reliability of classification, especially in such a case that the number of classes is relatively large [13]. There are also other implementations for multiclass SVMs, e.g., [1,3,4,8,9,11,15].

In this paper we reveal that the ADAG still is dependent on the sequence of its nodes, although it is less dependent on the order of binary classes in the sequence than the DDAG; there are still differences in accuracy between different sequences. This led to the reliability of the algorithm. Here we propose a novel method that improves reliability by choosing an optimal sequence which has less chance to predict the wrong class and dynamically reordering the sequence during classification process according to each test data.

This paper is organized as follows. In the next section, we review the DDAG and the ADAG. In Section 3, we introduce the modification of the ADAG to improve the performance. The numerical experiments are illustrated in Section 4. All experiments are based on datasets of the Machine Learning Repository at Irvine [2]. The results show that our method yields higher accuracy of classification. Moreover the running time used by our method is less than that of Max Wins. Finally, the conclusions are given in Section 5.

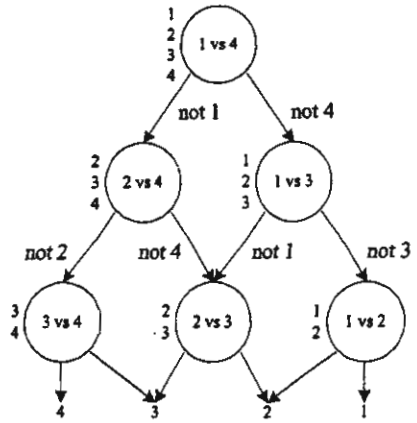


Fig. 1. The DDAG finding the best class out of four classes.

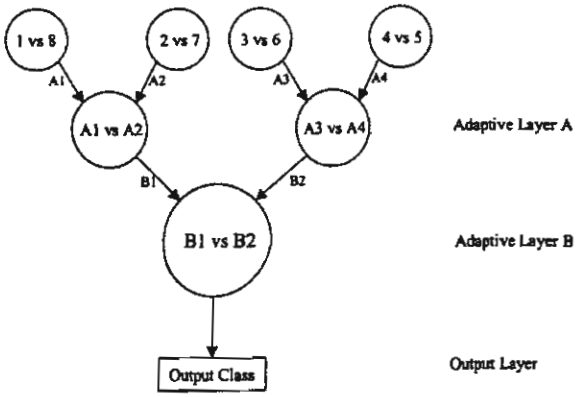


Fig. 2. The structure of an Adaptive DAG for an 8-class problem.

2. SVM Classification

This section describes two previous works on multi-class SVMs, which are related to our proposed method, i.e., the DDAG [6,10] and the ADAG [13].

2.1. DDAG

Platt et al. [10] presented a learning architecture, the Decision Directed Acyclic Graph (DDAG), which is used to combine many two-class classifiers into a multiclass classifier. For a k-class problem, its training phase is the same as the one-against-one method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, it uses a rooted binary directed acyclic graph which has $k(k\text{-}1)/2$ internal nodes and k leaves (**Fig.1**). Each node is a binary SVM of the i^{th} and j^{th} classes. Given a test sample x, starting at the root node, the binary decision function is evaluated. Then it moves to either the left or the right depending on the output value. Therefore, we go through a path before reaching a leaf node which indicates the predicted class.

There are some issues on the DDAG as pointed out by [13]. First, it gives outputs whose probabilities are not uniformly distributed, and thus its output depends on the sequence of binary classifiers in nodes, affecting the reliability of the algorithm. In addition, the correct class placed in a node near the root node is clearly at a disadvantage by comparison with the correct class near leaf nodes since it is exposed to a higher risk of being incorrectly rejected. Second, the number of node evaluations for the correct class is still unnecessarily high. This results in higher cumulative error and lower accuracy. The depth of the DDAG is k-1, which is the number of times the correct class has to be tested against other classes, on average, and scales linearly with k.

2.2. ADAG

Ussivakul and Kijsirikul [13] proposed an approach to alleviate the problem of the DDAG structure described above. An Adaptive DAG (ADAG) is a DAG with a reversed triangular structure. For a k-class problem, its training phase is the same as the DDAG method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, the nodes are arranged in a reversed triangle with $k/2$ nodes (rounded up) at the top, $k/2^2$ nodes in the second layer and so on until the lowest layer of a final node. It has k-1 internal nodes, each of which is labeled with an element of Boolean function (**Fig.2**). Given a test example x, starting at the top level, the binary decision function is evaluated. The node is then exited via the outgoing edge with a message of the preferred class. In each round, the number of candidate classes is reduced by half. Based on the preferred classes from its parent nodes, the binary function of the next-level node is chosen. The reduction process continues until reaching the final node at the lowest level. The value of the decision function is the value associated with the message from the final leaf node. Like the DDAG, the ADAG requires only k-1 decision nodes to be evaluated in order to derive an answer. Note that the correct class is evaluated against other classes for $\log_2 k$ times or less, considerably less than the number of evaluations required by the DDAG, which scales linearly with k.

An ADAG can be implemented using a list, where each node eliminates one class from the list (**Fig.3**). The list is initialized with a list of all classes. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the class is kept in the left element's position while the other class will be discarded from the list. Then, the ADAG proceeds to test the second and the elements before the last of the list. The testing process of each round ends when either one or no class remains untested in the list. After each round, the list is reduced to $k/2$ elements (rounded up). Then, the ADAG process repeats until only one class remains in the list.

Using the reversed triangular structure, the ADAG reduces the number of times the correct class is tested against other classes, and thus reduces the cumulative errors. However, there are still differences in accuracy between the different sequences of nodes. Next we will

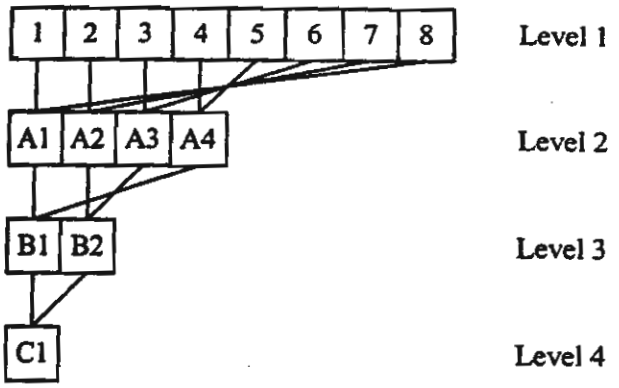


Fig. 3. Implementation through the list of the ADAG.

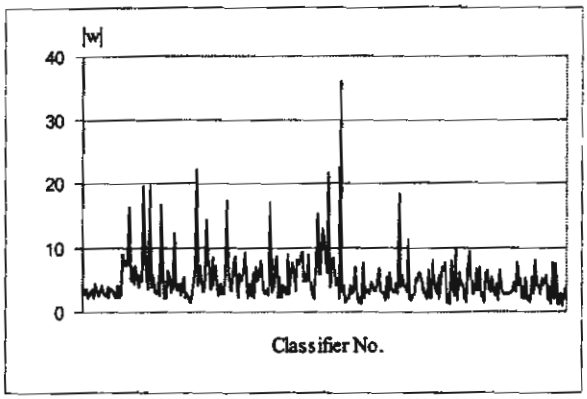


Fig. 4. The |w| values of 325 classifiers.

describe our method that improves the ADAG by finding a best sequence of nodes.

3. Reordering Adaptive Directed Acyclic Graphs

In this section, we introduce the modification of the ADAG to improve the performance of the original ADAG. This approach determines a best sequence of nodes in the ADAG by dynamically reordering the sequence during classification process according to each test data.

3.1. The Effect of |w|

The main idea of support vector machine classification is to construct an optimal hyperplane to separate the data of two classes. Suppose we have a data set D of l samples in an n-dimensional space belonging to two different classes (+1 and -1):

$$D = \{(\mathbf{x}_k, y_k) \mid k \in \{1,..,l\}, \mathbf{x}_k \in \Re^n, y_k \in \{+1,-1\}\}. \quad \ldots\ldots (1)$$

The hyperplane in the n dimensional space is determined by the pair $(\mathbf{w},b)$ where $\mathbf{w}$ is an n-dimensional vector orthogonal to the hyperplane and b is the offset constant. The hyperplane $(\mathbf{w} \cdot \mathbf{x})+b$ separates the data if and only if

$$\begin{aligned} (\mathbf{w} \cdot \mathbf{x}_i) + b > 0 \quad &\text{if} \quad y_i = +1 \\ (\mathbf{w} \cdot \mathbf{x}_i) + b < 0 \quad &\text{if} \quad y_i = -1. \end{aligned} \quad \ldots\ldots (2)$$

If we additionally require that $\mathbf{w}$ and b be such that the point closest to the hyperplane has a distance of $1/|\mathbf{w}|$, then we have

$$\begin{aligned} (\mathbf{w} \cdot \mathbf{x}_i) + b \geq 1 \quad &\text{if} \quad y_i = +1 \\ (\mathbf{w} \cdot \mathbf{x}_i) + b \leq -1 \quad &\text{if} \quad y_i = -1 \end{aligned} \quad \ldots\ldots (3)$$

which is equivalent to

$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \quad \forall i. \quad \ldots\ldots (4)$$

To find the optimal separating hyperplane, we have to find the hyperplane that maximizes the minimum distance between the hyperplane and any sample of training data. The distance between two closest samples from different classes is

$$d(\mathbf{w},b) = \min_{\{\mathbf{x}_i | y_i=1\}} \frac{(\mathbf{w} \cdot \mathbf{x}_i) + b}{|\mathbf{w}|} - \max_{\{\mathbf{x}_i | y_i=-1\}} \frac{(\mathbf{w} \cdot \mathbf{x}_i) + b}{|\mathbf{w}|}. \quad \ldots\ldots (5)$$

From (3), we can see that the appropriate minimum and maximum values are ±1. Therefore, we need to maximize

$$d(\mathbf{w}, b) = \frac{1}{|\mathbf{w}|} - \frac{-1}{|\mathbf{w}|} = \frac{2}{|\mathbf{w}|}. \quad \ldots\ldots (6)$$

As shown in Equations (5) and (6), the distance between the two closest samples from different classes is $2/|\mathbf{w}|$. The greater the distance, the less confusion between these two classes will be. Below we show some examples to illustrate that the |w| value affects the accuracy of data classification. The experiment is based on the English letter image recognition dataset from [2] which has 26 classes. Hence there are 325 classifiers. In this case, the dataset is trained by using Polynomial kernel degree 3. In **Fig.4**, |w| values of all classifiers are depicted. The average of all of them is 4.87.

Figure 5 shows two test examples, which were evaluated using the ADAG method. For the first example (**Fig.5(a)**), the correct class was class 8 but the classifier in the second level incorrectly eliminated the correct class from the list. The |w| value of this classifier was 21.50, which was much bigger than the average (4.87). For the second example (**Fig.5(b)**), the correct class was class 9 but the classifier in the fifth level eliminated the correct class from the list. The |w| value of this classifier was 36.03. Both test examples were misclassified because of classifiers with high |w| values.

We evaluated 4,000 test examples in this experiment. **Table 1** shows the frequency of |w| values that caused

Table 1. The frequency of |w| values that cause wrong classification.

|w|	Frequency	No. of classifiers
<4.0	4	169
4.0-5.0	16	51
5.0-6.0	11	30
6.0-7.0	10	22
7.0-8.0	22	21
8.0-9.0	11	10
9.0-10.0	12	6
>10.0	82	16

Table 2. Description of the datasets used in the experiments.

Dataset	#training data	#test data	#class	#dimension
Glass	214	5-fold	6	9
Satimage	4,435	2,000	6	36
Segment	2,310	3-fold	7	18
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35
Letter	15,963	4,037	26	16
Isolet	6,238	1,559	26	617

incorrect classification. The distribution of the |w| values of all classifiers is illustrated in the third column. As shown in the table, classifiers with small |w| values caused very few wrong classifications, whereas those with large |w| values gave a high number of wrong classifications. For instance, 169 classifiers with |w| less than 4.0 gave wrong classifications only 4 times, but those with |w| greater than 10.0 created a great number of incorrect classifications (82 times).

To conclude this subsection, we compare |w| values to the average of all |w| values. From the table, the classifiers with high |w| values, e.g. those with |w| higher than the average, caused more wrong classifications than those with low |w| values. In summary, the |w| value affects the accuracy of classification, and moreover it indicates the magnitude of the confusion between two classes.

3.2. Reordering ADAG

We propose a method, called Reordering ADAG, to improve the accuracy of the original ADAG. For a k-class problem, the Reordering ADAG's training phase is the same as the ADAG method by solving $k(k\text{-}1)/2$ binary SVMs. However, the testing phase is organized as follows. Like the ADAG, a Reordering ADAG can be implemented using a list, where each node eliminates one class from the list. The differences are the initialization of the list and the order of sequence in each level (**Fig.6**). In the first step, we use a reordering algorithm described in the next subsection to choose the optimal sequence to be the initial list. We use the list to evaluate every test example. In the second step, as in the ADAG, a test point of the Reordering ADAG is evaluated against the decision node

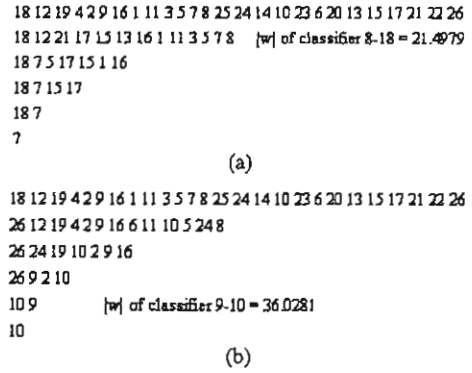


Fig. 5. Examples of the ADAG method.

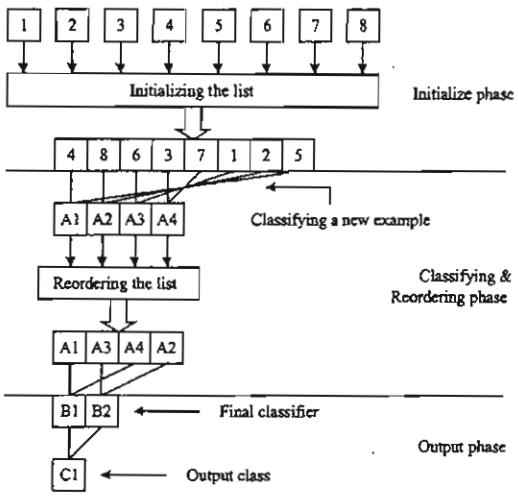


Fig. 6. Implementation through the list of the Reordering ADAG.

that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the class is kept in the left element's position whereas the other class will be discarded from the list. Then, the Reordering ADAG proceeds to test the second and the elements before the last of the list and so on. The testing process continues until either one or no class remains untested in the list. In the third step, unlike the ADAG, the Reordering ADAG will reorder the list before processing in the next level by using the reordering algorithm. This sequence differs for each test example, and it depends on the results of nodes from the previous level. The second and the third steps are repeated until there is only one class remains.

3.3. Reordering Algorithm

For the reason described above, we consider |w| values in order to choose the optimal sequence, from all possible $\frac{k!}{2^{\lfloor \frac{k}{2} \rfloor}}$ sequences, with less chance to predict the wrong class. A low |w| value means less confusion between two classes. Each classifier has one |w| value. Among classifiers, $k/2$ classifiers which have small |w| values will be considered to be used in data classification.

Table 3. Comparison of accuracies using the Polynomial kernel.

Dataset	d	DDAG	d	ADAG	d	Max Wins	d	Reordering
Glass	2	71.069**	2	71.135*	2	71.078**	2	**71.528**
Satimage	6	88.408	6	88.430	6	88.453	6	**88.800**
Segment	6	96.538*	4	96.620**	4	96.631**	4	**96.840**
Shuttle	8	**99.924**	8	**99.924**	8	**99.924**	8	**99.924**
Vowel	3	64.237	3	64.293	3	64.329	3	**64.719**
Soybean	5	90.400	5	90.446	3	90.471	3	**91.176**
Letter	3	95.508*	3	95.984	3	96.125	3	**96.235**
Isolet	3	97.032	3	97.030	3	97.040	3	**97.051**

Table 4. Comparison of accuracies using the RBF kernel.

Dataset	c	DDAG	c	ADAG	c	Max Wins	c	Reordering
Glass	0.08	72.850***	0.08	72.759***	0.09	73.238***	0.09	**74.536**
Satimage	3.0	91.971	3.0	91.968	3.0	**91.984**	3.0	91.950
Segment	0.7	97.276***	0.7	97.288***	0.7	97.302**	0.7	**97.446**
Shuttle	3.0	**99.897**	3.0	**99.897**	3.0	**99.897**	3.0	**99.897**
Vowel	0.2	65.425	0.2	65.589	0.2	65.340	0.2	**67.532**
Soybean	0.07	90.353	0.08	90.412	0.08	90.468	0.07	**90.882**
Letter	3.0	97.901	3.0	97.909	3.0	97.918	3.0	**97.969**
Isolet	0.01	96.939	0.01	96.932	0.01	96.916	0.01	**96.987**

The reordering algorithm is illustrated in **Fig.7.** k is the number of classes, $k/2$ is the number of classifiers which will be used in the sequence, and j is the current classifier being added to the sequence. *Set-of-tried-classifiers* keeps classifiers which were tried to be selected for using in data classification. *Sequence-of-the-selected-classes* keeps classes that will be used in a form of list described above. Threshold θ is the highest $|w|$ value that can be used in the sequence. If it cannot find the optimal sequence according to the threshold, the threshold is increased by a small constant ε.

Note that different thresholds provide different accuracy. The effective way to choose an appropriate threshold is to select the value that can eliminate classifiers which have high $|w|$ values. However, very low threshold values may cause long reordering time.

4. Numerical Experiments

In this section, we present experimental results on several datasets from the UCI Repository of machine learning databases [2] including glass, satimage, segment, shuttle, vowel, soybean, letter and isolet (**Table 2**). These datasets are different in the number of classes, the number of dimensions, and sizes. For glass and segment problems, there is no provided test data so we used 5-fold cross validation and 3-fold cross validation respectively. For the soybean problem, we discarded the last four classes because of missing values.

In these experiments we scaled both training data and test data to be in [-1,1] and employed the following Polynomial and RBF kernels.

$$\text{Polynomial degree d: } k(\mathbf{x},\mathbf{y}) = |\mathbf{x}\cdot\mathbf{y}+1|^d \quad \ldots\ldots (7)$$

$$\text{Radial basis function: } k(\mathbf{x},\mathbf{y}) = e^{-|\mathbf{x}-\mathbf{y}|^2/c} \quad \ldots\ldots\ldots (8)$$

Let *set-of-tried-classifiers*[i] be the set of classifiers which has been tried to be selected as classifier i in the output sequence. Let Ci, $|w|_i$ and θ be classifier i, the $|w|$ value of classifier i, and the threshold, respectively. Let $TCi[1]$ and $TCi[2]$ be the two classes corresponding to classifier i.

1. Initializing phase:
   $j = 0$;
   For $i = 1$ to $k/2$ do *set-of-tried-classifiers*[i] = { };
   For $i = 1$ to $k/2$ *sequence-of-the-selected-classes*[i] = nil;
   Sort all $k(k-1)/2$ classifiers by ascending the $|w|$ values;
2. Selecting phase:
   For $i = 1$ to $k(k-1)/2$ do
      If ($j<k/2$)
         If ($|w|_i < \theta$)
            If ((($TCi[1], TCi[2]$) $\notin$ *sequence-of-the-selected-classes*) and
            ($Ci \in$ *set-of-tried-classifiers*[$j+1$]))
               j++;
               sequence-of-the-selected-classes[j] = $TCi[1]$;
               sequence-of-the-selected-classes[$k-j+1$] = $TCi[2]$;
               set-of-tried-classifiers[j] = *set-of-tried-classifiers*[j] $\cup$ {Ci};
               If (j= $k/2$)
                  Goto End;
               Else Goto Selecting phase;
         Else
            If ($\forall i$, $Ci \in$ *set-of-tried-classifiers*[j])
               set-of-tried-classifiers[j] = { };
            Goto Backtracking phase;
3. Backtracking phase:
   sequence-of-the-selected-classes[j] = nil;
   sequence-of-the-selected-classes[$k-j+1$] = nil;
   j--;
   If ($j<0$) /* This means that all classifiers were already tested
                 and it cannot find the optimal sequence. */
      $\theta = \theta + \varepsilon$;
   Goto Selecting phase;
4. End.

Fig. 7. Reordering algorithm.

In the experiments, we compare four algorithms, i.e., the DDAG, the original ADAG, the Reordering ADAG, and the Max Wins algorithm. For the DDAG and the ADAG, we examined all possible sequences for datasets having not more than 7 classes, whereas we randomly selected 50,000 sequences for datasets having more than 7 classes. **Table 3** and **4** present the results of comparing these methods for Polynomial and RBF kernels, respectively. We present the optimal parameters (d and c in Equations (7) and (8)) of the kernels and the corresponding accuracies for each method. The best accuracy among these methods is illustrated in bold-face. ***, ** and * in the tables mean 99%, 95%, and 90% confidence interval for estimating the difference between accuracies of three algorithms and the Reordering ADAG using a one-tailed paired t-test.

The results show that our method yields higher accuracy than the DDAG, the original ADAG and Max Wins in all datasets, except for the shuttle problem where the accuracies of four methods are the same, and for the satimage problem where Max Wins gives the best accuracy in the RBF kernel. The results also show that our method performs statistically significantly better than the other methods in the glass and segment problems, and significantly better than the DDAG in the letter problem in case of the polynomial kernel. Another advantage of our method compared to the DDAG and the original ADAG is that our method always provides one best accuracy for each dataset using the reordering algorithm, whereas, depending on the sequence of binary classifiers,

Table 5. Comparison of the computational time using the Polynomial kernel.

Dataset	d	Reordering ADAG			Max Wins
		Classifying (seconds)	Reordering (seconds)	Total (seconds)	Classifying (seconds)
Satimage	6	2.61	0.22	2.83	9.47
Segment	4	0.18	0.56	0.74	0.41
Shuttle	8	1.75	1.47	3.22	5.15
Vowel	3	0.12	0.19	0.31	0.61
Soybean	3	0.27	4.28	4.55	1.86
Letter	3	8.58	2.56	11.14	125.58
Isolet	3	130.73	1.08	131.81	1671.98

Table 6. Comparison of the computational time using the RBF kernel.

Dataset	c	Reordering ADAG			Max Wins
		Classifying (seconds)	Reordering (seconds)	Total (seconds)	Classifying (seconds)
Satimage	3.0	11.75	0.20	11.95	37.13
Segment	0.7	0.39	0.18	0.57	0.83
Shuttle	3.0	3.23	1.49	4.72	9.27
Vowel	0.2	0.10	0.28	0.38	0.61
Soybean	0.07	0.32	0.84	1.16	2.20
Letter	3.0	61.90	2.67	64.57	802.85
Isolet	0.01	124.32	1.22	125.54	1369.11

the DDAG and the original ADAG may give low accuracies. This shows the effectiveness of the Reordering ADAG.

Table 5 and **6** present the comparison of the computational time between the Reordering ADAG and the Max Wins for Polynomial and RBF kernels by using a 400 MHz Pentium II processor. There is no computational time of the glass dataset because it has too few test examples to measure the time. The results show that our method requires low computational time in almost all of the datasets, especially when the number of classes and/or the number of dimensions are relatively large. For a k class problem, the Max Wins requires $k(k\text{-}1)/2$ classifiers for the classification whereas the Reordering ADAG requires only k-1 classifiers. Hence the larger the number of classes, the more running time the Max Wins requires than the Reordering ADAG. Moreover, the number of dimensions affects the running time of each classifier. For the Reordering ADAG, the number of classes and the threshold θ affect the running time for reordering. However, it takes some time even when there are many classes.

5. Conclusions

In this paper, we have presented a new approach for multiclass SVMs, which is the modification of the original ADAG. The experiments denote that $|w|$ affects the accuracy of classification. A low $|w|$ value creates less confusion between two classes. Our approach eliminates the dependency of the sequence of nodes in the original ADAG by selecting an appropriate sequence, from all possible sequences, which consists of classifiers with small $|w|$ values. The experimental results show that our new approach yields a higher accuracy than the DDAG, the original ADAG and the Max Wins, which was probably the most accurate method for multiclass SVMs. Moreover the running time used by the Reordering ADAG is less than Max Wins, especially when the number of classes and/or the number of dimensions are relatively large.

Acknowledgments

We are grateful to the anonymous referees for useful comments and suggestions. This work was supported by the Thailand Research Fund. The views and conclusions contained in this paper are those of the authors and the Thailand Research Fund is not necessarily of the same opinion.

References:

[1] E. Allwein, R. Schapire and Y Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers", Int. Conf. on Machine Learning, 2000.

[2] C. Blake, E. Keogh and C. Merz, "UCI Repository of Machine Learning Databases", Department of Information and Computer Science, University of California, Irvine, 1998. http://www.ics.uci.edu/~mlearn/MLSummary. html

[3] K. Crammer and Y Singer, "On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines", J. of Machine Learning Research, 2, pp. 265-292, December, 2001.

[4] X. Dong, W. Zhaohui and P. Yunhe, "A New Multi-class Support Vector Machines", IEEE Int. Conf. on System, Man, and Cybernatics, 3, pp. 1673-1676, 2001.

[5] J. Friedman, "Another Approach to Polychotomous Classification", Technical report, Department of Statistics, Stanford University, 1996.

[6] C. Hsu and C. Lin, "A Comparison of Methods for Multiclass Support Vector Machines", IEEE Trans. on Neural Networks, 13, pp. 415-425, March, 2002.

[7] T. Joachims., "Making large-scale SVM learning practical", Advances in Kernel Methods - Support Vector Learning, MIT Press, 1998.

[8] J. Kindermann, E. Leopold and G. Paass, "Multi-class Classification with Error Correcting Codes", Treffender GI-Fachgruppe 1.1.3, Maschinelles Lernen, GMD Re. 114, 2000.

[9] E. Mayoraz, and E. Alpaydm, "Support Vector Machines for Multiclass Classification", IDIAP-RR 6, IDIAP, 1998.

[10] J. Platt, N. Cristianini and J. Shawe-Taylor, "Large Margin DAGs for Multiclass Classification", Advances in Neural Information Processing Systems, MIT Press, 12, pp. 547-553, 2000.

[11] C. Sekhar, K. Takeda and F. Itakura, "Close-class-set Discrimination Method for Large-class-set Pattern Recognition using Support Vector Machines", IEEE Int. Joint Conf. on Neural Networks, pp. 577-582, 2002.

[12] N. Thubthong and B. Kijsirikul, "Support Vector Machines for Thai Phoneme Recognition", Int. Conf. on Intelligent Technologies, pp. 229-234, 2000.

[13] N. Ussivakul and B. Kijsirikul, "Adaptive DAG: Another Approach for Multiclass Classification", Int. Conf. on Intelligent Technologies, 2001.

[14] V. Vapnik, "Statistical Learning Theory", New York, Wiley, 1998.

[15] J. Weston and C. Watkins, "Multi-Class Support Vector Machines", Technical Report CSD-TR-98-04, Department of Computer science, Royal Holloway, University of London, May, 1998.

Name:
Thimaporn Phetkaew

Affiliation:
Ph.D. Student, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand

Address:
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330, Thailand

Brief Biographical History:
2000-present Ph.D. Student, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand

Main Works:
- "Reordering Adaptive Directed Acyclic Graphs for Multiclass Support Vector Machines", Proceedings of the Third International Conference on Intelligence Technologies and Third Vietnam-Japan Symposium on Fuzzy Systems and Applications, pp. 276-284, 2002.

Name:
Wanchai Rivepiboon

Affiliation:
Associate Professor, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand

Address:
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University,Bangkok 10330, Thailand Tel: (66-02)2186988, (66-01)9883897 Fax: (66-02)2186955

Brief Biographical History:
1985-1987 Chair, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand
1998-present Head, Software Engineering Laboratory, Department of Computer Engineering, Faculty of Engineering,Chulalongkorn University, Bangkok 10330 Thailand
2000-2003 Dean, Faculty of Informatics, Mahasarakham University, Mahasarakham 44150, Thailand
2000-present Deputy Director, Office of Information Technology Administration for Educational Development, Ministry of University Affairs, Bangkok 10330 Thailand

Main Works:
- "Branch Index: An Access Method of Aggregation Hierarchy as a Tree in OODB", International Journal of Information Technology, Vol. 7, 2002.
- "Formal Specification Scheme for Database Applications Using Requirements Particle Networks", International Journal for Computer-Aided Engineering and Software, Vol. 19, No. 8, pp. 932-952, 2002.

Membership in Learned Societies:
- The Computer Association of Thailand
- IEEE Computer Society

Name:
Boonserm Kijsirikul

Affiliation:
Assistant Professor, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand

Address:
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand

Brief Biographical History:
1993-2000 Lecturer, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand
1998-present Head, Machine Intelligence and Knowledge Discovery Laboratory, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand
2000-2003 Assistant Professor, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand

Main Works:
- "Approximate Match of Rules Using Backpropagation Neural Networks", Machine Learning Journal, Vol. 44, Issue 3, pp. 273-299, September, 2001.
- "A Method for Isolated Thai Tone Recognition Using Combining Neural Networks", Computational Intelligence Journal, Vol. 18, No. 3, pp. 313-335, August, 2002. Blackwell Publishers Inc, Boston, USA and Oxford, UK, 2002.

Membership in Learned Societies:
- AAAI Member

ternational Journal of Hybrid Intelligent Systems 2 (2005) 253–267
OS Press

First-Order Logical Neural Networks

Boonserm Kijsirikul* and Thanupol Lerdlamnaochai
Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Phayathai Rd., Pathumwan, Bangkok 10330, Thailand

Abstract. Inductive Logic Programming (ILP) is a well-known machine learning technique for learning concepts from relational data. Nevertheless, ILP systems are not robust enough to noisy or unseen data in real world domains. Furthermore, in multi-class problems, if the example is not matched with any learned rules, it cannot be classified. This paper presents a novel hybrid learning method to alleviate this restriction by enabling Neural Networks to handle first-order logic programs directly. The proposed method, called First-Order Logical Neural Network (FOLNN), employs the standard feedforward neural network and integrates inductive learning from examples and background knowledge. We also propose a method for determining the appropriate variable substitution in FOLNN learning by using Multiple-Instance Learning (MIL). In the experiments, the proposed method has been evaluated on two first-order learning problems, i.e., the Finite Element Mesh Design and Mutagenesis and compared with the state-of-the-art, the PROGOL system. The experimental results show that the proposed method performs better than PROGOL.

Keywords: Hybrid system, first-order logic, inductive logic programming, neural networks

1. Introduction

Machine Learning [19] is concerned with the development of procedures to make computers learn and improve with experience like humans. Specifically, a machine learning technique learns a concept by constructing hypotheses fitting the given training examples. The learned hypothesis is then employed to classify unseen data. There are several successful well-known techniques in machine learning such as Inductive Logic Programming (ILP) [17,20], Artificial Neural Networks (ANNs) [3], Decision Tree learning [23], Bayesian Networks [22], etc.

Inductive Logic Programming (ILP) is only one of machine learning techniques which adapts the first-order logical concepts for hypothesis learning. The advantages of ILP are the ability to employ background knowledge that allows the trainer to give useful knowledge to the learner more easily and the ability to induce the human readable representations in form of a set of first-order rules (if-then rules). In addition, the first-order rules are more expressive than propositional rules. Consider the advantage of first-order rule over a propositional one for representing concept *Daughter* as shown in Fig. 1.

As shown in Fig. 1, the propositional rule which cannot use variables is specific only for the first and the second persons being *Jannifer* and *Andrew*, respectively. Therefore, it is rarely useful for unseen pairs of people. On the other hand, the first-order rule contains variables that make the rule much more expressive and general and perfectly defines relations between the arguments of the target concept.

Commonly, most ILP systems have been used in two-class classification problems. The systems receive positive and negative examples and background knowledge as inputs and produce a set of rules.

*Corresponding author. Tel.: +66 2218 6976; Fax: +66 2218 6955; E-mail: boonserm.k@chula.ac.th.

Propositional logic:	IF (Name1=Jannifer) ∧ (Name2=Andrew) ∧ (Father1 = Andrew) ∧ (Female1=True) THEN Daughter1,2 = True
First-order logic:	IF Father(y,x) ∧ Female(y) THEN Daughter(x,y)

Fig. 1. The propositional and first-order logic rules.

All of the inputs and the learned rules are described by first-order logic. The learned rules should cover many positive examples and few negative examples. If we want to learn concepts for multi-class problems by using these two-class systems, we will begin by constructing a set of rules for the first class by using its examples as positive and the other examples as negative. Then we follow this process to create a set of rules for the other classes. The learned rules are then used to classify test examples. If a test example is perfectly covered by a rule of some class, the corresponding class is the classification result. However, these first-order rules learned by ILP systems have the restriction to handle imperfect data in real world domains such as noisy unseen data. This problem noticeably occurs especially in multi-class classification. In the case of two-class problems, any test example not covered by the induced rules is classified as negative. However, in multi-class classification, if an example is not covered by any learned rule, the method could not determine the correct class for the example. The simple solution is to assign the majority class recorded from training examples to the unable labeled test data [9]. Also, there is more efficient method to solve this problem by using the concept of intelligent hybrid systems [30].

Differently from (symbolic) ILP systems, (numerical) artificial neural networks perform the statistical learning to encode natures of data in a set of weights. Neural networks contain the ability to process inconsistent and noisy data. Moreover, they compute the most reasonable output for each input. Neural networks, because of their potential in noise tolerance and multi-class classification, offer attractiveness for combining with symbolic components to avoid the restrictions of symbolic rule-based systems described above. Although the ability of neural networks could alleviate the problem in symbolic rule-based systems, a learned hypothesis from neural networks is not available in a form that is legible for humans. Therefore neural networks require an interpretation by rule-based systems [30]. Several works show that the integration between robust neural networks and symbolic knowledge representation can improve classification accuracy such as Towell and Shavlik's KBANN [29], Mahoney and Mooney's RAPTURE [18], the works proposed by Parekh and Honavar [21] and Garcez et al. [10]. Nevertheless, these researches have been restricted to propositional theory refinement. Some models have been proposed for first-order theory. SHRUTI [26] employed a model making a restricted form of unification; actually this system only propagates bindings. The work proposed by Botta et al. [4] created a network consisting of restricted form of first-order logic. Kijsirikul et al. [16] proposed a feature generation method and a partial matching technique for first-order logic but their method still uses an ILP system in its first-step learning and does not select the appropriate values for variable substitution. FOCA, which is a hybrid first-order knowledge-based neural network, demonstrates better performance over standard ILP systems on some benchmark problems [2]. The approximation of semantics of logic programs and neural networks can be found in [11,12].

In this paper, we are interested in solving the problem by using the concept of hybrid systems. A more flexible learning system that directly learns first-order logic programs by neural networks, called First-Order Logical Neural Network (FOLNN) has been proposed. FOLNN is a neural-symbolic learning system based on the feedforward neural network that integrates inductive learning from examples and background knowledge. We also propose the method that makes use of Multiple-Instance Learning

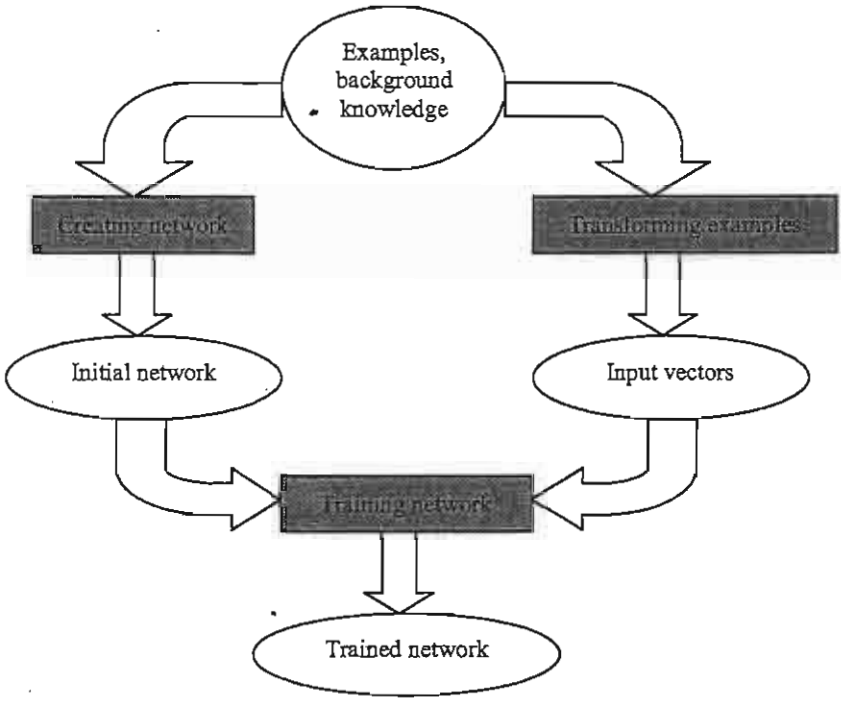


Fig. 2. FOLNN processes.

L) [6,14] for determining the variable substitution in our model. A modified version of Backprop-ion (BP) algorithm (see Section 2.3) is then applied to train the network within the framework of . The proposed method has been evaluated on two standard first-order learning datasets i.e., the te Element Mesh Design [8] and Mutagenesis [27]. We also test FOLNN on the noisy data to see robustness of the system. The results show that the proposed method effectively learns first-order esentation and provides more accurate results than an ILP system.

he rest of this paper is organized as follows. Section 2 describes the processes of first-order learning FOLNN. The experimental results on first-order problems are shown in Section 3. Finally the clusions are given in Section 4.

irst-Order Logical Neural Network (FOLNN)

he main reason for integrating robust neural networks and symbolic components is to improve the racy of classification by taking the advantages of neural networks which are noise tolerance and ability of multi-class classification. Combining these two techniques together is normally known eural-symbolic learning systems [10]. Our proposed method, FOLNN is also this type of learning ems. The overview of FOLNN processes is shown in Fig. 2.

irst, FOLNN receives positive and negative examples and background knowledge as inputs. Remark all inputs are in form of first-order logic. The background knowledge is used to determine the cture of the neural network. Before training the network, a process of transforming examples is uired. The process transforms examples represented in form of first-order logic into real value t vectors that can be used by the network. The transformed input vectors are then fed to train the vork. The training process will adapt the connection weights of the network so that it correctly sifies training examples. The Backpropagation (BP) algorithm [25] with sigmoid activation function

father(x,x)	father(x,y)	father(x,z)
father(y,x)	father(y,y)	father(y,z)
father(z,x)	father(z,y)	father(z,z)

Fig. 3. The expanded literals.

is selected to perform the training process. In the testing process, a test data is also converted to an input vector before it is fed to the network as same as the training process.

The following subsections explain the detail of the FOLNN learning processes that are (1) creating an initial network, (2) transforming the examples, and (3) training the network.

2.1. Creating an initial network

In this subsection, we present the first step of the FOLNN algorithm, creating an initial network from examples and background knowledge. The FOLNN structure is based on the feedforward neural network, but FOLNN receives examples and background knowledge in form of first-order logic programs as the inputs and directly learns from these inputs. Nevertheless, creating networks from first-order logic inputs is quite different from propositional logic inputs because in propositional logic there is no variable describing relations between literals. Therefore, we cannot directly employ the method of constructing neural networks for propositional logic inputs, such as the method of C-IL2P proposed by d'Avila Garcez et al. [10].

First, we restrict the number of variables to be used in the learning process, and then construct all possible input neurons each of which corresponds to a literal with different variables. In fact, not all possible combinations of variables will be useful, and some of them can be pruned. An input neuron corresponding to a literal will be pruned if the truth value of the literal is false according to background knowledge in all situations. For example, if we restrict the number of variables to 3, predicate *father* is expanded as shown in Fig. 3.

In all nine expanded literals, the truth values of three literals, i.e., *father*(*x*,*x*), *father*(*y*,*y*) and *father*(*z*,*z*), always are false according to background knowledge (there is no person who can be his own father). Therefore, these literals can be pruned and the six remaining literals will be used for creating the network.

FOLNN is a three-layer feedforward network, composed of one input layer, one output layer and one hidden layer [13]. The functionality of each layer is defined as follows.

- The input layer is the first layer that receives and computes input data, and then transmits the processed data to the hidden layer. This layer represents the literals for describing the target rule. The number of units in this layer depends on the number of predicates in background knowledge. The number of units for a predicate equals to the number of all possible permutations of variables of the predicate.
- The hidden layer connects between the input layer and the output layer. The hidden layer helps the network to learn the complex classification. The number of units in this layer depends on the complication of the learning concept, and is determined by the experiments.
- The output layer is the last layer of the network which produces the output for the classification. The target concept is represented in this layer so that the number of units in the output layer equals to the number of classes.

An initial network is created by using the above definition. To illustrate the construction of the network, consider the task of finite element mesh design (see Section 3.1.1 and [8] for details). The goal of finite

- **background knowledge:** short(A1), not_loaded(A1), fixed(A1), short(A2), fixed(A2), not_loaded(A3), long(A3)
- **examples of class 1:** mesh(A1,1), mesh(A2,1)
- **example of class 2:** mesh(A3,2)

Fig. 4. Inputs for learning the concept *rich(x)*.

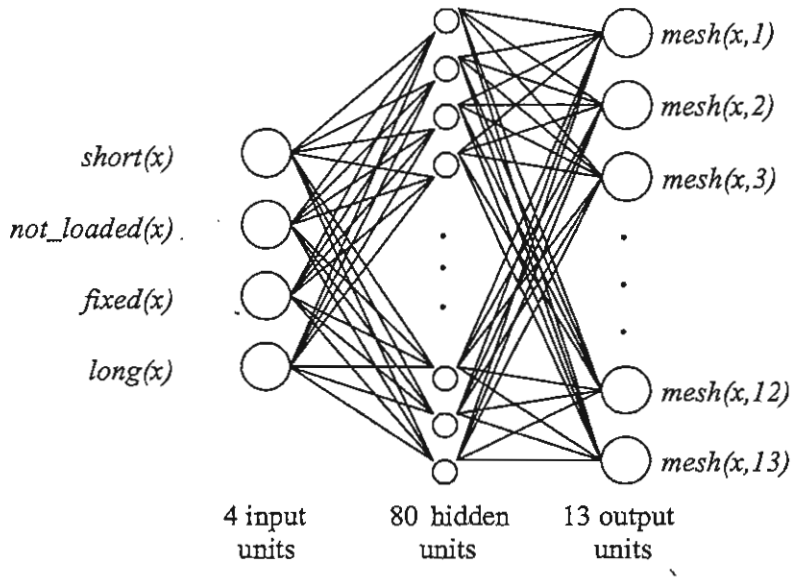


Fig. 5. The constructed network created from the inputs in Fig. 4.

ment mesh design is to learn rules for describing how many elements should be used to model each ge of a structure. Each example is of the form *mesh(Edge,Number_of_elements)*, where *Edge* is an ge label and *Number_of_elements* indicates the number of partitions. *Number_of_elements* is a number tween 1 and 13 and in our case is the class label. Figure 4 shows an example of background knowledge d training data.

As shown in the figure, there are two examples of class 1, i.e., *mesh(A1,1)*[1] and *mesh(A2,1)*, and one ample of class 2, i.e., *mesh(A3,2)*. Background knowledge contains four predicates, which are *short, t_loaded, fixed* and *long*, all of which have arity one. We apply the above definition to create an initial work. If we restrict only one variable for being used in the network construction, each predicate of ty one will be represented by one unit in the input layer. Therefore, four input units are created. The mber of nodes in the hidden layer in this case is set to 80 (determined by the experiment). Moreover, output layer contains 13 nodes each of which is for the corresponding class (class 1 to class 13). To mmarize, the constructed network will have 4 input units, 80 hidden units and 13 output units. The tained network is shown in Fig. 5. In addition, all network weights are initialized to small random mbers. The completely constructed network then receives examples for refining the network. The ocess for feeding examples to the network is described in the next subsection.

Now consider background knowledge containing predicates having arity two for learning the concept *sh* as shown in Fig. 6.

[1] Throughout the paper, symbols starting with uppercase letters designate constants, whereas those starting with lowercase ignate variables.

- **background knowledge:** short(A1), not_loaded(A1), fixed(A1),
  short(A2), fixed(A2),
  not_loaded(A3), long(A3), opposite(A3,A4),
  equal(A3,A5),
  short(A4), not_loaded(A4), free(A4),
  opposite(A4,A3),
  long(A5), free(A5), equal(A5,A3)
- **examples of class 1:** mesh(A1,1), mesh(A2,1)
- **example of class 2:** mesh(A3,2)

Fig. 6. Inputs with relational literals in background knowledge.

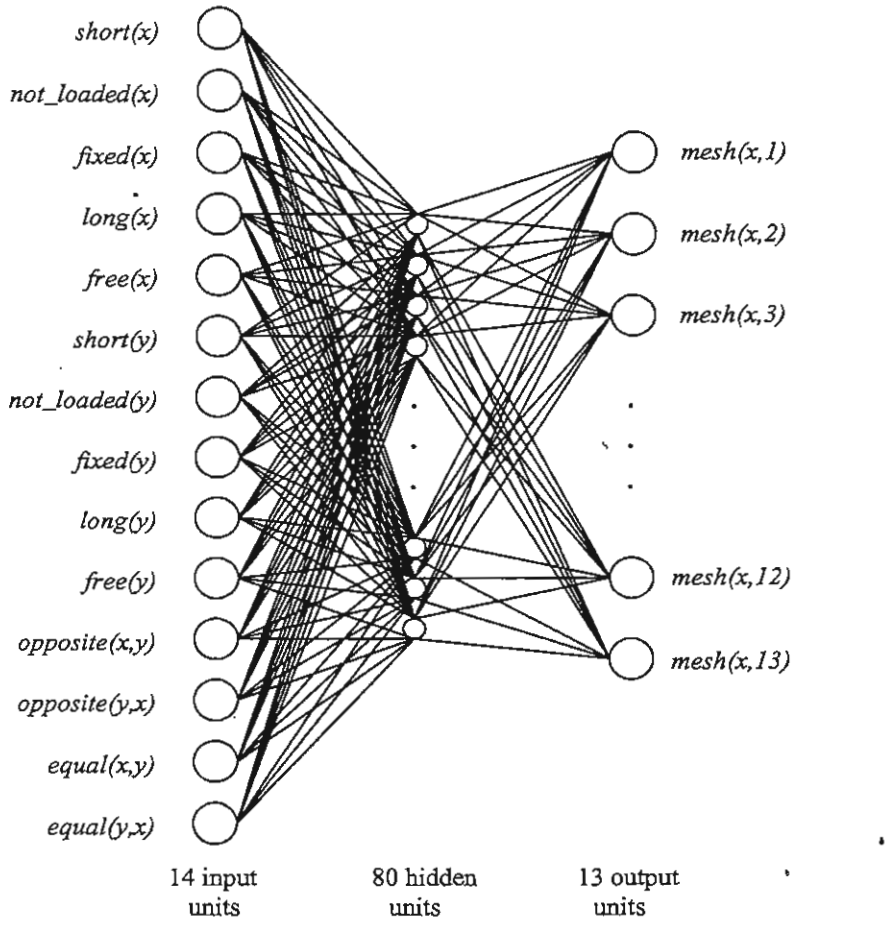


Fig. 7. The network constructed from the inputs in Fig. 6.

In these inputs, background knowledge contains the same predicates as in Fig. 4, which are *short, not_loaded, fixed* and *long*, and three new predicates *free, opposite* and *equal.* Among these new predicates, *opposite* and *equal* have arity two. If we set the number of restricted variables to 2, then each predicate of arity one is represented by two units and the predicate *opposite* is expanded to four literals which are *opposite*(*x,x*), *opposite*(*x,y*), *opposite*(*y,x*) and *opposite*(*y,y*). Nevertheless, we can definitely prune *opposite*(*x,x*) and *opposite*(*y,y*) and the other two units remain. In the same way, we will also have two literals for predicate *equal.* Therefore, in this case, the constructed network will have 14 input units in total. The hidden and output units are the same as the network in Fig. 5. The obtained network is shown in Fig. 7.

Table 1
Input and target values for the examples in Fig. 4 and the network in Fig. 5

	short(x)	*not_loaded*(x)	*fixed*(x)	*long*(x)	*mesh*(x,1)	*mesh*(x,2)
mesh(A1,1)	1	1	1	0	1	0
mesh(A2,1)	1	0	1	0	1	0
mesh(A3,2)	0	1	0	1	0	1

. Transforming inputs

.1. Feeding examples to the network

n general, neural networks receive inputs in real value form while inputs of the ILP system (background owledge and examples) are in logical form. Therefore, we transform the logical inputs to the form that be processed by the neural network. Each training example of the network is composed of inputs for neurons in the input layer and the target values for output neurons.

The training examples are fed into the network one by one and independently transformed to the work inputs as follows. The value for each input unit is defined as:

$$X_{ij} = \begin{cases} 1 & \text{if } L_i\theta_j \text{ is true in background knowledge} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

ere X_{ij}, L_i, and θ_j are input value for input unit i when feeding example j, literal represented by ut unit i, and variable substitution for example j, respectively. The input value for an input unit will 1 if there exists substitution that makes the truth value of the literal true in background knowledge. herwise the input value for that unit is 0. In addition, the target value for an output unit is defined as:

$$T_{kj} = \begin{cases} 1 & \text{if } L_k\theta_j \text{ is a positive example} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

ere T_{kj} and L_k are target value for output unit k when feeding example j, and literal represented by tput unit k, respectively.

For instance, with the same inputs in Fig. 4, if we feed positive example *mesh*(A1,1) to the network Fig. 5, units *short*(x), *not_loaded*(x), *fixed*(x) and *long*(x) will receive 1, 1, 1 and 0 as their inputs spectively, because when variable x is substituted by *A1*, the literals *short*(A1), *not_loaded*(A1) and *ed*(A1) are true in background knowledge, while the literal *long*(A1) is false. The target value for the tput unit of class 1 (*mesh*(x,1)) is 1 because *mesh*(A1,1) is a positive example of class 1. In addition, other example of class 1, *mesh*(A2,1), gives 1, 0, 1, 0 for the input units and 1 for the output unit of ss 1. The example of class 2, *mesh*(A3,2) gives 0, 1, 0, 1 for the input units and 1 for the output unit class 2. The input and the target values of each example are summarized in Table 1. In the table, we not show the value of the other output units of class 3 to class 13 as they are all 0.

Learned hypotheses from symbolic rule-based systems may contain new variables not occurring in the ad of the rule. The new variables are used to define relations between target concept arguments. For ample, rule "*mother*(x,y) ← *father*(z,y), *husband*(z,x)" means that if z is father of y and z is husband x then x will be mother of y. Here x, y and z are variables and can be substituted by any person. w variable z is created to define another person who does not appear in the head of the rule, and used in predicates *father* and *husband* to define relation between x and y. This causes the problem determining the correct substitutions for new variables for constructing certain input values of the work.

Table 2
Input value for each substitution for *mesh(A3,2)*

Input value	*opposite*(x,y)	*opposite*(y,x)	*equal*(x,y)	*equal*(y,x)
Substitute x by *A3*, and y by *A1*	0	0	0	0
Substitute x by *A3*, and y by *A2*	0	0	0	0
Substitute x by *A3*, and y by *A3*	0	0	0	0
Substitute x by *A3*, and y by *A4*	1	1	0	0
Substitute x by *A3*, and y by *A5*	0	0	1	1

To illustrate the problem, consider again the inputs and the network in Figs 6 and 7. The input values can be simply determined for the neurons of literals having no new variables, i.e., *short*(x), *not loaded*(x), *fixed*(x), *long*(x) and *free*(x). However, it is not easy to determine the input values for the last nine neurons, containing variable y since the definite substitutions are only made for the variables in the head of the rule while no certain substitutions are made for the other variables (relational variables) which are not defined in examples. In this case, we know only which constant must be substituted into variable x but not to variable y because there are many possible constants that can be substituted into y. Actually, correct substitutions are unknown which makes definite inputs cannot be created.

For example, if example *mesh(A3,2)* is fed into the network, variable x in unit *opposite*(x,y) is certainly substituted by *A3*. However, it is quite ambiguous by which term (*A1, A2, A3, A4* or *A5*) variable y should be replaced. If we select *A4* for the substitution, this literal will become *opposite(A3,A4)* which is true in background knowledge and the truth value for this input unit will be 1. However, if we select *A1* for the substitution, the literal will be false. Table 2 shows all possible substitutions and the truth values of the input nodes. From the above example, the input value for unit *opposite*(x,y) is not certain and cannot be simply resolved for network training. This problem occurs when background knowledge contains relational data and the learner cannot determine the appropriate value for the variable substitution.

To solve this problem, we apply the power of Multiple-Instance Learning (MIL) to provide input data for our network. The detail of MIL is described in the following subsection.

2.2.2. *Multiple-instance learning*

At present, the majority of work in machine learning falls into three learning frameworks i.e., supervised learning, unsupervised learning and reinforcement learning. In supervised learning, the algorithm attempts to learn a concept that correctly classifies the labels of the training examples and generalizes to predict correct labels of examples outside of the training set. Note that the labels of training data are already defined by the external teacher. In unsupervised learning, on the other hand, training examples are not labeled. Unsupervised learning tries to learn the structure of the underlying source of the examples. Some examples of unsupervised learning are clustering and Principal Component Analysis (PCA) [5]. In the last learning style, reinforcement learning [15], the agent adapts his decision process based on environmental feedback result to learn a policy, which can map states to actions potentially. Examples are not labeled with the correct action; instead an occasional reinforcement signal denoting the utility of some state is received.

Although these three frameworks have many efficient algorithms and theoretical tools to analyze them, there are many learning problems that do not fall into any of these established frameworks. Specifically, situations in which the examples are ambiguously labeled tend to be difficult for these frameworks. These situations are called learning from ambiguous examples.

Multiple-Instance Learning, which is a form of semi-supervised learning, is a new framework for learning from ambiguity. In the MIL framework [6,14], the training set is composed of a set of bags, each of which is a collection of different numbers of instances. Each instance is described by a vector

Table 3
Transformation of example *mesh(A3,2)* into input data of FOLNN

bag of example *mesh(A3,2)*	Substitute x by *A3*, and y by *A1*	Substitute x by *A3*, and y by *A2*	Substitute x by *A3*, and y by *A3*	Substitute x by *A3*, and y by *A4*	Substitute x by *A3*, and y by *A5*
short(x)	0	0	0	0	0
not_loaded(x)	1	1	1	1	1
fixed(x)	0	0	0	0	0
long(x)	1	1	1	1	1
free(x)	0	0	0	0	0
short(y)	1	1	0	1	0
not_loaded(y)	1	0	1	1	0
fixed(y)	1	1	0	0	0
long(y)	0	0	1	0	1
free(y)	0	0	0	1	1
opposite(x,y)	0	0	0	1	0
opposite(y,x)	0	0	0	1	0
equal(x,y)	0	0	0	0	1
equal(y,x)	0	0	0	0	1

atures. Like supervised learning, each example is labeled. Unlike supervised learning, an example ot a simple feature vector, but it is a set of instances. A bag (an example) is labeled as a negative if all the instances in it are negative. On the other hand, if a bag contains at least one positive nce then it is labeled as a positive bag. With this concept, we define FOLNN training data as a f training examples $\{B_1, B_2, \ldots B_n\}$, where n is the number of examples including positive and tive ones. A bag is labeled as a positive bag if an example is positive, and negative otherwise (in i-class classification, all bags are labeled as positive of their classes). The positive bag is given its target value and the negative bag is assigned 0, as defined in Eq. (2). Each bag contains m_i nces $\{B_{i1}, B_{i2}, \ldots, B_{imi}\}$ where B_{ij} is an instance with one possible binding (substitution). This is y important key because now we can use all cases of variable substitutions as one bag for learning; fore the appropriate value selection would not be a problem. Consider an example of positive bag *(A3,2)* as input data (see Table 3).

shown in Table 3, the bag *mesh(A3,2)* has five instances each of which is one case of substitution stitute y by *A1, A2, A3, A4* or *A5*). Also, the bags *mesh(A1,1)* and *mesh(A2,1)* have five instances as as the positive bag *mesh(A3,2)*. Note that the size of input vector of each instance is 14, equal to umber of units in the input layer. For training, these bags are fed to the network one by one and the ork weights are adapted by a modified version of the Backpropagation (BP) algorithm as described next subsection. Note that if there is no new variable in the input neurons, then each bag example ins only one instance and this problem can be reduced to supervised learning.

Training the network

is is the last step of the FOLNN algorithm for adapting network weights to correctly classify training ples. To train the network, the constructed training bags are fed to the network. Weight adaptation sed on a verion of the BP algorithm modified for MIL [31], which we will refer to as MIL-BP. -BP employs gradient descent to attempt to minimize the squared error between the network output s and the target values. Moreover, the activation function we use is sigmoid function, based on a thed and differentiable threshold function. Suppose the network has p input units, o output units, ne hidden layer. Given this specification, the global error function (E) of the network is defined as

follows.

$$E = \sum_{i=1}^{n} E_i \tag{3}$$

where E_i is the error on bag i. E_i is defined according to the type of bag i as:

$$E_i = \begin{cases} \min\limits_{1 \leqslant j \leqslant m_i} \sum\limits_{k=1}^{o} E_{ijk} & \text{if } B_i = + \\ \max\limits_{1 \leqslant j \leqslant m_i} \sum\limits_{k=1}^{o} E_{ijk} & \text{if } B_i = - \end{cases} \tag{4}$$

$$E_{ijk} = \begin{cases} 0 & \text{if } (B_i = +) \text{ and } (0.5 \leqslant o_{ijk}),\ l_{ik} = 1 \\ 0 & \text{if } (B_i = -) \text{ and } (o_{ijk} < 0.5), \text{ for all } k \\ \frac{1}{2}(l_{ik} - o_{ijk})^2 & \text{otherwise} \end{cases} \tag{5}$$

where

- E_{ijk} is error of output unit k on instance j in bag example i,
- $B_i = +$ is a positive bag example,
- $B_i = -$ is a negative bag example,
- o_{ijk} is actual output of output unit k from bag example i, instance j, and
- l_{ik} is target output of output unit k from bag example i.

With the defined error function above, the MIL-BP algorithm is adapted for training FOLNN. In each training epoch, the training bags created from the previous process are fed to the network one by one. Then the error E_{ijk} is computed according to Eq. (5). If the network classifies an instance correctly, the error for that instance is 0; otherwise, the error is half the squared differences between the target output l_{ik} and the unit output o_{ijk}. For a positive bag B_i, if E_{ijk} is 0 then all the rest of instances of this bag are disregarded, and the weights are not changed for this epoch because we assume that the correct positive instance is found. Otherwise the process continues and when all instances of B_i are fed, the error of bag E_i is computed by using Eq. (4) and the weights in the network are changed according to the weight update rule of the standard BP [25]. Then the next bag is fed to the network and the training process is repeated until one of the following two stopping criteria is satisfied: (1) the number of training iterations increases to some predefined threshold; (2) the global error E in Eq. (3) decreases to some predefined threshold. After having been trained, the refined network can be employed to classify unseen data.

3. Results

In the previous section, the three steps of the FOLNN algorithm were described. In this section, we evaluate FOLNN by performing experiments on the ILP problems and also compare the results obtained by FOLNN with those obtained by an ILP system. The experiments are divided into two subsections, testing with original data and noisy data.

7. Datasets

To evaluate the proposed method, FOLNN, two standard datasets for testing ILP systems are used in r experiments, i.e., the finite element mesh design and the mutagenesis datasets. The detail of each ataset is described as follows.

7.1. Finite element mesh design

The finite element method, widely used by engineers and other modelers, is a way of analyzing stresses physical structures which are represented quantitatively as finite collections of elements. The goal f finite element mesh design [8] is to learn general rules describing how many elements should be ed to model each edge of a structure. The dataset for the finite element mesh design consists of 5 uctures and has 13 classes (13 possible number of partitions for an edge in a structure). Additionally, ere are 278 examples each of which is of the form *mesh(Edge, Number_of_elements)* where *Edge* is edge label (unique for each edge) and *Number_of_elements* indicates the number of partitions. The ckground knowledge defines some of the factors that influence the resolution of a finite element design. contains relations describing the types of an edge (e.g. *circuit, short*), boundary conditions (e.g. *free, ed*), loadings (e.g. *not_loaded, one_side_loaded*) and the relations describing the structure of the object .g. *neighbour, opposite*).

7.2. Mutagenesis

This problem is a two-class learning problem for predicting the mutagenicity of the molecules, ether a molecule is active or inactive in terms of mutagenicity. It is important as it is relevant to understanding and prediction of carcinogenesis. The dataset for the mutagenesis [27] consists of 8 molecules, of which 125 are mutagenic (active) and 63 are non-mutagenic (inactive). A molecule described by listing its atoms as *atom(AtomID,Element,Type,Charge)* and the bonds between atoms as *nd(Atom1, Atom2, BondType)*. Furthermore, there are also four attributes provided for analysis of the mpounds i.e., hydrophobicity (*logP*), energy level of the lowest unoccupied molecular orbit (*LUMO*) l two boolean attribute identifying compounds with three or more benzyl rings and compounds termed enthryls (*l1* and *la*). Note that *LogP* and *LUMO* are real-valued attributes, and these real values are ectly used as inputs to FOLNN without being transformed to boolean values as the neural network ability to process real value data. The users have to specify which attributes are the real-valued ones.

8. PROGOL

The ILP system used to compare with our method is PROGOL, a state-of-the-art ILP system [24]. ing positive examples, negative examples, background knowledge and *mode declarations* as inputs, OGOL produces a most specific rule for a random seed example. The mode declarations specify, for h argument of each predicate, the type of the argument and whether it should be a constant, a variable nd before the predicate is called, or a variable bound by the predicate. PROGOL performs a complete ch of the hypothesis space bounded below by the most specific rule, using A*-like search. Because ie ability of its complete search, PROGOL requires more time and memory than many ILP systems. ertheless, PROGOL usually works well with two-class problems, positive and negative classes. For ti-class problems, we train PROGOL to induce a set of rules for each class. Positive examples of one are treated as negative examples of the other class.

Table 4
The percent accuracies of FOLNN and PROGOL on first-order datasets; FEM – Finite Element Mesh Design, MUTA – Mutagenesis

Dataset	FOLNN	PROGOL
FEM	59.18	57.80
MUTA	88.27	84.58[1]

Table 5
Performance comparison on the noisy mutagenesis dataset

Noise level in dataset	PROGOL 0% noise setting	PROGOL 10% noise setting	PROGOL 15% noise setting	FOLNN
10%	64.23[3]	69.72[3]	71.29[2]	84.01
15%	60.56[4]	61.54[3]	65.31[3]	81.28

3.3. Experiments

We performed three-fold cross validation [19] on each dataset. The dataset is randomly partitioned into three roughly equal-sized subsets with roughly same proportion of classes as that of the original dataset. Each subset is used as a test set once, and the remaining subsets are used as the training set. The training set is employed to train the network as described in Section 2, and then the trained network is used to classify test data. The output class corresponds to the output unit that gives the maximum value. The final result is the average result over three-fold data. For each fold, the momentum of the BP algorithm is set to 0.97, and the learning rate is 0.0001. The accuracies of PROGOL are computed by assigning the majority class and negative class to uncovered examples in the case of multi-class and two-class problems, respectively. Also we calculate confidence levels for the difference in accuracy between FOLNN and PROGOL by using a one-tailed paired t test. Superscripts in Tables 4 and 5 denote the calculated results: [1] is 98.0%, [2] is 99.0%, [3] is 99.5%, [4] is 99.75% and no superscripts denote confidence levels below 90.0%.

3.3.1. Results on first-order datasets

For the finite element mesh design dataset, the constructed network contains 130 units in the input layer (determined by predicates in background knowledge), 13 output units (as the number of classes) and one hidden layer with 80 hidden units (determined by the experiment). For the mutagenesis dataset, the constructed network has 235 input units, 100 hidden units and 2 output units. Note that the network is fully connected in both cases. The weights of the networks are randomly initialized and then adapted by using MIL-BP.

Our proposed method, FOLNN, has been compared with PROGOL [24], the state-of-the-art ILP system. The average results over three-fold data on FEM and MUTA datasets are summarized in Table 4. The experimental results show that the accuracies of our proposed method, FOLNN, are better than PROGOL in both datasets. These results are according to nature of learned rules generated by PROGOL. The induced rules are represented in the form "$H \leftarrow (L_1 \wedge L_2 \wedge \ldots \wedge L_n)$". If any one of preconditions of the rule, e.g. L_i, is false, the whole rule is then false; this is the weakness of the rule. On the other hand, FOLNN does not have this problem as it is able to learn the importance of each literal L_i in terms of the network weights.

We also did experiments by using the *standard* backpropagation algorithm to adjust weights of the network; we consider each instance as one bag. The accuracies of the neural network using the standard

BP are 74.33% and 39.74% for the datasets MUTA and FEM, respectively. This shows that MIL-BP is n effective algorithm for training our FOLNN.

3.2. Results on a noisy dataset

As mentioned before, ILP systems have the restriction to handle noisy data. To see how well our arner handles noisy data, we evaluate FOLNN on noisy datasets, in addition to the results on the iginal dataset. The mutagenesis dataset is selected for this task. Using the three-fold data of the utagenesis dataset in the last experiment, 10% and 15% class noise is randomly added into the training t, but no noise is added into the test set. In our case, adding $x\%$ of noise means that the target class lue is replaced with the wrong value in x out of 100 data by random selection. Suppose that if the ass value is 1 then it will be changed to 0 and vice versa. The accuracies of PROGOL and FOLNN on isy datasets are shown in Table 5.

Since PROGOL has an ability to handle noise in data as its option, "$x\%$ noise setting" in the table ecifies that noise option of PROGOL is set to $x\%$. This parameter sets an upper bound on the rcentage of negative examples allowed to be covered by an acceptable clause. As can be seen in the ble 5, our proposed algorithm still provides average accuracies higher than PROGOL. Although, we n set the configuration for PROGOL to handle noisy data, its accuracy drops fast, compared to that of)LNN. When 10% noise is added into the dataset, PROGOL provides the highest accuracy of 71.29% ile FOLNN provides 84.01%. Furthermore, PROGOL gives 65.31% as its maximum accuracy in the % noise dataset, while FOLNN gives 81.28%. The PROGOL performance significantly drops due to sensitivity to noise which is the main disadvantage of first-order rules directly induced by the ILP stem. When noise is added, the learned rule is overfitting to the noise, and thus the rule contains er-specific literals that are not true in general. The over-specific literals decrease the performance of learned rules. However, the accuracy of our method decreases much slower and is much higher than t of PROGOL. FOLNN, because of the ability of noise tolerance by combining with neural networks, nore robust against noise than the original first-order rules. FOLNN prevents overfitting noisy data employing neural networks to give higher weights to important features and give less attention to mportant ones.

Conclusions

earning first-order logic programs by using neural networks is still an open problem. In this paper, present a novel hybrid connectionist symbolic system based on the standard feedforward neural work that incorporates inductive learning from examples and background knowledge, called FOLNN st-Order Logical Neural Network). The contribution of this paper is twofold. Firstly, for research eural networks, the paper shows the successful application of the neural network to the learning of -order logic programs. Secondly, for ILP research, the paper demonstrates a new type of learning ems which is based on the robust neural networks. We believe that by integrating the neural network 1 inductive learning from examples and background knowledge, the combined system contains the ity of noise tolerance from connectionist systems and the ability of using background knowledge 1 ILP systems. Therefore FOLNN competently alleviates the problem of first-order rules induced he ILP system which are not robust enough to noisy or unseen data. The prominent advantage of .NN is that it can learn from inputs provided in form of first-order logic programs directly. Except LP systems, other learners cannot directly learn from this kind of programs as they cannot select the

appropriate values for variable substitution, but our method can solve this problem by applying MIL to learn from ambiguous variable substitutions.

In the experiments, FOLNN shows its efficiency in comparison with the well-known ILP system, PROGOL. FOLNN presents the ability of noise tolerance and produces the better performance than PROGOL. This is because of the ability of the neural network that outperforms PROGOL in the presence of noisy data by giving higher weights to important attributes and less attention to unimportant ones. Although our main objective is to learn the first-order logic, FOLNN can be applied to other tasks such as learning from propositional datasets containing missing values in some attributes.

One interesting issue in the field of intelligent hybrid system is knowledge extraction. Knowledge extraction from a trained network is one phase of the neural-symbolic learning system. It is the phase that is to translate the refined concepts which are in form of neural networks into human readable rules. Knowledge extraction is of significant interest in data mining and knowledge discovery applications such as medical diagnosis. However, in this work we do not attentively focus on this step and have not yet explored rule extraction from trained networks. Nevertheless, we surmise that many researches [1,7,10, 28], can be adapted to extract rules from our networks.

Acknowledgement

This work was supported by the Thailand Research Fund. We would like to thank the anonymous referees for their valuable comments.

References

[1] R. Andrew, J. Diederich and A.B. Tickle, Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks, *Knowledge-Based Systems* **8** (1995), 373–389.

[2] R. Basilio, G. Zaverucha and V.C. Barbosa, Learning Logic Programs with Neural Networks, in: *Proceedings of the 11th International Conference on Inductive Logic Programming, number 2157 in Lecture Notes in Artificial Intelligence,* C. Rouveirol and M. Sebag, eds, Springer, 2001, pp. 15–26.

[3] C.M. Bishop, *Neural Networks for Pattern Recognition,* Oxford University Press, 1995.

[4] M. Botta, A. Giordana and R. Piola, *FONN: Combining First Order Logic with Connectionist Learning,* Proceedings of the 14th International Conference on Machine Learning, Nashville, TN, 1997, 48–56.

[5] C. Chatfield and A.J. Collins, *Introduction to Multivariate Analysis,* London: Chapman and Hall, 1980.

[6] Y. Chevaleyre and J.D. Zucker, *A Framework for Learning Rules from Multiple Instance Data,* Proceedings of the 12th European Conference on Machine Learning, number 2167 in Lecture Notes in Computer Science, Freiburg, Germany, 2001, 49–60.

[7] M.W. Craven, *Extracting Comprehensible Models from Trained Neural Networks,* Department of Computer Science: University of Wisconsin-Madison, 1996.

[8] B. Dolsak and S. Muggleton, The Application of Inductive Logic Programming to Finite Element Mesh Design, in: *Inductive Logic Programming,* S. Muggleton, ed., Academic Press, 1992, pp. 453–472.

[9] S. Dzeroski, S. Schulze-Kremer, K.R. Heidtke, K. Siems and D. Wettschereck, *Applying ILP to Diterpene Structure Elucidation from 13C NMR Spectra,* Proceedings of the 6th International Workshop on Inductive Logic Programming, number 1314 in Lecture Notes in Artificial Intelligence, Berlin: Springer-Verlag, 1996, 14–27.

[10] A.S.d.A. Garcez, K.B. Broda and D.M. Gabbay, *Neural-Symbolic Learning Systems,* London: Springer-Verlag, 2002.

[11] P. Hitzler and A.K. Seda, Continuity of Semantic Operators in Logic Programming and Their Approximation by Artificial Neural Networks, in: *KI 2003: Advances in Artificial Intelligence, 26th Annual German Conference on AI,* A. Guenter, R. Kruse and B. Neumann, eds, Berlin: Springer-Verlag, 2003, pp. 355–369.

[12] S. Hölldobler, Y. Kalinke and H.P. Störr, Approximating the Semantics of Logic Programs by Recurrent Neural Networks, *Applied Intelligence* **11** (1999), 45–58.

[13] S. Hölldobler and Y. Kalinke, *Towards a Massively Parallel Computational Model for Logic Programming,* Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing (ECCAI), Amsterdam, The Netherlands, 1994, 68–77.

X. Huang, S.C. Chen and M.L. Shyu, *An Open Multiple Instance Learning Framework and Its Application in Drug Activity Prediction Problems,* Proceedings of the 3rd IEEE Symposium on BioInformatics and BioEngineering (BIBE'03), Bethesda, Maryland, 2003, 53–59.
L.P. Kaelbling, M.L. Littman and A.W. Moore, Reinforcement Learning: A Survey, *Journal of Artificial Intelligence Research* 4 (1996), 237–285.
B. Kijsirikul, S. Sinthupinyo and K. Chongkasemwongse, Approximate Match of Rules Using Backpropagation Neural Networks, *Machine Learning* **44** (2001), 273–299.
N. Lavrac and S. Dzeroski, *Inductive Logic Programming Techniques and Applications,* New York: Ellis Horwood, 1994.
J.J. Mahoney and R.J. Mooney, Combining Connectionist and Symbolic Learning to Refine Certainty-Factor Rule-Bases, *Connection Science* **5** (1993), 339–364.
T.M. Mitchell, *Machine Learning,* New York: The McGraw-Hill Companies Inc., 1997.
S.H. Nienhuys-Cheng and R.d. Wolf, *Foundation of Inductive Logic Programming,* New York: Springer-Verlag, 1997.
R. Parekh and V. Honavar, *Constructive Theory Refinement in Knowledge Based Neural Networks,* Proceedings of the International Joint Conference on Neural Networks, Anchorage, Alaska, 1998, 2318–2323.
J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* San Francisco: Morgan Kaufmann Publishers Inc., 1988.
J.R. Quinlan, *C4.5:Programs for Machine Learning,* San Francisco: Morgan Kaufmann Publishers Inc., 1993.
S. Roberts, *An Introduction to Progol,* Technical Manual: University of York, 1997.
D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning Internal Representations by Error Propagation, in: *Parallel Distributed Processing,* (Vol. 1), D.E. Rumelhart and J.L. McClelland, eds, Cambridge, MA: The MIT Press, 1986, pp. 318–362.
L. Shastri and V. Ajjanagadde, From Simple Associations to Systematic Reasoning: A Connectionist Representation of Rules, Variables, and Dynamic Bindings Using Temporal Synchrony, *Behavioral and Brain Sciences* **16** (1993), 417–494.
A. Srinivasan, S.H. Muggleton, M.J.E. Sternberg and R.D. King, Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction, *Artificial Intelligence* **85** (1996), Elsevier Science Publishers, 277–299.
G.G. Towell and J.W. Shavlik, The Extraction of Refined Rules from Knowledge-Based Neural Networks, *Machine Learning* **13** (1993), 71–101.
G.G. Towell and J.W. Shavlik, Knowledge-Based Artificial Neural Networks, *Artificial Intelligence* **70**(1–2) (1994), 119–165.
S. Wermter and R. Sun, An Overview of Hybrid Neural Systems, in: *Hybrid Neural Systems, number 1778 in Lecture Notes in Artificial Intelligence,* S. Wermter and R. Sun, eds, Springer, 2000, pp. 1–13.
Z.H. Zhou and M.L. Zhang, *Neural Network for Multi-Instance Learning,* Proceedings of the International Conference on Intelligent Information Technology, Beijing, China, 2002, 455–459.

Adaptive Directed Acyclic Graphs: Algorithms for Multiclass Support Vector Machines

Boonserm Kijsirikul and Thimaporn Phetkaew

Department of Computer Engineering, Chulalongkorn University, Thailand.
email: boonserm.k@chula.ac.th, thimaporn.p@student.chula.ac.th

Abstract. Constructing multiclass Support Vector Machines (SVMs) is a challenging research problem. The Decision Directed Acyclic Graph (DDAG) method reduces evaluation time, while maintaining accuracy compared to the Max Wins, which is probably the currently most accurate method for multiclass SVMs. We proposed a method, called Adaptive Directed Acyclic Graph (ADAG), which is based on the previous approach, the DDAG, and is designed to remedy some weakness of the DDAG caused by its structure. We proved that the expected accuracy of the ADAG is higher than that of the DDAG. We also proposed an enhancement version of the ADAG, called Reordering Adaptive Directed Acyclic Graph (RADAG), to find one best accuracy from all possible ADAG. The RADAG choose the optimal order of binary classifiers in nodes in the graph of the ADAG by using the reordering algorithm with minimum-weight perfect matching. We empirically evaluated our approachs by comparing the ADAG and the RADAG with the DDAG and the Max Wins on several data sets.

1 Introduction

Support Vector Machines (SVMs) have been well developed for binary classification [28]. However, many real world problems involve multiclass classification, such as optical character recognition, phoneme classification, text classification, etc. Several approaches of extending SVMs for solving the problem of multiclass classification have been proposed [28, 10, 7, 15, 18, 16, 9, 30, 22, 1, 17, 26].

The standard approaches for constructing N-class SVMs are to consider the problem as a collection of binary SVMs, such as One-Against-the-Rest [28] and One-Against-One [16]. The One-Against-the-Rest (1-v-R) approach works by constructing a set of N binary classifiers. The i^{th} classifier is trained with all of the examples in the i^{th} class with positive labels, and all other examples with negative labels. The final output is the class that corresponds to the classifier with the highest output value. On the other hand, the One-Against-One (1-v-1) approach simply constructs all possible binary classifiers from a training set of N classes. Each classifier is trained on only two out of N classes. Thus, there will be $N(N-1)/2$ classifiers. In the Max Wins algorithm [11] which is one of 1-v-1 approaches, a test example is classified by all of classifiers. Each classifier provides one vote for its preferred class and the majority vote is used to make the final output. Although Max Wins offers faster training time compared to the 1-v-R method, it is very inefficient in term of evaluation time. Using a new

learning architecture, DDAG, Platt et al. [21] proposed an algorithm that reduces evaluation time, while maintaining accuracy compared to the Max Wins. The comparison experiments in several methods on large problems in [12] show that the Max Wins and the DDAG may be more suitable for practical use.

In this paper we point out some limitations of the DDAG caused by its structure which needs an unnecessarily high number of node evaluations for the correct class and results in high cumulative error. Our two modified versions of the DDAG, the ADAG and the RADAG, will increase accuracy by minimizing the number of node evaluations for the correct class. This advantage is due to the tournament-based architecture that is structurally flatter than the DDAG. We prove that the expected accuracy of the ADAG is higher than that of the DDAG, and also empirically evaluate our approaches by comparing the ADAG and the RADAG with the algorithm of the DDAG and the Max Wins on several data sets from UCI Machine Learning Repository [4] and Thai printed character data set [20].

2 Decision Directed Acyclic Graphs

A disadvantage of the 1-v-1 SVMs is their inefficiency of classifying data as the number of SVMs grows superlinearly with the number of classes. Platt et al. introduced a new learning architecture, DDAG, and an algorithm, DAGSVM, to remedy this disadvantage [21].

2.1 The DDAG Architecture & the DAGSVM Algorithm

A Directed Acyclic Graph (DAG) is a graph whose edge has an orientation and no cycles. Platt et al. used a rooted binary DAG, which has a unique node with no arcs pointing into it, and other nodes which have either 0 or 2 arcs leaving them, to be a class of functions used in classification tasks. In a problem with N classes, a rooted binary DAG has N leaves labeled by the classes where each of the $N(N-1)/2$ internal nodes is labeled with an element of Boolean function. The nodes are arranged in a triangular shape with the single root node at the top, two nodes in the second layer and so on until the final layer of N leaves.

To evaluate a DDAG, starting at the root node, the binary function at a node is evaluated. The node is then exited via the left edge, if the binary function is -1; or the right edge, if the binary function is 1. The next node's binary function is then evaluated. The value of the decision function is the value associated with the final leaf node (see Figure 1). Only $N-1$ decision nodes will be evaluated in order to derive an answer. The DDAG can be implemented using a list, where each node eliminates one class from the list. The implementation list is initialized with a list of all classes. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the other class is eliminated from the list, and the DDAG proceeds to test the first and last elements of the new list. The DDAG terminates when only one class remains in the list.

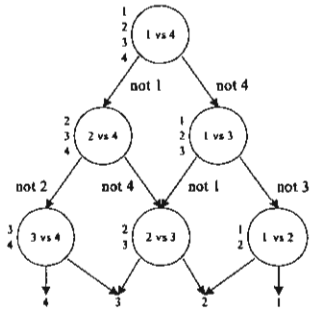


Fig. 1. The DDAG finding the best class out of four classes.

The DAGSVM algorithm creates a DDAG whose nodes are maximum margin classifiers over a kernel-induced feature space. Such a DDAG is obtained by training each '*i vs j*' node only on the subset of training points labeled by i or j. The final class decision is derived by using the DDAG architecture. For the DAGSVM, the choice of the class order in the list (or DDAG) is arbitrary.

2.2 Issues on DDAG

Systematically innovated, the DDAG has outperformed the standard algorithm in terms of speed. However, the DDAG has the main weakness that the number of node evaluations for the correct class is unnecessarily high. This results in high cumulative error and, hence, the accuracy. The depth of the DDAG is $N-1$ and this means that the number of times the correct class has to be tested against other classes, on average, scales linearly with N. Let consider a case of 20-class problem. If the correct class is evaluated at the root node, it is tested against other classes for 19 times before it is correctly classified as the output. Despite large margin, there exists probability of misclassification, let say 1%, and this will cause $1 - 0.99^{19} = 17.38\%$ of cumulative error in this situation. This shortcoming becomes more severe if the number of classes increases. The issue raised here motivates us to modify the DDAG.

3 Adaptive Directed Acyclic Graphs

In this section we introduce a new approach to alleviate the problem of the DDAG structure. The new structure, the Adaptive DAG, lowers the depth of the DAG, and consequently the number of node evaluations for the correct class.

3.1 The ADAG Architecture

An Adaptive DAG (ADAG) is a DAG with a reversed triangular structure. In an N-class problem, the system comprises $N(N-1)/2$ binary classifiers. The ADAG has $N-1$ internal nodes, each of which is labeled with an element of

Boolean function. The nodes are arranged in a reversed triangle with $N/2$ nodes (rounded up) at the top, $N/2^2$ nodes in the second layer and so on until the lowest layer of a final node, as shown in Figure 2(a).

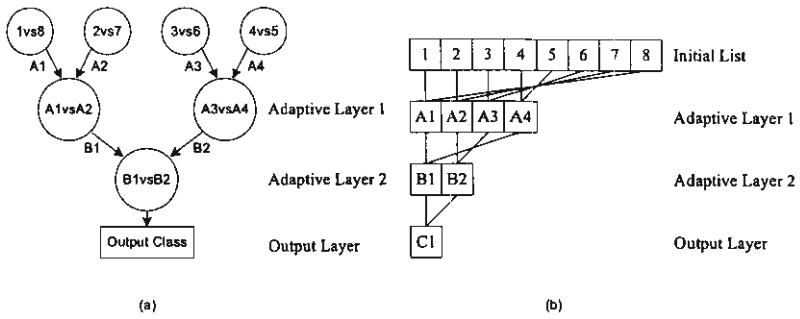


Fig. 2. (a) The structure of an adaptive DAG for an 8-class problem, and (b) the corresponding implementation through the list.

To classify using the ADAG, starting at the top level, the binary function at each node is evaluated. The node is then exited via the outgoing edge with a message of the preferred class. In each round, the number of candidate classes is reduced by half. Based on the preferred classes from its parent nodes, the binary function of the next-layer node is chosen. The reduction process continues until reaching the final node at the lowest level. The value of the decision function is the value associated with the message from the final leaf node (see Figure 2(a)). Like the DDAG, the ADAG requires only $N-1$ decision nodes to be evaluated in order to derive an answer. Note that the correct class is evaluated against other classes for $log_2 N$ times (rounded up) or less, considerably lower than the number of evaluations required by the DDAG, which scales linearly with N.

3.2 Implementation

An ADAG can be implemented using a list, where each node eliminates one class from the list (see Figure 2(b)). The implementation list is initialized with a list of all classes. A test point is evaluated against the decision node that corresponds to the first and last elements of the list. If the node prefers one of the two classes, the class is kept in the left element's position while the other class will be eliminated from the list. Then, the ADAG proceeds to test the second and the elements before the last of the list. The testing process of each round ends when either one or no class remains untested in the list. In case that there is one class remaining untested, the class will be put at the right-most position of the next round list. After each round, a list with N elements is reduced to a list with $N/2$ elements (rounded up). Then, the ADAG process repeats until only one class remains in the list.

4 Analyses of DDAG & ADAG

The following theorems give the expected accuracy of the DDAG and ADAG. The proofs of the theorems will be given in Appendix.

Theorem 1. *Let p be the probability that the correct class will be eliminated from the implementation list, when it is tested against another class, and let the probability of one of any two classes, except for the correct class, being eliminated from the list be 0.5. Then under a uniform distribution of the position of the true class in the list, the expected accuracy of the DDAG is $(1/N)[(1-p)/p + (1-p)^{N-1} - (1-p)^N/p]$, where N is the number of classes.*

Proof. See Appendix.

Theorem 2. *Let p be the probability that the correct class will be eliminated from the implementation list, when it is tested against another class, and let the probability of one of any two classes, except for the correct class, being eliminated from the list be 0.5. Then under a uniform distribution of the position of the true class in the list, the expected accuracy of the ADAG is $((2N - 2^{\lceil log_2 N \rceil})/N)(1-p)^{\lceil log_2 N \rceil} + ((2^{\lceil log_2 N \rceil} - N)/N)(1-p)^{\lceil log_2 N \rceil - 1}$, where N is the number of classes, and $\lceil x \rceil$ is the least integer greater than or equal to x.*

Proof. See Appendix.

The above theorems show that the accuracy of the ADAG decreases much slower than that of the DDAG, with the increase of the number of classes. For example, in case of $p = 0.01$ and $N = 20$, according to the above theorems, the expected accuracy of the ADAG and DDAG are 95.68% and 90.18%, respectively.

According to the theorems mentioned above, the ADAG should have much advantage over the DDAG when the number of classes increases. We evaluated the performance of the ADAG. Figure 3 shows the trends of the accuracy of classification of the DDAG and the ADAG when the number of classes is varying. The experiment is based on the English letter image recognition which has 26 classes [4]. The dataset is trained by using Polynomial kernel degree 4. For the DDAG, the number of evaluations for the correct class is unnecessary high. This results in higher cumulative error and lower the accuracy. Using the reversed triangular structure, the ADAG reduces the number of times the correct class is tested against other classes, and thus reduces the cumulative errors. This greatly improves the performance of the DDAG. The improvement comes in the form of higher accuracy with higher confidence of achieving the accuracy. Our approach is empirically proved to increase accuracy and confidence, especially in problems with the higher number of classes.

5 Reordering Adaptive Directed Acyclic Graphs

The accuracy of a binary classifier depends heavily on its generalization performance. In this section we introduce a method to select an optimal order of

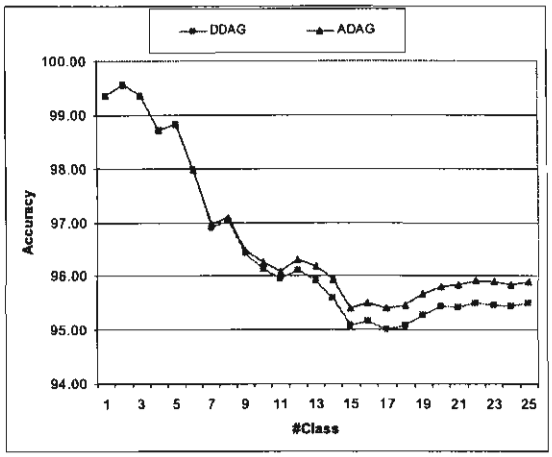


Fig. 3. The trends of the classification accuracy when the number of classes is varying.

classes in the list (binary classifiers) by dynamically reordering the order of classes during classification process according to each test data. Only binary classifiers which have small generalization errors will be used in data classification. We called this method the Reordering Adaptive Directed Acyclic Graph (RADAG).

5.1 The RADAG Architecture

The structure of the ADAG and the RADAG are the same. But the differences are the initialization of the binary classifiers in the top level and the order of classes in lower levels (see Figure 4). In the first step, we use a reordering algorithm with minimum-weight perfect matching described in the next subsection to choose the optimal order of classes to be the initial order. We use the order to evaluate every test example. In the second step, as in the ADAG, test points of the RADAG are evaluated against the decision nodes. In the third step, unlike the ADAG, the RADAG will reorder the order of classes before processing in the next level by using the reordering algorithm with minimum-weight perfect matching. This order differs for each test example, and it depends on the results of nodes from the previous level. The second and the third steps are repeated until there is only one class remains.

5.2 The Reordering Algorithm

In this subsection we describe the reordering algorithm with minimum-weight perfect matching.

5.3 Generalization Error of Classifiers

The ability of a hypothesis to correctly classify data not in the training set is known as its generalization [8]. Generalization analysis of pattern classifiers is

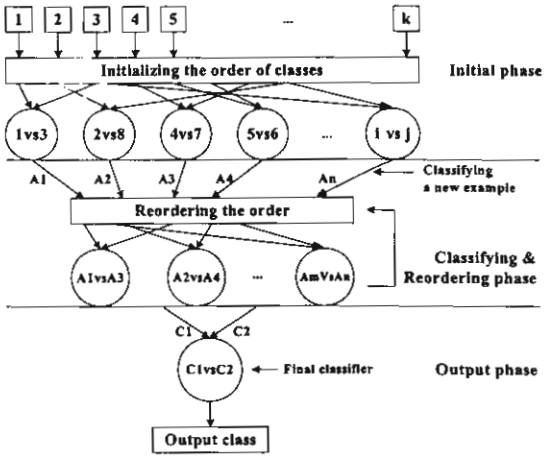


Fig. 4. Classifying process of the RADAG.

concerned with determining the factors that affect the accuracy of a pattern classifier [3]. Generalization performance of SVMs can be approximated by bounding on the generalization error. Define the class F of real-valued functions on the ball of radius R in $\mathbb{R}^n$ as $F = \{\mathbf{w} \mapsto \mathbf{w} \cdot \mathbf{x} : ||\mathbf{w}|| \leq 1, ||\mathbf{x}|| \leq R\}$. There is a constant c such that, for all probability distributions, with probability at least $1 - \delta$ over l independently generated examples $\mathbf{z}$, if a classifier $h = sgn(f) \in sgn(F)$ has margin at least γ on all the examples in $\mathbf{z}$, then the error of h is no more than

$$\frac{c}{l}\left(\frac{R^2}{\gamma^2}log^2 l + log\left(\frac{1}{\delta}\right)\right) \tag{1}$$

Furthermore, in case that the training data cannot be separated by the hyperplane without error, with probability at least $1 - \delta$, every classifier $h \in sgn(F)$ has error no more than

$$\frac{k}{l} + \sqrt{\frac{c}{l}\left(\frac{R^2}{\gamma^2}log^2 l + log\left(\frac{1}{\delta}\right)\right)} \tag{2}$$

where k is the number of labeled examples in $\mathbf{z}$ with margin less than γ. Below we show an example of the generalization error of classifiers. The experiment is based on the English letter image recognition data set from [4], which has 26 classes. Hence there are 325 classifiers. The classifiers are trained by using the Polynomial kernel of degree 3. In Figure 5, the generalization errors of all classifiers expressed by Equation 2 are depicted. The generalization errors of all classifiers are varying. The average of all of them is 28.82.

5.4 Algorithm

For the reason described above, we consider the generalization errors in order to choose the optimal order with less chance to predict the wrong class from

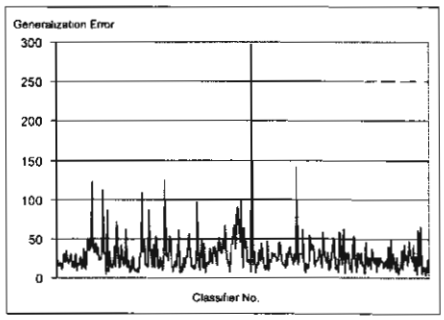


Fig. 5. The generalization errors of 325 classifiers.

all possible $\frac{N!}{2^{\lfloor N/2 \rfloor}\lfloor N/2 \rfloor!}$ orders. Among $N(N-1)/2$ classifiers, $N/2$ classifiers which have small generalization errors will be considered to be used in data classification.

Let $G = (V, E)$ be a graph with node set V and edge set E. Each node in G denotes one class and each edge denotes one binary classifier which has a generalization error from equation 2 (see Figure 6(a)). The output of the reordering algorithm for graph G is a subset of edges with the minimum sum of generalization errors of all edges and each node in G is met by exactly one edge in the subset (see Figure 6(b)). Given a real weight c_e being generalization error for each edge e of G, the problem of reordering algorithm can be solved by the minimum weight perfect matching that finds a perfect matching M of minimum weight $\sum(c_e : e \in M)$.

For $U \subseteq V$, let $E(U) = \{(i,j) : (i,j) \in E, i \in U, j \in U\}$. $E(U)$ is the set of edges with both endpoints in U. The set of edges incident to node i in the node-edge incidence matrix is denoted by $\delta(i)$. The convex hull of perfect matchings on a graph $G = (V, E)$ with $|V|$ even is given by
a) $x \in R_+^m$
b) $\sum_{e \in \delta(v)} x_e = 1$ for $v \in V$
c) $\sum_{e \in E(U)} x_e \leq \lfloor \frac{|U|}{2} \rfloor$ for all odd sets $U \subseteq V$ with $|U| \geq 3$
or by (a),(b) and
d) $\sum_{e \in \delta(U)} x_e \geq 1$ for all odd sets $U \subseteq V$ with $|U| \geq 3$
where $|E| = m$ and $x_e = 1$ means that e is in the matching. So the minimum weight of a perfect matching is at least as large as the value of

$$min \sum_{e \in E} c_e x_e \tag{3}$$

where x satisfies "(a),(b) and (c)" or "(a),(b) and (d)". Therefore, the reordering problem is to solve the linear program in Equation 3.

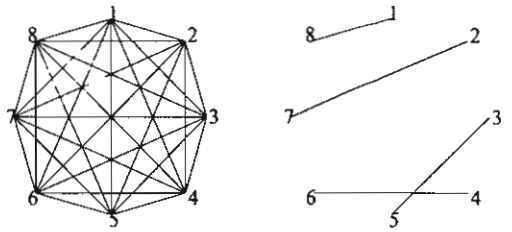


Fig. 6. (a) A graph for an 8-class problem (b) An example of the output of the re-ordering algorithm.

5.5 Estimating the Difference between Means of Generalization Errors

In this subsection we will explain test concerning means of generalization errors of classifiers of interest (the binary classifier of the correct class and other classes in the order of classes), which are selected by the ADAG and the RADAG. For the ADAG, we randomly selected the classifiers to be used in data classification, whereas for the RADAG the classifiers are selected by using the minimum-weight perfect matching algorithm. This may be analyzed using a method called the one tailed paired t test.

To estimate the difference between two means, we wish to test

$$H_0 : \mu_1 - \mu_2 = 0,$$
$$H_1 : \mu_1 - \mu_2 > 0.$$

The point estimate of the difference $\mu_1 - \mu_2$ of two population means is given by $\bar{X}_1 - \bar{X}_2$. The form of the paired t test is

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_d/\sqrt{n}} \tag{4}$$

where S_d is the standard deviation of the sample of differences $\bar{X}_1 - \bar{X}_2$.

Assume that the distribution of generalization errors is normal distribution. The simulation is based on the dataset which has 26 classes, and hence there are 325 binary classifiers. In the experiment, a generalization error is randomly selected to each classifier with equal probability, uniformly distributed. We examine 5,200 orders of classifiers, where for the first 200 orders we assume that the correct class is class 1, for the second 200 orders we assume that the correct class is class 2 and so on. For the ADAG, we randomly selected 5,200 orders of classifiers. For the RADAG, 5,200 orders are selected by using the minimum-weight perfect matching algorithm so all orders are the same. Then we compute sum of generalization errors of classifiers of interest that correspond to the binary classifier of the correct class and other classes in the order. The experiments are repeated 100 times. Then we calculate the sample mean $\bar{X}_1$ ($\bar{X}_{ADAG}$) and $\bar{X}_2$

($\bar{X}_{RADAG}$). Using $\alpha = 0.001$ with a one-tailed test and degree of freedom = 99, the null hypothesis is rejected. This means that the RADAG obtains classifiers whose generalization errors are statistically significantly lower than the ADAG.

6 Experiments

In the experiments, we compared the accuracy of classification of four algorithms, i.e., the Max Wins, the DDAG, the ADAG, and the RADAG. We also compared the computational time between the ADAG, the RADAG and the Max Wins.

6.1 Data Set and Experimental Setting

The experiments are based on several data sets from the UCI Machine Learning Repository [4] including glass, satimage, segment, shuttle, vowel, soybean, letter and isolet (see Table 1). For the soybean problem, we discarded the last four classes because of missing values. In addition, we examined our methods with Thai printed character recognition [20], which has 68 classes including 44 consonants, and 26 vowels and tonal masks. For the training set, the characters were printed by laser printer with 600 dpi resolution. Then they were copied by a copier machine with saturated ink. There are two test sets. For the test set of Thai printed character 1 data set, the charecters were printed by laser printer with 600 dpi resolution. For the test set of Thai printed character 2 data set, the characters were printed by laser printer with 600 dpi resolution. Then they were copied by a copier machine with pale ink. These data sets are different in the number of classes, the number of dimensions, and sizes. 68 classes is the most classes and 617 dimensions is the most dimensions which are used in the experiment.

In these experiments we scaled both training data and test data to be in [-1,1] and employed Polynomial and RBF kernels. In the training phase, the $N(N-1)/2$ binary classifiers were constructed by using the software package called SVM^{light} version 5.0 [13, 14]. For the DDAG and the ADAG, we examined all possible orders of classes for datasets having not more than 8 classes, whereas we randomly selected 50,000 orders for datasets having more than 8 classes. We then calculated the average of accuracy of these orders.

Table 1. Description of the datasets used in the experiments.

Dataset	Training data	Test data	Classs	Dimension
Glass	214	-	6	9
Satimage	4,435	2,000	6	36
Segment	2,310	-	7	18
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35
Letter	15,963	4,037	26	16
Isolet	6,238	1,559	26	617
ThaiPrintedCharacter1	3,264	3,264	68	128
ThaiPrintedCharacter2	3,264	3,264	68	128

6.2 Experimental Results

Table 2 and Table 3 present the results of comparing four methods including the Max Wins, the DDAG, and our methods, the ADAG and the RADAG. We present the optimal parameters, d in the Polynomial kernel $|(\mathbf{x} \cdot \mathbf{y} + 1)|^d$ and c in the RBF kernel $exp(-|\mathbf{x} - \mathbf{y}|^2/c)$. The best accuracy among these methods is illustrated in bold-face. ****, ***, ** and * in the tables mean 99.99%, 99%, 95%, and 90% confidence interval for estimating the difference between accuracies of the ADAG and other methods, using a one-tailed paired t-test. ++++, +++, ++ and + mean 99.99%, 99%, 95%, and 90% confidence interval for estimating the difference between accuracies of the RADAG and other methods using a one-tailed paired t-test.

To estimate the difference between accuracies, we added up the training set and test set into one set. Then we use a k-fold cross-validation method in which the set is partitioned into k disjoint, equal-sized subsets. In this k-fold cross-validation approach, each example from the set is used exactly once in a test set, and k-1 times in a training set [19]. In our experiment, we use 5-fold crsss-validation for the glass dataset and 10-fold crsss-validation for all others.

For estimating the difference between errors of two learning methods, the mean difference $\bar{Y}$ in errors from all disjoint subsets is returned as an estimate of the difference between the two learning algorithms [19]. The approximate $N\%$ confidence interval for estimating the difference using $\bar{Y}$ is given by

$$(\bar{Y} - t_{N,k-1} S_{\bar{Y}}, \bar{Y} + t_{N,k-1} S_{\bar{Y}}) \tag{5}$$

where $\bar{Y}$ is the sample mean defined as

$$\bar{Y} = \frac{1}{k} \sum_{i=1}^{k} Y_i$$

Y_i is the difference in error between two learning methods from the i^{th} subset, and $S_{\bar{Y}}$ is the estimated standard deviation of the sample mean defined as

$$S_{\bar{Y}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} \left(Y_i - \bar{Y}\right)^2}.$$

As shown in the tables, our methods yield higher accuracy than the Max Wins and the DDAG in almost all of datasets. The results show that the ADAG performs statistically significantly better than the DDAG in satimage, shuttle, vowel, soybean and letter problems. In case of the RBF kernel, the ADAG performs statistically significantly better than the DDAG in shuttle, soybean, letter and isolet problems. The results also show that the RADAG performs statistically significantly better than the other methods in the segment, shuttle, vowel and letter problems in case of the Polynomial kernel. In case of the RBF kernel, the RADAG performs statistically significantly better than the other methods in the glass, shuttle, vowel, soybean and letter problems. These show the effectiveness of the ADAG and the RADAG.

6.3 Computational Time

Table 4 and Table 5 present the comparison of the running time for classifying test data between the ADAG, the RADAG and the Max Wins for Polynomial

Table 2. A comparison of the accuracy of classification using the Polynomial kernel.

Dataset	d	Max Wins	d	DDAG	d	ADAG	d	RADAG
Glass	2	71.078	2	71.069	2	71.135	2	71.063
Satimage	6	89.615	6	89.599^{***}	6	89.622	6	**89.681**
Segment	8	97.351_{++}	8	97.360_{++}	8	97.383_{+}	8	**97.533**
Shuttle	8	99.923_{+}	8	99.919^{**}_{++}	8	99.922_{++}	8	**99.930**
Vowel	3	98.901_{+}	3	98.872^{*}_{+}	3	98.894_{+}	2	**98.990**
Soybean	3	92.470	5	92.202^{*}	5	92.281	3	**92.698**
Letter	3	96.512_{+}	3	95.994^{*****}_{++++}	3	96.379_{+++}	4	**96.674**
Isolet	3	97.488	3	97.484	3	97.485	3	**97.499**
ThaiPrintedCharacter1	2	**99.246**	2	99.239	2	99.242	2	99.234
ThaiPrintedCharacter2	2	**99.677**	2	99.657	2	99.670	2	99.617

Table 3. A comparison of the accuracy of classification using the RBF kernel.

Dataset	c	Max Wins	c	DDAG	c	ADAG	c	RADAG
Glass	0.09	73.238_{+}	0.08	72.850_{+}	0.08	72.759_{+}	0.09	**74.319**
Satimage	3.0	92.148	3.0	92.129	3.0	92.141	3.0	**92.152**
Segment	0.7	**97.656**	0.7	97.652	0.7	97.650	0.7	97.576
Shuttle	3.0	99.928	3.0	99.926^{***}_{++}	3.0	99.927_{+}	3.0	**99.931**
Vowel	0.2	98.980_{+}	0.2	98.965_{+}	0.2	98.975_{+}	0.2	**99.091**
Soybean	0.08	92.533_{+}	0.07	91.739^{**}_{++}	0.08	92.570_{+}	0.07	**93.016**
Letter	3.0	96.512_{+}	3.0	95.994^{*****}_{++++}	3.0	96.379_{++++}	3.0	**96.634**
Isolet	0.01	97.527	0.01	97.517^{*}	0.01	97.523	0.003	**97.589**
ThaiPrintedCharacter1	0.003	**99.387**	0.003	**99.387**	0.003	**99.387**	0.003	**99.387**
ThaiPrintedCharacter2	0.004	**99.663**	0.004	**99.663**	0.004	**99.663**	0.004	**99.663**

and RBF kernels by using a 400 MHz Pentium II processor. There is no running time of the glass dataset because it has too few test examples to measure the time. For the segment problem, there is no provided test data so we use 5-fold crsss-validation. The results show that both the ADAG and the RADAG require low running time in all data sets, especially when the number of classes and/or the number of dimensions are relatively large. For an N-class problem, the Max Wins requires $N(N-1)/2$ classifiers for the classification whereas the ADAG and the RADAG require only $N-1$ classifiers. Hence the larger the number of classes the more running time the Max Wins requires than the ADAG and the RADAG. Moreover, the number of dimensions affects the running time of each classifier. As the result, the larger the number of dimensions the more running time the Max Wins requires than the ADAG and the RADAG. For the RADAG, the number of classes affects the running time for reordering. However, it takes a little time even when there are many classes.

The Max Wins needs $O(N^2)$ number of comparisons for the problem with N classes. The DDAG reduces the number of comparisons down to $O(N)$. By reducing the depth of the path, the ADAG requires $O(N)$ comparisons of binary classifiers with accuracy higher than that of the DDAG. The RADAG needs a little time more than the ADAG for reordering the order of classes. Note that, currently, the minimum-weight perfect matching algorithm, which is used in the reordering algorithm, runs in time bounded by $O(N(M + NlogN))$ [6], where N is the number of nodes (classes) in the graph and $M = N(N-1)/2$ is the number of edges (binary classifiers). The RADAG will reorder the order of classes in every level, except for the last level. The order of classes in the top level is

Table 4. A comparison of the computational time using the Polynomial kernel.

Dataset	Test data	Class	Dimension	d	RADAG			Max Wins (seconds)
					ADAG (seconds)	Reordering (seconds)	Total (seconds)	
Satimage	2,000	6	36	6	1.90	0.50	2.40	9.47
Segment	462	7	18	8	0.11	0.08	0.19	0.41
Shuttle	14,500	7	9	8	1.75	3.38	5.13	5.15
Vowel	462	11	10	2	0.12	0.25	0.37	0.61
Soybean	340	15	35	3	0.30	0.25	0.55	1.86
Letter	4,037	26	16	4	8.48	4.20	12.68	125.58
Isolet	1,559	26	617	3	116.02	1.48	117.50	1,671.98
ThaiPrintedCharacter1	3,264	68	128	3	94.63	13.83	108.46	2,996.64
ThaiPrintedCharacter2	3,264	68	128	3	96.41	13.19	109.60	3,042.98

Table 5. A comparison of the computational time using the RBF kernel.

Dataset	Test data	Class	Dimension	c	RADAG			Max Wins (seconds)
					ADAG (seconds)	Reordering (seconds)	Total (seconds)	
Satimage	2,000	6	36	3.0	11.75	0.51	12.26	37.13
Segment	462	7	18	0.7	0.24	0.10	0.34	0.82
Shuttle	14,500	7	9	3.0	3.36	0.63	3.99	9.27
Vowel	462	11	10	0.2	0.10	0.27	0.37	0.61
Soybean	340	15	35	0.07	0.32	0.45	0.77	2.20
Letter	4,037	26	16	3.0	62.27	3.69	65.96	802.85
Isolet	1,559	26	617	0.01	100.42	1.60	102.02	1,369.11
ThaiPrintedCharacter1	3,264	68	128	0.002	86.25	16.46	102.71	2,772.18
ThaiPrintedCharacter2	3,264	68	128	0.001	55.75	14.37	70.12	1,877.77

reordered only one time and we use the order to evaluate every test example. Hence for classifying each test data, we need $log_2 N - 2$ times of reordering, where each time the number of classes is reduced by half. Therefore, the running time of the RADAG is bounded by $O(c_1 N) + O(c_2 N^3 log_2 N)$, where c_1 is much larger than c_2.

7 Conclusion

We have proposed a new approach, Adaptive Directed Acyclic Graph (ADAG), that alleviates the problem of the DDAG caused by its structure which needs an unnecessarily high number of evaluations for the correct class. Using the reversed triangular structure, the ADAG reduces the number of times the correct class is tested against other classes, and thus reduces the cumulative errors. We proved that the expected accuracy of the ADAG is higher than that of the DDAG.

We also proposed an enhancement version of the ADAG, Reordering Adaptive Directed Acyclic Graph (RADAG), to choose an optimal order of classes in the list in the ADAG method. By the use of minimum-weight perfect matching, the RADAG can reorder the order of classes in polynomial time and only binary classifiers which have small generalization errors will be considered to be used in decision nodes.

Our experimental results are also evidence that the ADAG and the RADAG yield higher accuracy of classification than the Max Wins and the DDAG, especially in such a case that the number of classes is relatively large. Moreover, the

running time used by the ADAG and the RADAG is much less than that used by the Max Wins, especially when the number of classes and/or the number of dimensions are relatively large.

Since the DDAG reduces evaluation time while maintaining accuracy compared to the Max Wins, which is one of the SVMs' fastest methods in multiclass classification, this modification of the DDAG will help improve accuracy even further. In our ongoing work, we are studying how to reduce the number of evaluations and how to enhance the performance in terms of both accuracy and running time.

Appendix

In the following analyses of the DDAG and ADAG, we assume that the probability of the correct class being in any position in the list is a uniform distribution. We also assume that the probability of the correct class being eliminated from the list is p, when it is tested against another class, and that the probability of one of any two classes, except for the correct class, being eliminated from the list is 0.5 when they are tested against each other.

We first illustrate the expected accuracy of the DDAG by the following example.

Example: (Expected accuracy of the DDAG for a 4-class problem). Consider a four-class problem. Figure 7 shows all *probability calculation paths* where the correct class will be correctly classified by the DDAG. There are 8 calculation paths for this problem. The correct class will be correctly classified if it is not eliminated from the list. This means that when it is at the edge (the first or the last element) of the list in each calculation path, all other classes have to be excluded from the list.

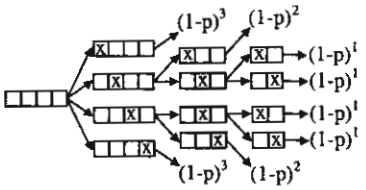


Fig. 7. An example of a four-class problem.

Under a uniform distribution, the probability is 1/4 that the correct class will be at any position of the *initial* list. In the case that the correct class (indicated by 'X' in the figure) is at the edge of the current list, it will be correctly classified if all other classes are eliminated from the list. The probability of this is $(1-p)^{N-1}$, where N is the number of elements in the current list. In the case that the correct class is not at the edge, we have two possible choices, i.e. to remove the first element and to remove the last element from the list. This reduces the number of elements one by one. From the above example, the probability that the correct class is correctly classified is:

$(1/4)(1-p)^3 + (1/4)(1/2)^1(1-p)^2 + (1/4)(1/2)^2(1-p)^1 + (1/4)(1/2)^2(1-p)^1 +$
$(1/4)(1/2)^2(1-p)^1 + (1/4)(1/2)^2(1-p)^1 + (1/4)(1/2)^1(1-p)^2 + (1/4)(1-p)^3$
$= (1/4)[(1-p)/p + (1-p)^3 - (1-p)^4/p]$ □

We now give the theorem for expected accuracy of the DDAG as follows.

Proof of Theorem 1 (Expected accuracy of the DDAG). As shown in the above example, the correct class will be correctly classified if when it is at the edge of the list, all other classes have to be excluded from the list. Consider the cases when we *first* obtain a list with i elements where the correct class is at the left-most position, where $2 \leq i \leq N-1$. The list can be written as $XO \ldots O$, where X and O represent the correct class and a wrong class, respectively. This list is obtained from a list containing $i+1$ elements with one wrong class preceding the correct class as shown by $OXO \ldots O$. Before the list $OXO \ldots O$ is obtained, $N-i-1$ wrong classes must be excluded from an initial list. Thus beginning from all possible different initial lists, the number of possible calculation paths ending with a list of i elements where the correct class is at the left-most position is 2^{N-i-1} (as there are two possible choices for one wrong class; to remove the first element or to remove the last element of the list). Therefore the number of possible calculation paths ending with a list of i elements where the correct class is at the edge (the left-most and right-most positions) is 2^{N-i}. The probability of obtaining the list of i elements where the correct class is at the edge is equal to $(1/N)(0.5)^{N-i}(1-p)^{i-1}$, as $N-i$ wrong classes must be eliminated from the list and after that the correct class is at the edge and $i-1$ wrong classes must be excluded. The total probability that the correct class of this pattern will be correctly classified is thus $2^{(N-i)}(1/N)(0.5)^{N-i}(1-p)^{i-1} = (1/N)(1-p)^{i-1}$.

Next consider the cases when the correct class is at the edge of a list with N elements. For these cases, the total probability that the correct class will be correctly classified is obviously $2(1/N)(1-p)^{N-1}$.

Finally we sum up all above probabilities and we have the expected accuracy as:
$(\sum_{i=2}^{N-1}(1/N)(1-p)^{i-1}) + 2(1/N)(1-p)^{N-1} = (\sum_{j=1}^{N-1}(1/N)(1-p)^{j}) + (1/N)(1-p)^{N-1}$
$= (1/N)[(1-p)(1-(1-p)^{N-1})/p + (1-p)^{N-1}] = (1/N)[(1-p)/p + (1-p)^{N-1} - (1-p)^N/p]$
□

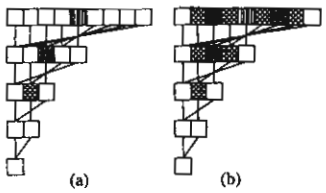


Fig. 8. The positions of classes that can be *bye-getting* elements.

Proof of Theorem 2 (Expected accuracy of the ADAG). Given N classes of examples, the height (the number of adaptive layers and the output layer) of the ADAG is obviously $\lceil log_2 N \rceil$. To be selected as the *winner* (as the output of the ADAG), some elements have to be compared with others for $\lceil log_2 N \rceil$ times, and there are some elements, called *bye-getting* elements that are compared with others for less than $\lceil log_2 N \rceil$ times. As the architecture of the ADAG always puts a bye-getting element (the

middle element) of the current list at the edge of the list of the next layer, any element can get at most one bye. Therefore, a bye-getting element will be compared with others for $\lceil log_2 N \rceil - 1$ times. There will be bye-getting elements only when the number of classes cannot be represented by 2^X, where X is a positive integer. These bye-getting elements will be at the middle of the initial list and of the current list representing each adaptive layer; e.g. the 5^{th}, 3^{rd} and 2^{nd} elements in the initial list, the list of the first adaptive layer, and the list of the second adaptive layer in Figure 8(a), respectively. A bye-getting element at the i^{th} adaptive layer can come from *two* elements in the $(i-1)^{st}$ layer, as shown in Figure 8(b). Therefore, 7 elements of the initial list in the figure can possibly be bye-getting ones. Let $F(N)$ be the number of all possible bye-getting elements of the list with N elements. It is obvious that with $F(2) = 0$, and

$$F(N) = (N \ mod \ 2) + 2 \cdot F(\lceil N/2 \rceil). \tag{6}$$

Next we will prove by induction on X that $F(N) = 2^{\lceil log_2(N) \rceil} - N$,when N is an integer between $2^X + 1$ and 2^{X+1},and X is a positive integer greater than or equal to 1.

First consider the base case of $X = 1$. In this case, N is equal to 3 or 4. It is obvious that $F(3) = 1$ which satisfies $F(3) = 2^{\lceil log_2(3) \rceil} - 3$, and $F(4) = 0$ satisfying $F(4) = 2^{\lceil log_2(4) \rceil} - 4$.

Next we prove general cases of $X \geq 2$ by induction: suppose $F(N)$ is equal to $2^{\lceil log_2(N) \rceil} - N$ when N is an integer between $2^{X-1} + 1$ and 2^X, and we will show that $F(M)$ is also equal to $2^{\lceil log_2(M) \rceil} - M$ when M is an integer between $2^X + 1$ and 2^{X+1}.

In case of $M = 2N$ (an even number) and $2^X + 2 \leq M \leq 2^{X+1}$, it is clear that $F(M) = 2F(N)$ according to Equation 6 (in case that M is an even number, a bye-getting elements of $F(N)$ can possibly come from two bye-getting elements of $F(M)$). Therefore, we will have the following.

$$\begin{aligned} F(M) &= 2F(N) = 2(2^{\lceil log_2 N \rceil} - N) = 2^{\lceil log_2 N \rceil + 1} - 2N = 2^{\lceil log_2 2N \rceil} - 2N \\ &= 2^{\lceil log_2 M \rceil} - M \end{aligned}$$

This proves the case of M is an even number.

Next in case of $M = 2N - 1$ (an odd number) and $2^X + 1 \leq M \leq 2^{X+1} - 1$, it is also clear that $F(M) = 2F(N) + 1$ according to Equation 1 (in case that M is an odd number, a bye-getting elements of $F(N)$ can possibly come from two bye-getting elements of $F(M)$, and there is one more bye-getting (the middle) elements of $F(M)$ that gets a bye of this round). Therefore, we will have the following.

$$\begin{aligned} F(M) &= 2F(N) + 1 = 2(2^{\lceil log_2 N \rceil} - N) + 1 = 2^{\lceil log_2 N \rceil + 1} - 2N + 1 \\ &= 2^{\lceil log_2 2N \rceil} - (2N - 1) \end{aligned}$$

As $2^X + 1 \leq 2N - 1 < 2N \leq 2^{X+1}$, we have $log_2(2^X + 1) \leq log_2(2N - 1) < log_2(2N) \leq log_2(2^{X+1})$, which means $\lceil log_2(2N - 1) \rceil = \lceil log_2(2N) \rceil = X + 1$. The above formula then becomes as follows.

$$F(M) = 2^{\lceil log_2(2N-1) \rceil} - (2N - 1) = 2^{\lceil log_2 M \rceil} - M$$

This proves the case of M is an odd number. The above then proves that $F(N)$ is equal to $2^{\lceil log_2(N) \rceil} - N$ when N is an integer between $2^X + 1$ and 2^{X+1}, where $X \geq 1$.

Having the value of $F(N)$ as above, we then can calculate the expected accuracy of the ADAG. As under a uniform distribution, the probability that the correct class is at any position of the initial list is $1/N$. Therefore, the expected accuracy is calculated by weighting the bye-getting correct elements with $F(N)/N$, and the non-bye-getting correct elements with $(N - F(N))/N$. Finally, we have the expected accuracy of the ADAG as follows.

$$\begin{aligned} &(N - F(N))/N \cdot (1-p)^{\lceil log_2 N \rceil} + F(N)/N \cdot (1-p)^{\lceil log_2 N \rceil - 1} \\ &= ((2N - 2^{\lceil log_2 N \rceil})/N) \cdot (1-p)^{\lceil log_2 N \rceil} + ((2^{\lceil log_2 N \rceil} - N)/N) \cdot (1-p)^{\lceil log_2 N \rceil - 1}. \end{aligned}$$

This proves the theorem. □

References

1. Abe, S., and Inoue, T. (2002) *Fuzzy support vector machines for multiclass problems*, The European Symposium on Artificial Neural Networks, 113-118.
2. Allwein, E., Schapire, R., and Singer, Y. (2000) *Reducing multiclass to binary: A unifying approach for margin classifiers*, International Conference on Machine Learning.
3. Bartlett, P. L., and Shawe-Taylor, J. (1999) *Generalization performance of support vector machines and other pattern classifiers*, Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, USA, 43-54.
4. Blake, C., Keogh, E., and Merz, C. (1998) *UCI repository of machine learning databases*, Department of Information and Computer Science, University of California, Irvine.
5. Burges, C. (1998) *A tutorial on support vector machines for pattern recognition*, Data Mining and Knowledge Discovery, 2(2):121-167.
6. Cook, W., and Rohe, A. (1997) *Computing minimum-weight perfect matchings*, Technical Report 97863, Forschungsinstitut für Diskrete Mathematik, Universität Bonn.
7. Crammer, K., and Singer, Y. (2001) *On the algorithmic implementation of multiclass kernel-based vector machines*, Journal of Machine Learning Research, 2:265-292.
8. Cristianini N., and Shawe-Taylor, J. (2000) *An introduction to support vector machines and other kernel-based learining methods*, Cambridge University Press.
9. Dietterich, T. G., and Bakiri, G. (1995) *Solving multiclass learning problems via error-correcting output codes*, Journal of Artificial Intelligence Research, 2:263-286.
10. Dong, X., Zhaohui, W., and Yunhe, P. (2001) *A new multi-class support vector machines*, IEEE International Conference on System, Man, and Cybernatics, 3:1673-1676.
11. Friedman, J. H. (1996) *Another approach to polychotomous classification*, Technical report, Stanford University, Department of Statistics.
12. Hsu, C., and Lin, C. (2002) *A comparison of methods for multiclass support vector machines*, IEEE Transactions on Neural Networks, 13:415-425.
13. Joachims. T. (1998) *Making large-scale SVM learning practical*, Advances in Kernel Methods - Support Vector Learning, MIT Press.
14. Joachims, T. (1999) SVM^{light}, http://ais.gmd.de/~thorsten/svm_light.
15. Kindermann, J., Leopold, E., and Paass, G. (2000) *Multi-class classification with error correcting codes*, Treffen der GI-Fachgruppe 1.1.3, Maschinelles Lernen, GMD Re. 114.
16. Knerr, S., Personnaz, L., and Dreyfus, G. (1990) *Single-layer learning revisited: A stepwise procedure for building and training a neural network*, In Fogelman-Soulie and Herault, editors, Neurocomputing: Algorithms, Architectures and Applications, NATO ASI Series. Springer.
17. Li, K., Huang, H., and Tian, S. (2002) *A novel multi-class SVM classifier based on DDAG*, The 1st International Conference on Machine Learning and Cybernatics, 1203-1207.
18. Mayoraz, E., and Alpaydm, E. (1998) *Support vector machines for multi-class classification*, IDIAP-RR 6, IDIAP.
19. Mitchell, T. (1997) *Machine Learning*, McGraw Hill.
20. Pichitdej, S. (2001) *Thai Printed Character Recognition Using a Neural Network Ensemble*, Degree of Master Computer Engineering in Department of Computer Engineering Faculty of Engineering Chulalongkorn University.

21. Platt, J., Cristianini, N., and Shawe-Taylor, J. (2000) *Large margin DAGs for multiclass classification*, Advance in Neural Information Processing System, 12, MIT Press.
22. Roth, V., and Tsuda, K. (2001) *Pairwise coupling for machine recognition of hand-printed japanese characters*, IEEE International Conference on Computer Society, 1:1120-1125.
23. Schölkopf, B. (1997) *Support vector learning*, Ph.D. Thesis, R.Oldenbourg Verlag Publications, Munich, Germany.
24. Schwenker, F. (2000) *Hierarchical support vector machines for multi-class pattern recognition*, The 4th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 561-565.
25. Sekhar, C., Takeda, K., and Itakura, F. (2002) *Close-class-set discrimination method for large-class-set pattern recognition using support vector machines*, IEEE International Joint Conference on Neural Networks, 577-582.
26. Takahashi, F., and Abe, S. (2002) *Decision-tree-based multiclass support vector machines*, The 9th International Conference on Neural Information Processing, 3:1418-1422.
27. Thubthong, N. and Kijsîrikul, B. (2001) *Support vector machines for Thai phoneme recognition*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 9(6):803-813.
28. Vapnik, V. (1998) *Statistical Learning Theory*, New York, Wiley.
29. Vapnik, V. (1999) *An overview of statistical learning theory*, IEEE Transactions on Neural Networks, 10:988-999.
30. Weston, J., and Watkins, C. (1998) *Multi-class support vector machines*, Technical Report CSD-TR-98-04, Department of Computer science, Royal Holloway, University of London.

Multiclass Classification of Support Vector Machines by Reordering Adaptive Directed Acyclic Graph*

Thimaporn Phetkaew[1], Boonserm Kijsirikul[2] and Wanchai Rivepiboon[3]

Department of Computer Engineering, Chulalongkorn University, Phayathai, Patumwan, Bangkok, 10330, Thailand

Thimaporn.P@student.chula.ac.th[1] and Boonserm.K[2],Wanchai.R[3]@chula.ac.th

Received 24 November 2003

Abstract The problem of extending binary support vector machines (SVMs) for multiclass classification is still an ongoing research issue. Ussivakul and Kijsirikul proposed the Adaptive Directed Acyclic Graph (ADAG) approach that provides accuracy comparable to that of Max Wins, which is probably the currently most accurate method for multiclass SVMs, and requires low computation. However, different sequences of binary classifiers in nodes in the ADAG may provide different accuracy. In this paper we present a new method for multiclass classification, Reordering Adaptive Directed Acyclic Graph (RADAG), which is the modification of the original ADAG method. We propose an algorithm to choose an optimal sequence of binary classifiers in nodes in the ADAG by considering the generalization error bounds of all classifiers. We apply minimum-weight perfect matching with the reordering algorithm in order to select binary classifiers which have small generalization errors to be used in data classification and in order to find the best sequence of binary classifiers in nodes in polynomial time. We then compare the performance of our method with previous methods including the ADAG and the Max Wins algorithm. Experiments denote that our method gives higher accuracy. Moreover it runs faster than Max Wins, especially when the number of classes and/or the number of dimensions are relatively large.

Keywords Support Vector Machines, Multiclass Classification, ADAG, RADAG

* This work was supported by The Thailand Research Fund.

1 Introduction

Support vector machines (SVMs) [8] were primarily designed for two-class classification problems with their outstanding performance in real world applications. However, extending SVMs for multiclass classification is still an ongoing research issue. Previous methods for solving the multiclass problem of SVMs are typically to consider the problem as the combination of two-class decision functions, e.g. one-against-one and one-against-the-rest [5]. The one-against-the-rest approach works by constructing a set of k binary classifiers for a k-class problem. The i^{th} classifier is trained with all of the examples in the i^{th} class with positive labels, and all other examples with negative labels. The final output is the class that corresponds to the classifier with the highest output value. Friedman [5] suggested the Max Wins algorithm in which each one-against-one classifier casts one vote for its preferred class, and the final result is the class with the most votes. The Max Wins algorithm offers faster training time compared to the one-against-the-rest method. The Decision Directed Acyclic Graph (DDAG) method proposed by Platt et al. reduces training and evaluation time, while maintaining accuracy compared to the Max Wins [6]. The comparison experiments by several methods on large problems in [5] show that the Max Wins algorithm and the DDAG may be more suitable for practical use. Ussivakul and Kijsirikul [7] proposed the Adaptive Directed Acyclic Graph (ADAG) method which is the modification of the DDAG. This method reduces the dependency of the sequence of binary classifiers in nodes in the structure as well as lowers the number of tests required to evaluate for the correct class. Their approach yields higher accuracy and reliability of classification, especially in such a case that the number of classes is relatively large.

In this paper we reveal that the ADAG still is dependent on the sequence of its nodes, although it is less dependent on the order of binary classes in the sequence than the DDAG; there are still differences in accuracy between different sequences. This led to the reliability of the algorithm. Here we propose a novel method that improves reliability by choosing an optimal sequence, which has less chance to predict the wrong class, and dynamically reordering the sequence during classification process according to each test data. We also reveal that the problem of selecting the appropriate sequence can be solved by minimum-weight perfect matching.

This paper is organized as follows. In the next section, we review SVMs and the formulation to solve multiclass problems, i.e., the DDAG and the ADAG. In Section 3, we introduce the modification of the ADAG to improve the performance by using the reordering algorithm with minimum-weigh perfect matching. The numerical experiments are illustrated in Section 4. Finally, the conclusions are given in Section 5.

2 SVM classification

This section describes the basic idea of SVMs [7] and two previous works on multiclass SVMs which are related to our proposed method, i.e., the DDAG [5,6] and the ADAG [7].

2.1 Support vector machines

The main idea of support vector machine classification is to construct a hyperplane to separate the two classes.

2.1.1 Linear support vector machines

Suppose we have a data set D of l samples in an n-dimensional space belonging to two different classes (+1 and −1):

$$D = \{(\mathbf{x}_k, y_k) \mid k \in \{1,..,l\}, \mathbf{x}_k \in \Re^n, y_k \in \{+1,-1\}\} \quad \text{.....(1)}$$

The hyperplane in the n dimensional space is determined by the pair $(\mathbf{w},b)$ where $\mathbf{w}$ is an n-dimensional vector orthogonal to the hyperplane and b is the offset constant. The hyperplane $(\mathbf{w}\cdot\mathbf{x}) + b$ separates the data if and only if

$$(\mathbf{w}\cdot\mathbf{x}_i)+b>0 \quad \text{if} \quad y_i=+1$$
$$(\mathbf{w}\cdot\mathbf{x}_i)+b<0 \quad \text{if} \quad y_i=-1. \quad \text{.....(2)}$$

If we additionally require that $\mathbf{w}$ and b be such that the point closest to the hyperplane has a distance of $1/|\mathbf{w}|$, then we have

$$(\mathbf{w}\cdot\mathbf{x}_i)+b\geq 1 \quad \text{if} \quad y_i=+1$$
$$(\mathbf{w}\cdot\mathbf{x}_i)+b\leq -1 \quad \text{if} \quad y_i=-1 \quad \text{.....(3)}$$

which is equivalent to

$$y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b]\geq 1 \quad \forall i. \quad \text{.....(4)}$$

To find the optimal separating hyperplane, we have to find the hyperplane that maximizes the minimum distance between the hyperplane and any sample of training data. The distance between two closest samples from different classes is

$$d(\mathbf{w},b)=\min_{\{\mathbf{x}_i|y_i=1\}}\frac{(\mathbf{w}\cdot\mathbf{x}_i)+b}{|\mathbf{w}|}-\max_{\{\mathbf{x}_i|y_i=-1\}}\frac{(\mathbf{w}\cdot\mathbf{x}_i)+b}{|\mathbf{w}|}.$$
.....(5)

From (3), we can see that the appropriate minimum and maximum values are ±1. Therefore, we need to maximize

$$d(\mathbf{w},b)=\frac{1}{|\mathbf{w}|}-\frac{-1}{|\mathbf{w}|}=\frac{2}{|\mathbf{w}|}. \quad \text{.....(6)}$$

Thus, the problem is equivalent to:

- minimize $|\mathbf{w}|^2/2$
- subject to the constraints
  (1) $y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b]\geq 1 \quad \forall i.$

For non-separable case, the training data cannot be separated by a hyperplane without error. The previous constraints then must be modified. A penalty term consisting of the sum of deviations ξ_i from the boundary is added to the minimization problem. Now, the problem is to

- minimize $\frac{|\mathbf{w}|^2}{2}+C\sum_{i=1}^{l}\xi_i$
- subject to the constraints
  (1) $y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b]\geq 1-\xi_i$,
  (2) $\xi_i\geq 0 \quad \forall i.$

The penalty term for misclassifying training samples is weighted by a constant C. Selecting a large value of C puts a high price on deviations and increases computation by effecting a more exhaustive search for ways to minimize the number of misclassified samples.

By forming the Lagrangian and solving the dual problem, this problem can be translated into:

- minimize
  $$L(\mathbf{w},b,\alpha)=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j)$$
  .....(7)
- subject to the constraints:
  (1) $0\leq\alpha_i\leq C$, $\forall i$
  (2) $\sum_{i=1}^{l}\alpha_i y_i=0$

where α_i are called Lagrange multipliers. There is one Lagrange multiplier for each training sample. In the solution, those samples for which $\alpha_i > 0$ are called *support vectors*, and are ones such that the equality in (4) holds. All other training samples having $\alpha_i = 0$ could be removed from the training set without affecting the final hyperplane.

Let α^0, an l-dimensional vector denote the minimum of $L(\mathbf{w},b,\alpha)$. If $\alpha_i^0 > 0$ then $\mathbf{x}_i$ is a *support vector*. The optimal separating hyperplane $(\mathbf{w}^0, b^0)$ can be written in terms of α^0 and the training data, specifically in terms of the support vectors:

$$\mathbf{w}^0=\sum_{i=1}^{l}\alpha_i^0 y_i\mathbf{x}_i=\sum_{\text{support vectors}}\alpha_i^0 y_i\mathbf{x}_i. \quad \text{.....(8)}$$
$$b^0=1-\mathbf{w}^0\cdot\mathbf{x}_i \quad \text{for } \mathbf{x}_i \text{ with } y_i=1 \text{ and } 0<\alpha_i<C.$$
.....(9)

The optimal separating hyperplane classifies points according to the sign of $f(\mathbf{x})$,

$$f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^0 \cdot \mathbf{x} + b^0\right) = \text{sign}\left(\sum_{\text{suport vectors}} \alpha_i^0 y_i (\mathbf{x}_i \cdot \mathbf{x}) + b^0\right) \quad \ldots.(10)$$

Support vector $\mathbf{x}_i$ with $\alpha_i^0 = C$ may or may not be misclassified. All other $\mathbf{x}_i$'s are correctly classified.

2.1.2 Non-linear support vector machines

The above algorithm is limited to linear separating hyperplanes. SVMs get around this problem by mapping the sample points into a higher dimensional space using a non-linear mapping chosen in advance. This is, we choose a map $\Phi : \Re^n \mapsto \mathrm{H}$ where the dimensionality of H is greater than n. We then seek a separating hyperplane in the higher dimensional space; this is equivalent to a non-linear separating surface in $\Re^n$.

The data only ever appears in our training problem (7) in the form of dot products, so in the higher dimensional space we are only dealing with the data in the form $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$. If the dimensionality of H is very large, then this could be difficult, or very computationally expensive. However, if we have a kernel function such that $k(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$, then we can use this in place of $\mathbf{x}_i\cdot\mathbf{x}_j$ everywhere in the optimization problem, and never need to know explicitly what Φ is. Some widely used kernels are:

Polynomial degree d: $k(\mathbf{x},\mathbf{y}) = \left|\mathbf{x}\cdot\mathbf{y} + 1\right|^d$ ...(11)

Radial basis function: $k(\mathbf{x},\mathbf{y}) = e^{-\left|\mathbf{x}-\mathbf{y}\right|^2/c}$ ...(12)

2.2 DDAG

Platt et al. [6] presented a learning architecture, the Decision Directed Acyclic Graph (DDAG), which is used to combine many two-class classifiers into a multiclass classifier. For a k-class problem, its training phase is the same as the one-against-one method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, it uses a rooted binary directed acyclic graph which has $k(k\text{-}1)/2$ internal nodes and k leaves (see Figure 1). Each node is a binary SVM of the i^{th} and j^{th} classes. Given a test sample x, starting at the root node, the binary decision function is evaluated. Then it moves to either left or right depending on the output value. Therefore, we go through a path before reaching a leaf node which indicates the predicted class.

There are some issues on the DDAG as pointed out by [7]. First, it gives outputs whose probabilities are not uniformly distributed, and thus its output depends on the sequence of binary classifiers in nodes, affecting reliability of the algorithm. In addition, the correct class placed in a node near the root node is clearly at disadvantage by comparison with the correct class near leaf nodes since it is exposed to higher risk of being incorrectly rejected. Second, the number of node evaluations for the correct class is unnecessary high. This results in higher cumulative error and lower the accuracy. The depth of the DDAG is k-1 and this means that the number of times the correct class has to be tested against other classes, on average, scales linearly with k.

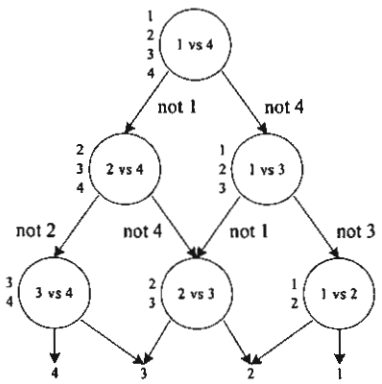


Figure 1: The DDAG finding the best class out of four classes

2.3 ADAG

Ussivakul and Kijsirikul [7] proposed an approach to alleviate the problem of the DDAG structure described above. An Adaptive DAG (ADAG) is a DAG with a reversed triangular structure. For a k-class problem, its training phase is the same as the DDAG method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, the nodes are arranged in a reversed triangle with $k/2$ nodes (rounded up) at the top, $k/2^2$ nodes in the second layer and so on until the lowest layer of a final node. It has k-1 internal nodes, each of which is labeled with an element of Boolean function (see Figure 2). Given a test example x, starting at the top level, the binary decision function is evaluated. The node is then exited via the outgoing edge with a message of the preferred class. In each round, the number of candidate classes is reduced by half. Based on the preferred classes from its parent nodes, the binary function of the next-level node is chosen. The reduction process continues until reaching the final node at the lowest level. The value of the decision function is the value associated with the message from the final leaf node. Like the DDAG, the ADAG requires only k-1 decision nodes to be evaluated in order to derive an answer. Note that the correct class is evaluated against other classes for $\log_2 k$ times or less, considerably lower than the number of evaluations required by the DDAG, which scales linearly with k.

Using the reversed triangular structure, the ADAG reduces the number of times the correct class is tested against other classes, and thus reduces the cumulative errors. However, there are still differences in accuracy between different sequences of nodes. Next we will describe our method that improves the ADAG by finding a best sequence of nodes.

3 The proposed method

In this section, we introduce the modification of the ADAG to improve the performance of the original ADAG. This approach determines a best sequence of nodes in the ADAG by dynamically reordering the sequence during classification process according to each test data.

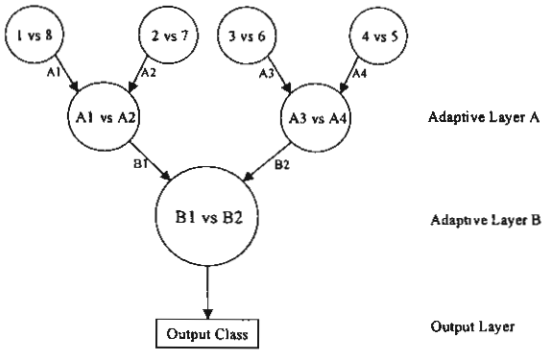


Figure 2: The structure of an Adaptive DAG for an 8-class problem

3.1 Generalization Performance of Support Vector Machines

The ability of a hypothesis to correctly classify data not in the training set is known as its generalization. Generalization analysis of pattern classifiers is concerned with determining the factors that affect the accuracy of a pattern classifier [1]. Generalization performance of Support Vector Machines can be approximated by bounding on the generalization error.

Define the class F of real-valued functions on the ball of radius R in $\Re^n$ as $F = \{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} : \|\mathbf{w}\| \le 1, \|\mathbf{x}\| \le R\}$. There is a constant c such that, for all probability distributions, with probability at least 1-δ over m independently generated examples $\mathbf{z}$, if a classifier $h = \mathrm{sgn}(f) \in \mathrm{sgn}(F)$ has margin at least γ on all the examples in $\mathbf{z}$, then the error of h is no more than

$$\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m + \log\left(\frac{1}{\delta}\right)\right) \quad \text{.....(13)}$$

Furthermore, with probability at least 1-δ, every classifier $h \in \mathrm{sgn}(F)$ has error no more than

$$\frac{k}{m}+\sqrt{\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m+\log\left(\frac{1}{\delta}\right)\right)} \quad \text{.....(14)}$$

where k is the number of labeled examples in $\mathbf{z}$ with margin less than γ.

Below we show an example of the generalization error. The experiment is based on the English letter image recognition dataset from [2], which has 26 classes. Hence there are 325 classifiers. In this case, we construct classifiers by using the Polynomial kernel of degree 3. In Figure 3, the generalization errors of all classifiers expressed by Equation (14) are depicted. The generalization errors of all classifiers are varying.

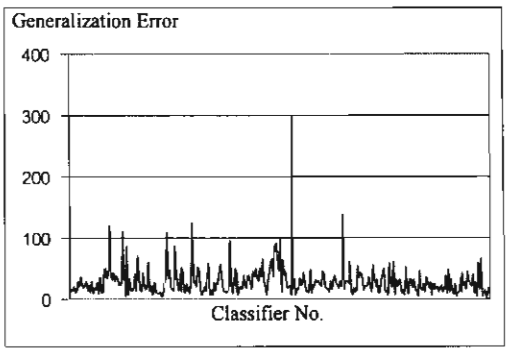


Figure 3: The generalization errors of 325 classifiers

3.2 Reordering ADAG

We propose a method, called Reordering Adaptive Directed Acyclic Graph (RADAG), to improve the accuracy of the original ADAG. For a k-class problem, the RADAG's training phase is the same as the ADAG method by solving $k(k\text{-}1)/2$ binary SVMs. However, the testing phase is organized as follows. The differences are the initialization of the binary classifiers in the top level and the order of sequence in each level (see Figure 4). In the first step, we use a reordering algorithm with minimum-weight perfect matching described in the next subsection to choose the optimal sequence to be the initial sequence. We use the sequence to evaluate every test example. In the second step, as in the ADAG, test points of the RADAG are evaluated against the decision nodes. In the third step, unlike the ADAG, the RADAG will reorder the sequence before processing in the next level by using the reordering algorithm with minimum-weight perfect matching. This sequence differs for each test example, and it depends on the results of nodes from the previous level. The second and the third steps are repeated until there is only one class remains.

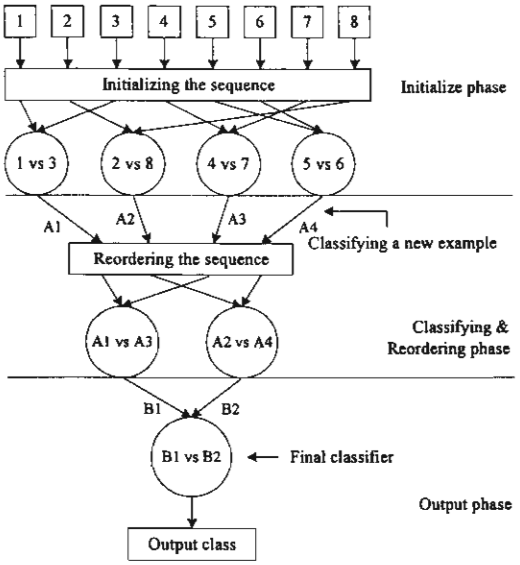


Figure 4: Classifying process of the RADAG

3.3 Reordering algorithm

For the reason described above, we consider the generalization errors in order to choose the optimal sequence from all possible $\frac{k!}{2^{\lfloor \frac{k}{2} \rfloor}}$ sequences with less chance to predict the wrong class. Among classifiers, $k/2$ classifiers which have small generalization errors will be considered to be used in data classification.

Let $G = (V, E)$ be a graph with node set V and edge set E. Each node in G denotes one class and each edge denotes one binary classifier

which has a generalization error expressed by Equation (14) (see Figure 5(a)). The output of the reordering algorithm for graph G is a subset of edges with the minimum sum of generalization errors of all edges and each node in G is met by exactly one edge in the subset (see Figure 5(b)). Given a real weight c_e being generalization error for each edge e of G, the problem of reordering algorithm can be solved by the minimum weight perfect matching that finds a perfect matching M of minimum weight $\Sigma(c_e : e \in M)$.

For $U \subseteq V$, let $E(U) = \{(i,j):(i,j)\in E,\ i\in U,\ j\in U\}$. $E(U)$ is the set of chosen classifiers. Let $\delta(i)$ denote the set of edges incident to node i; the set of classifiers with one class being i. The perfect matchings on a graph $G = (V, E)$ with $|V|$ even is given by

(a) $x \in R_+^m$

(b) $\sum_{e\in\delta(v)} x_e = 1$ for $v \in V$

(c) $\sum_{e\in E(U)} x_e \le \left\lfloor \frac{|U|}{2} \right\rfloor$ for all odd sets $U \subseteq V$ with $|U|\ge 3$

or by (a),(b) and

(d) $\sum_{e\in\delta(U)} x_e \ge 1$ for all odd sets $U \subseteq V$ with $|U|\ge 3$

.....(15)

where $|E| = m$, the number of classifiers, and $x_e = 1$ means that classifier e is chosen to be used in the sequence. Therefore, the reordering problem is to solve the following linear program:

$$\min \sum_{e\in E} c_e x_e \quad \text{.....(16)}$$

where x satisfies "(a),(b) and (c)" or "(a),(b) and (d)".

Currently, the minimum-weight perfect matching algorithm runs in time bounded by $O(n(m + n \log n))$ [3], where n is the number of nodes (classes) in the graph and m is the number of edges (binary classifiers). Hence the reordering algorithm can reorder the sequence in that polynomial time.

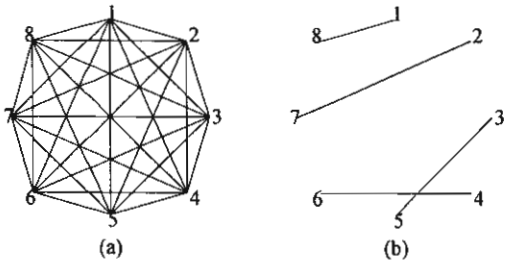


Figure 5: (a) A graph for an 8-class problem. (b) An example of the output of the reordering algorithm.

4 Numerical experiments

In this section, we present experimental results on several datasets from the UCI Repository of machine learning databases [2] including glass, satimage, segment, shuttle, vowel, soybean, letter and isolet (see Table 1). These datasets are different in the number of classes, the number of dimensions, and sizes. For the glass and segment problems, there is no provided test data so we used 5-fold cross validation. For the soybean problem, we discarded the last four classes because of missing values.

Table 1: Description of the datasets used in the experiments

Dataset	#training data	#test data	#class	#dimension
Glass	214	5-fold	6	9
Satimage	4,435	2,000	6	36
Segment	2,310	5-fold	7	18
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35
Letter	15,963	4,037	26	16
Isolet	6,238	1,559	26	617

In these experiments we scaled both training data and test data to be in [-1,1] and employed Polynomial and RBF kernels. In the experiments, we compare three algorithms, i.e., the DDAG, the original ADAG, the RADAG, and the Max Wins algorithm. For the ADAG, we examined all

possible sequences for datasets having not more than 7 classes, whereas we randomly selected 50,000 sequences for datasets having more than 7 classes. Table 2 and 3 present the results of the comparison of these methods for Polynomial and RBF kernels, respectively. We present the optimal parameters (d and c in Equations (11) and (12)) of the kernels and the corresponding accuracies. The best accuracy among three methods is illustrated in bold-face. ***, ** and * in the tables means 99%, 95%, and 90% confidence interval for estimating the difference between accuracies of three algorithms and the RADAG using a one-tailed paired t-test.

The results show that our method yields highest accuracy in almost all of datasets. The results also show that our method performs statistically significantly better than the other methods in the glass problem in case of the RBF kernel and significantly better than the DDAG in the segment and letter problems in case of the Polynomial kernel. Another advantage of our method compared to the DDAG and the original ADAG is that our method always provides one best accuracy for each dataset using the reordering algorithm, whereas, depending on the sequence of classes, the DDAG and the original ADAG may give low accuracies. This shows the effectiveness of the RADAG.

Table 4 and 5 present the comparison of the computational time between the RADAG and the Max Wins for Polynomial and RBF kernels by using a 400 MHz Pentium II processor. There is no computational time of the glass dataset because it has too few test examples to measure the time. The results show that our method requires low computational time in all datasets, especially when the number of classes and/or the number of dimensions are relatively large. For a k class problem, the Max Wins requires $k(k\text{-}1)/2$ classifiers for the classification whereas the RADAG requires only k-1 classifiers. Hence the larger the number of classes the more running time the Max Wins requires than the RADAG. Moreover, the number of dimensions affects the running time of each classifier. For the RADAG, the number of classes affects the running time for reordering. However, it takes little time even when there are many classes.

5 Conclusion

In this paper, we have presented a new approach for multiclass SVMs, called Reordering Adaptive Directed Acyclic Graph (RADAG), which is the modification of the original ADAG. Our approach eliminates the dependency of the sequence of binary classifiers in nodes in the original ADAG by selecting an appropriate sequence from all possible sequences which consists of classifiers with small generalization error. By the use of minimum-weight perfect matching, the RADAG can reorder the sequence in polynomial time. The experimental results show that our new approach yields higher accuracy than the original ADAG and even Max Wins which is probably the currently most accurate method for multiclass SVMs. Moreover the running time used by the RADAG is less than Max Wins, especially when the number of classes and/or the number of dimensions are relatively large. Our future work is to test the method on datasets with a very large number of classes and dimensions.

Table 2: A comparison of the accuracy of classification using the Polynomial kernel

Dataset	d	DDAG	d	ADAG	d	Max Wins	d	RADAG
Glass	2	71.069	2	**71.135**	2	71.078	2	71.063
Satimage	6	88.408***	6	88.430	6	88.453	6	**88.900**
Segment	6	96.538	8	97.408	8	97.379	8	**97.489**
Shuttle	8	**99.924**	8	**99.924**	8	**99.924**	8	**99.924**
Vowel	3	64.237	3	64.293	3	64.329	2	**64.502**
Soybean	5	90.400	5	90.446	3	90.471	3	**91.176**
Letter	3	95.508*	3	95.984	3	**96.125**	4	96.111
Isolet	3	97.032	3	97.030	3	97.040	3	**97.049**

Table 3: A comparison of the accuracy of classification using the RBF kernel

Dataset	c	DDAG	c	ADAG	c	Max Wins	c	RADAG
Glass	0.08	72.850**	0.08	72.759**	0.09	73.238**	0.09	**74.319**
Satimage	3.0	91.971	3.0	91.968	3.0	**91.984**	3.0	91.950
Segment	0.7	97.276	0.7	97.282	0.7	**97.298**	0.7	97.273
Shuttle	3.0	**99.897**	3.0	**99.897**	3.0	**99.897**	3.0	**99.897**
Vowel	0.2	65.425	0.2	65.589	0.2	65.340	0.2	**67.100**
Soybean	0.07	90.353	0.08	90.412	0.08	90.468	0.07	**90.882**
Letter	3.0	97.901	3.0	97.909	3.0	97.918	3.0	**97.969**
Isolet	0.01	96.939	0.01	96.932	0.01	96.916	0.01	**96.985**

Table 4: A comparison of the computational time using the Polynomial kernel

Dataset	d	RADAG			Max Wins
		Classifying (seconds)	Reordering (seconds)	Total (seconds)	Classifying (seconds)
Satimage	6	1.90	0.50	2.40	9.47
Segment	8	0.11	0.08	0.19	0.41
Shuttle	8	1.75	3.38	5.13	5.15
Vowel	2	0.12	0.25	0.37	0.61
Soybean	3	0.30	0.25	0.55	1.86
Letter	4	8.48	4.20	12.68	125.58
Isolet	3	116.02	1.48	117.50	1,671.98

Table 5: A comparison of the computational time using the RBF kernel

Dataset	c	RADAG			Max Wins
		Classifying (seconds)	Reordering (seconds)	Total (seconds)	Classifying (seconds)
Satimage	3.0	11.75	0.51	12.26	37.13
Segment	0.7	0.24	0.10	0.34	0.82
Shuttle	3.0	3.36	0.63	3.99	9.27
Vowel	0.2	0.10	0.27	0.37	0.61
Soybean	0.07	0.32	0.45	0.77	2.20
Letter	3.0	62.27	3.69	65.96	802.85
Isolet	0.01	100.42	1.60	102.02	1,369.11

References

[1] P. L. Bartlett, J. Shawe-Taylor, "Generalization performance of support vector machines and other pattern classifiers", in Advances in Kernel Methods - Support Vector Learning, B. Schoelkopf, C. J. C. Burges, A. J. Smola (eds), pp. 43-54, MIT Press, Cambridge, USA, 1999.

[2] C. Blake, E. Keogh and C. Merz, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html

[3] W. Cook, and A. Rohe, "Computing minimum-weight perfect matchings", Technical Report 97863, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 1997.

[4] J. H. Friedman, "Another approach to Polychotomous classification", Technical report, Department of Statistics, Stanford University, 1996.

[5] C.-W. Hsu, and C.-J. Lin, "A comparison of methods for multiclass Support Vector Machines", IEEE Trans.on Neural Networks, Vol. 13, pp. 415-425, March, 2002.

[6] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification", in Advances in Neural Information Processing Systems, MIT Press, Vol. 12, pp. 547-553, 2000.

[7] N. Ussivakul, and B. Kijsirikul, "Multiclass support vector machines using adaptive directed acyclic graph," in IEEE/INNS' Int. Joint Conf. On Neural Networks (IJCNN-2002), 2002.

[8] V. Vapnik, Statistical Learning Theory, New York, Wiley, 1998.

Learning Multiclass Support Vector Machines by Reordering Adaptive Directed Acyclic Graph

Thimaporn Phetkaew[1], Wanchai Rivepiboon[2] and Boonserm Kijsirikul[3]
Department of Computer Engineering
Chulalongkorn University
Phayathai, Patumwan, Bangkok, 10330, Thailand
E-mail: Thimaporn.P@student.chula.ac.th[1] and Wanchai.R[2], Boonserm.K[3]@chula.ac.th

Abstract

The problem of extending binary support vector machines (SVMs) for multiclass classification is still an ongoing research issue. Ussivakul and Kijsirikul proposed the Adaptive Directed Acyclic Graph (ADAG) approach that provides accuracy comparable to that of the standard algorithm—Max Wins and requires low computation. However, different sequences of nodes in the ADAG may provide different accuracy. In this paper we present a new method for multiclass classification, Reordering Adaptive Directed Acyclic Graph (RADAG), which is the modification of the original ADAG method. We propose an algorithm to choose an optimal sequence of binary classifiers in nodes in the ADAG by considering the generalization error bounds of all classifiers. We apply minimum-weight perfect matching with the reordering algorithm in order to select the best sequence of nodes in polynomial time. We then compare the performance of our method with previous methods including the ADAG and the Max Wins algorithm. Experiments denote that our method gives higher accuracy. Moreover it runs faster than Max Wins, especially when the number of classes and/or the number of dimensions are relatively large.

Key words: Multiclass Support Vector Machines, ADAG

Introduction

Support vector machines (SVMs) were primarily designed for two-class classification problems with its outstanding performance in real world applications. However, extending SVMs for multiclass classification is still an ongoing research issue. Previous methods for solving the multiclass problem of SVMs are typically to consider the problem as the combination of two-class decision functions, e.g. one-against-one and one-against-the-rest [5].

Friedman [4] suggested the Max Wins algorithm in which each one-against-one classifier casts one vote for its preferred class, and the final result is the class with the most votes. The Max Wins algorithm offers faster training time compared to the one-against-the-rest method. The decision Directed Acyclic Graph (DDAG) method proposed by Platt et al. reduces training and evaluation time, while maintaining accuracy compared to the Max Wins [6]. The comparison experiments in several methods on large problems in [5] show that the Max Wins algorithm and the DDAG may be more suitable for practical use. Ussivakul and Kijsirikul [7] proposed the Adaptive Directed Acyclic Graph (ADAG) method which is the modification of the DDAG. This method reduces the dependency of the sequence of binary classifiers in nodes in the structure as well as lowers the number of tests required to evaluate for the correct class. Their approach yields higher accuracy and reliability of classification, especially in such a case that the number of classes is relatively large.

In this paper we reveal that the ADAG still has the dependency on the sequence of its nodes, although it is less dependent on the order of binary classes in the sequence than the DDAG; there are still differences in accuracy between different sequences. This led to the reliability of the algorithm. Here we propose a novel method that improves reliability by choosing an optimal sequence, which has less chance to predict the wrong class, and dynamically reordering the sequence during classification process according to each test data. We also reveal that the problem of selecting the appropriate sequence can be solved by minimum-weight perfect matching.

This paper is organized as follows. In the next section, we review SVMs and the formulation to solve multiclass problems, i.e., the DDAG and the ADAG. In Section 3, we introduce the modification of the ADAG to improve the performance by using the reordering algorithm with minimum-weigh perfect matching. The numerical experiments are illustrated in Section 4. Finally, the conclusions are given in Section 5.

2. SVM classification

This section describes the basic idea of SVMs [7] and the formulation to solve multiclass problems.

2.1. Support vector machines

The main idea of support vector machine classification is to construct a hyperplane to separate the two classes.

2.1.1. Linear support vector machines

Suppose we have a data set D of l samples in an n-dimensional space belonging to two different classes (+1 and −1):

$$D = \{(\mathbf{x}_k, y_k) \mid k \in \{1,..,l\}, \mathbf{x}_k \in \Re^n, y_k \in \{+1,-1\}\} \quad (1)$$

The hyperplane in the n dimensional space is determined by the pair $(\mathbf{w},b)$ where $\mathbf{w}$ is an n-dimensional vector orthogonal to the hyperplane and b is the offset constant. The hyperplane $(\mathbf{w}\cdot\mathbf{x})+b$ separates the data if and only if

$$\begin{aligned} (\mathbf{w}\cdot\mathbf{x}_i)+b > 0 \quad &\text{if} \quad y_i = +1 \\ (\mathbf{w}\cdot\mathbf{x}_i)+b < 0 \quad &\text{if} \quad y_i = -1. \end{aligned} \quad (2)$$

If we additionally require that $\mathbf{w}$ and b be such that the point closest to the hyperplane has a distance of $1/|\mathbf{w}|$, then we have

$$\begin{aligned} (\mathbf{w}\cdot\mathbf{x}_i)+b \geq 1 \quad &\text{if} \quad y_i = +1 \\ (\mathbf{w}\cdot\mathbf{x}_i)+b \leq -1 \quad &\text{if} \quad y_i = -1 \end{aligned} \quad (3)$$

which is equivalent to

$$y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1 \quad \forall i. \quad (4)$$

To find the optimal separating hyperplane, we have to find the hyperplane that maximizes the minimum distance between the hyperplane and any sample of training data. The distance between two closest samples from different classes is

$$d(\mathbf{w},b) = \min_{\{\mathbf{x}_i | y_i=1\}} \frac{(\mathbf{w}\cdot\mathbf{x}_i)+b}{|\mathbf{w}|} - \max_{\{\mathbf{x}_i | y_i=-1\}} \frac{(\mathbf{w}\cdot\mathbf{x}_i)+b}{|\mathbf{w}|}. \quad (5)$$

From (3), we can see that the appropriate minimum and maximum values are ±1. Therefore, we need to maximize

$$d(\mathbf{w},b) = \frac{1}{|\mathbf{w}|} - \frac{-1}{|\mathbf{w}|} = \frac{2}{|\mathbf{w}|}. \quad (6)$$

Thus, the problem is equivalent to:

- minimize $|\mathbf{w}|^2/2$
- subject to the constraints
  (1) $y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1 \quad \forall i.$

For non-separable case, the training data cannot be separated by a hyperplane without error. The previous constraints then must be modified. A penalty term consisting of the sum of deviations ξ_i from the boundary is added to the minimization problem. Now, the problem is to

- minimize $\frac{|\mathbf{w}|^2}{2} + C\sum_{i=1}^{l}\xi_i$
- subject to the constraints
  (1) $y_i[(\mathbf{w}\cdot\mathbf{x}_i)+b] \geq 1-\xi_i,$
  (2) $\xi_i \geq 0 \quad \forall i.$

The penalty term for misclassifying training samples is weighted by a constant C. Selecting a large value of C puts a high price on deviations and increases computation by effecting a more exhaustive search for ways to minimize the number of misclassified samples.

By forming the Lagrangian and solving the dual problem, this problem can be translated into:

- minimize
$$L(\mathbf{w},b,\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j (\mathbf{x}_i\cdot\mathbf{x}_j) \quad (7)$$
- subject to the constraints:
  (1) $0 \leq \alpha_i \leq C, \forall i$
  (2) $\sum_{i=1}^{l}\alpha_i y_i = 0$

where α_i are called Lagrange multipliers. There is one Lagrange multiplier for each training sample. In the solution, those samples for which $\alpha_i > 0$ are called *support vectors*, and are ones such that the equality in (4) holds. All other training samples having $\alpha_i = 0$ could be removed from the training set without affecting the final hyperplane.

Let α^0, an l-dimensional vector denote the minimum of $L(\mathbf{w},b,\alpha)$. If $\alpha_i^0 > 0$ then $\mathbf{x}_i$ is a *support vector*. The optimal separating hyperplane $(\mathbf{w}^0, b^0)$ can be written in terms of α^0 and the training data, specifically in terms of the support vectors:

$$\mathbf{w}^0 = \sum_{i=1}^{l}\alpha_i^0 y_i \mathbf{x}_i = \sum_{\text{support vectors}}\alpha_i^0 y_i \mathbf{x}_i. \quad (8)$$

$$b^0 = 1 - \mathbf{w}^0\cdot\mathbf{x}_i \text{ for } \mathbf{x}_i \text{ with } y_i = 1 \text{ and } 0 < \alpha_i < C. \quad (9)$$

The optimal separating hyperplane classifies points according to the sign of $f(\mathbf{x})$,

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}\left(\mathbf{w}^0\cdot\mathbf{x} + b^0\right) \\ &= \text{sign}\left(\sum_{\text{suport vectors}}\alpha_i^0 y_i(\mathbf{x}_i\cdot\mathbf{x}) + b^0\right). \end{aligned} \quad (10)$$

Support vector $\mathbf{x}_i$ with $\alpha_i^0 = C$ may or may not be misclassified. All other $\mathbf{x}_i$'s are correctly classified.

2.1.2. Non-linear support vector machines

The above algorithm is limited to linear separating hyperplanes. SVMs get around this problem by mapping the sample points into a higher dimensional space using a non-linear mapping chosen in advance. This is, we choose a map $\Phi : \Re^n \mapsto H$ where the dimensionality of H is greater than n. We then seek a separating hyperplane in the higher dimensional space; this is equivalent to a non-linear separating surface in $\Re^n$.

The data only ever appears in our training problem (7) in the form of dot products, so in the higher dimensional space we are only dealing with the data in the form $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$. If the dimensionality of H is very large, then this could be difficult, or very computationally expensive. However, if we have a kernel function such that $k(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$, then we can use this in place of $\mathbf{x}_i\cdot\mathbf{x}_j$ everywhere in the optimization problem, and never need to know explicitly what Φ is. Some widely used kernels are:

$$\text{Polynomial degree } d\text{: } k(\mathbf{x},\mathbf{y}) = \left|\mathbf{x}\cdot\mathbf{y}+1\right|^d \qquad (11)$$

$$\text{Radial basis function: } k(\mathbf{x},\mathbf{y}) = e^{-\left|\mathbf{x}-\mathbf{y}\right|^2/c} \qquad (12)$$

2.2. DDAG

Platt et al. [6] presented a learning architecture, the Decision Directed Acyclic Graph (DDAG), which is used to combine many two-class classifiers into a multiclass classifier. For a k-class problem, its training phase is the same as the one-against-one method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, it uses a rooted binary directed acyclic graph which has $k(k\text{-}1)/2$ internal nodes and k leaves (see Figure 1). Each node is a binary SVM of the i^{th} and j^{th} classes. Given a test sample x, starting at the root node, the binary decision function is evaluated. Then it moves to either left or right depending on the output value. Therefore, we go through a path before reaching a leaf node which indicates the predicted class.

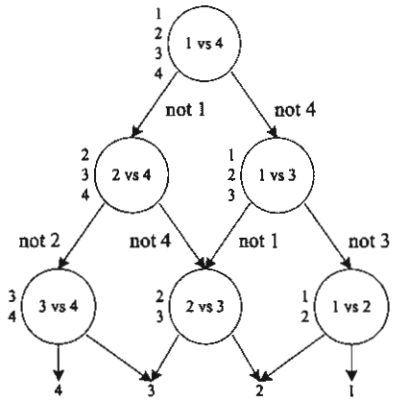


Figure 1. The DDAG finding the best class out of four classes

There are some issues on the DDAG as pointed out by [7]. First, it gives outputs whose probabilities are not uniformly distributed, and thus its output depends on the sequence of binary classifiers in nodes, affecting reliability of the algorithm. In addition, the correct class placed in a node near the root node is clearly at disadvantage by comparison with the correct class near leaf nodes since it is exposed to higher risk of being incorrectly rejected. Second, the number of node evaluations for the correct class is unnecessary high. This results in higher cumulative error and lower the accuracy. The depth of the DDAG is k-1 and this means that the number of times the correct class has to be tested against other classes, on average, scales linearly with k.

2.3. ADAG

Ussivakul and Kijsirikul [7] proposed an approach to alleviate the problem of the DDAG structure described above. An Adaptive DAG (ADAG) is a DAG with a reversed triangular structure. For a k-class problem, its training phase is the same as the DDAG method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, the nodes are arranged in a reversed triangle with $k/2$ nodes (rounded up) at the top, $k/2^2$ nodes in the second layer and so on until the lowest layer of a final node. It has k-1 internal nodes, each of which is labeled with an element of Boolean function (see Figure 2). Given a test example x, starting at the top level, the binary decision function is evaluated. The node is then exited via the outgoing edge with a message of the preferred class. In each round, the number of candidate classes is reduced by half. Based on the preferred classes from its parent nodes, the binary function of the next-level node is chosen. The reduction process continues until reaching the final node at the lowest level. The value of the decision function is the value associated with the message from the final leaf node. Like the DDAG, the ADAG requires only k-1 decision nodes to be evaluated in order to derive an answer. Note that the correct class is evaluated against other classes for $\log_2 k$ times or less, considerably lower than the number of evaluations required by the DDAG, which scales linearly with k.

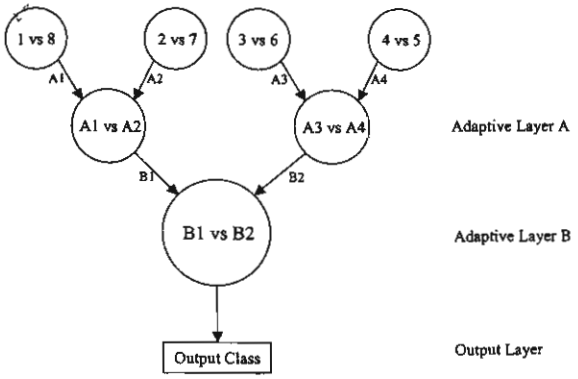


Figure 2. The structure of an Adaptive DAG for an 8-class problem

2.1.2. Non-linear support vector machines

The above algorithm is limited to linear separating hyperplanes. SVMs get around this problem by mapping the sample points into a higher dimensional space using a non-linear mapping chosen in advance. This is, we choose a map $\Phi : \Re^n \mapsto H$ where the dimensionality of H is greater than n. We then seek a separating hyperplane in the higher dimensional space; this is equivalent to a non-linear separating surface in $\Re^n$.

The data only ever appears in our training problem (7) in the form of dot products, so in the higher dimensional space we are only dealing with the data in the form $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$. If the dimensionality of H is very large, then this could be difficult, or very computationally expensive. However, if we have a kernel function such that $k(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$, then we can use this in place of $\mathbf{x}_i\cdot\mathbf{x}_j$ everywhere in the optimization problem, and never need to know explicitly what Φ is. Some widely used kernels are:

$$\text{Polynomial degree } d\text{: } k(\mathbf{x},\mathbf{y}) = \left|\mathbf{x}\cdot\mathbf{y}+1\right|^d \qquad (11)$$

$$\text{Radial basis function: } k(\mathbf{x},\mathbf{y}) = e^{-\left|\mathbf{x}-\mathbf{y}\right|^2/c} \qquad (12)$$

2.2. DDAG

Platt et al. [6] presented a learning architecture, the Decision Directed Acyclic Graph (DDAG), which is used to combine many two-class classifiers into a multiclass classifier. For a k-class problem, its training phase is the same as the one-against-one method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, it uses a rooted binary directed acyclic graph which has $k(k\text{-}1)/2$ internal nodes and k leaves (see Figure 1). Each node is a binary SVM of the i^{th} and j^{th} classes. Given a test sample x, starting at the root node, the binary decision function is evaluated. Then it moves to either left or right depending on the output value. Therefore, we go through a path before reaching a leaf node which indicates the predicted class.

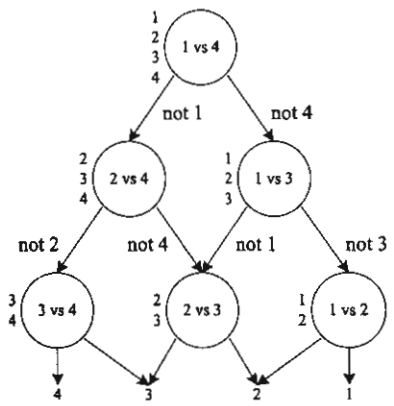


Figure 1. The DDAG finding the best class out of four classes

There are some issues on the DDAG as pointed out by [7]. First, it gives outputs whose probabilities are not uniformly distributed, and thus its output depends on the sequence of binary classifiers in nodes, affecting reliability of the algorithm. In addition, the correct class placed in a node near the root node is clearly at disadvantage by comparison with the correct class near leaf nodes since it is exposed to higher risk of being incorrectly rejected. Second, the number of node evaluations for the correct class is unnecessary high. This results in higher cumulative error and lower the accuracy. The depth of the DDAG is k-1 and this means that the number of times the correct class has to be tested against other classes, on average, scales linearly with k.

2.3. ADAG

Ussivakul and Kijsirikul [7] proposed an approach to alleviate the problem of the DDAG structure described above. An Adaptive DAG (ADAG) is a DAG with a reversed triangular structure. For a k-class problem, its training phase is the same as the DDAG method by solving $k(k\text{-}1)/2$ binary SVMs, one for each pair of classes. However, in the testing phase, the nodes are arranged in a reversed triangle with $k/2$ nodes (rounded up) at the top, $k/2^2$ nodes in the second layer and so on until the lowest layer of a final node. It has k-1 internal nodes, each of which is labeled with an element of Boolean function (see Figure 2). Given a test example x, starting at the top level, the binary decision function is evaluated. The node is then exited via the outgoing edge with a message of the preferred class. In each round, the number of candidate classes is reduced by half. Based on the preferred classes from its parent nodes, the binary function of the next-level node is chosen. The reduction process continues until reaching the final node at the lowest level. The value of the decision function is the value associated with the message from the final leaf node. Like the DDAG, the ADAG requires only k-1 decision nodes to be evaluated in order to derive an answer. Note that the correct class is evaluated against other classes for $\log_2 k$ times or less, considerably lower than the number of evaluations required by the DDAG, which scales linearly with k.

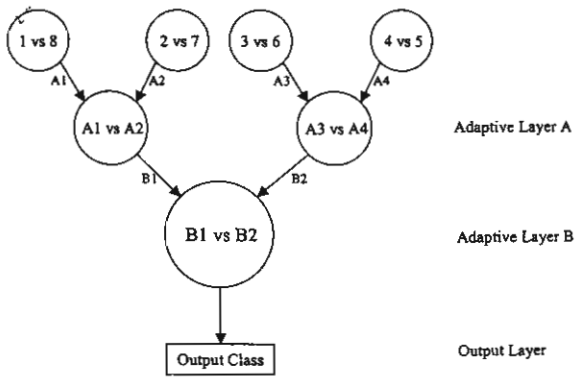


Figure 2. The structure of an Adaptive DAG for an 8-class problem

Using the reversed triangular structure, the ADAG reduces the number of times the correct class is tested against other classes, and thus reduces the cumulative errors. However, there are still differences in accuracy between different sequences of nodes. Next we will describe our method that improves the ADAG by finding a best sequence of nodes.

3. The proposed method

In this section, we introduce the modification of the ADAG to improve the performance of the original ADAG. This approach determines a best sequence of nodes in the ADAG by dynamically reordering the sequence during classification process according to each test data.

3.1. Generalization Performance of Support Vector Machines

Generalization analysis of pattern classifiers is concerned with determining the factors that affect the accuracy of a pattern classifier [1]. Generalization performance of Support Vector Machines can be approximated by bounding on the generalization error.

Define the class F of real-valued functions on the ball of radius R in $\Re^n$ as $F = \{\mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} : \|\mathbf{w}\| \le 1, \|\mathbf{x}\| \le R\}$. There is a constant c such that, for all probability distributions, with probability at least $1-\delta$ over m independently generated examples $\mathbf{z}$, if a classifier $h = \mathrm{sgn}(f) \in \mathrm{sgn}(F)$ has margin at least γ on all the examples in $\mathbf{z}$, then the error of h is no more than

$$\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m + \log\left(\frac{1}{\delta}\right)\right). \qquad (13)$$

Furthermore, with probability at least $1-\delta$, every classifier $h \in \mathrm{sgn}(F)$ has error no more than

$$\frac{k}{m} + \sqrt{\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m + \log\left(\frac{1}{\delta}\right)\right)} \qquad (14)$$

where k is the number of labeled examples in $\mathbf{z}$ with margin less than γ.

Below we show an example of the generalization error of classifier. The experiment is based on the English letter image recognition dataset from [2], which has 26 classes. Hence there are 325 classifiers. In this case, the dataset is trained by using the Polynomial kernel of degree 3. In Figure 3, the generalization errors of all classifiers expressed by Equation (14) are depicted. The generalization errors of all classifiers are varying.

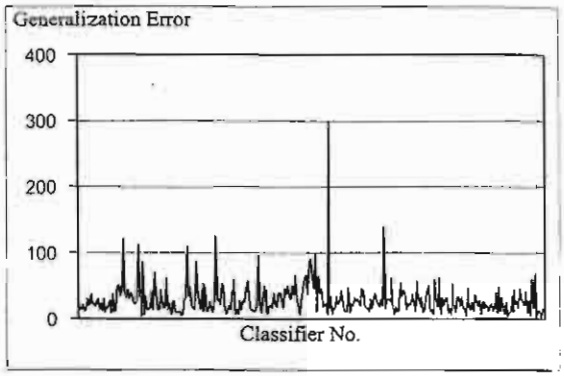


Figure 3. The generalization errors of 325 classifiers

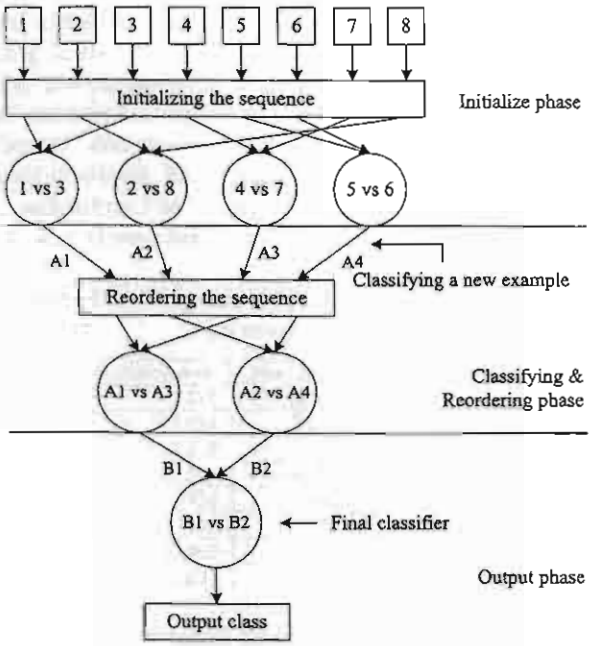


Figure 4. Classifying process of the RADAG

3.2. Reordering ADAG

We propose a method, called Reordering Adaptive Directed Acyclic Graph (RADAG), to improve the accuracy of the original ADAG. For a k-class problem, the RADAG's training phase is the same as the ADAG method by solving $k(k-1)/2$ binary SVMs. However, the testing phase is organized as follows. The differences are the initialization of the binary classifiers in the top level and the order of sequence in each level (see Figure 4). In the first step, we use a reordering algorithm with minimum-weight perfect matching described in the next subsection to choose the optimal sequence to be the initial sequence. We use the sequence to evaluate every test example. In the second step, as in the ADAG, test points of the RADAG are evaluated against the decision nodes. In the third step, unlike the ADAG, the RADAG will reorder the sequence before processing in the next level by using the reordering algorithm with minimum-weight

perfect matching. This sequence differs for each test example, and it depends on the results of nodes from the previous level. The second and the third steps are repeated until there is only one class remains.

3.3. Reordering algorithm

For the reason described above, we consider the generalization errors in order to choose the optimal sequence from all possible sequences with less chance to predict the wrong class. Among classifiers, $k/2$ classifiers which have small generalization errors will be considered to be used in data classification.

Let $G = (V, E)$ be a graph with node set V and edge set E. Each node in G denotes one class and each edge denotes one binary classifier which has a generalization error expressed by Equation (14) (see Figure 5(a)). The output of the reordering algorithm for graph G is a subset of edges with the minimum sum of generalization errors of all edges and each node in G is met by exactly one edge in the subset (see Figure 5(b)). Given a real weight c_e being generalization error for each edge e of G, the problem of reordering algorithm can be solved by the minimum weight perfect matching that finds a perfect matching M of minimum weight $\Sigma(c_e : e \in M)$.

For $U \subseteq V$, let $E(U) = \{(i,j):(i,j) \in E,\ i \in U, j \in U\}$. $E(U)$ is the set of chosen classifiers. Let $\delta(i)$ denote the set of edges incident to node i; the set of classifiers with one class being i. The perfect matchings on a graph $G = (V, E)$ with $|V|$ even is given by

(a) $x \in R_+^m$

(b) $\sum_{e \in \delta(v)} x_e = 1 \quad \text{for } v \in V$

(c) $\sum_{e \in E(U)} x_e \le \left\lfloor \frac{|U|}{2} \right\rfloor$ for all odd sets $U \subseteq V$ with $|U| \ge 3$

or by (a),(b) and

(d) $\sum_{e \in \delta(U)} x_e \ge 1$ for all odd sets $U \subseteq V$ with $|U| \ge 3$ (16)

where $|E| = m$, the number of classifiers, and $x_e = 1$ means that classifier e is chosen to be used in the sequence. Therefore, the reordering problem is to solve the following linear program:

$$\min \sum_{e \in E} c_e x_e \qquad (17)$$

where x satisfies "(a),(b) and (c)" or "(a),(b) and (d)".

Currently, the minimum-weight perfect matching algorithm runs in time bounded by $O(n(m + n \log n))$ [3], where n is the number of nodes (classes) in the graph and m is the number of edges (binary classifiers). Hence the reordering algorithm can reorder the sequence in that polynomial time.

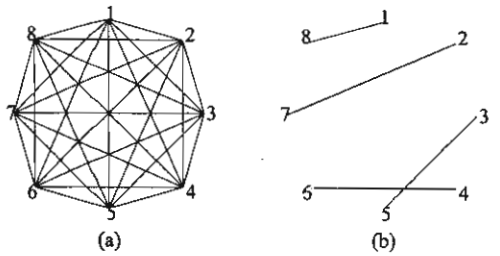


Figure 5. (a) A graph for an 8-class problem. (b) An example of the output of the reordering algorithm.

4. Numerical experiments

In this section, we present experimental results on several datasets from the UCI Repository of machine learning databases [2] including glass, satimage, segment, shuttle, vowel, soybean, letter and isolet (see Table 1). These datasets are different in the number of classes, the number of dimensions, and sizes. For the glass and segment problems, there is no test data so we used 5-fold cross validation. For the soybean problem, we discarded the last four classes because of missing values.

Table 1. Description of the datasets used in the experiments

Dataset	#training data	#test data	#class	#dimension
Glass	214	5-fold	6	9
Satimage	4,435	2,000	6	36
Segment	2,310	5-fold	7	18
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35
Letter	15,963	4,037	26	16
Isolet	6,238	1,559	26	617

In these experiments we scaled both training data and test data to be in [-1,1] and employed Polynomial and RBF kernels. In the experiments, we compare three algorithms, i.e., the original ADAG, the RADAG, and the Max Wins algorithm. For the ADAG, we examined all possible sequences for datasets having not more than 7 classes, whereas we randomly selected 50,000 sequences for datasets having more than 7 classes. Table 2 and 3 present the results of comparing these methods for Polynomial and RBF kernels, respectively. We present the optimal parameters (d and c in Equations (11) and (12)) of the kernels and the corresponding accuracies. The best accuracy among three methods is illustrated in bold-face.

The results show that our method yields highest accuracy in almost all of datasets. Another advantage of our method compared to the DDAG and the original ADAG is that our method always provides one best accuracy for each dataset using the reordering algorithm, whereas, depending on the sequence of classes, the DDAG and the original ADAG may give low accuracies. This shows the effectiveness of the RADAG.

perfect matching. This sequence differs for each test example, and it depends on the results of nodes from the previous level. The second and the third steps are repeated until there is only one class remains.

3.3. Reordering algorithm

For the reason described above, we consider the generalization errors in order to choose the optimal sequence from all possible sequences with less chance to predict the wrong class. Among classifiers, $k/2$ classifiers which have small generalization errors will be considered to be used in data classification.

Let $G = (V, E)$ be a graph with node set V and edge set E. Each node in G denotes one class and each edge denotes one binary classifier which has a generalization error expressed by Equation (14) (see Figure 5(a)). The output of the reordering algorithm for graph G is a subset of edges with the minimum sum of generalization errors of all edges and each node in G is met by exactly one edge in the subset (see Figure 5(b)). Given a real weight c_e being generalization error for each edge e of G, the problem of reordering algorithm can be solved by the minimum weight perfect matching that finds a perfect matching M of minimum weight $\Sigma(c_e : e \in M)$.

For $U \subseteq V$, let $E(U) = \{(i,j):(i,j)\in E, i\in U, j\in U\}$. $E(U)$ is the set of chosen classifiers. Let $\delta(i)$ denote the set of edges incident to node i; the set of classifiers with one class being i. The perfect matchings on a graph $G = (V, E)$ with $|V|$ even is given by

(a) $x \in R_+^m$

(b) $\sum_{e\in\delta(v)} x_e = 1 \quad \text{for } v \in V$

(c) $\sum_{e\in E(U)} x_e \le \left\lfloor \frac{|U|}{2} \right\rfloor$ for all odd sets $U \subseteq V$ with $|U| \ge 3$

or by (a),(b) and

(d) $\sum_{e\in\delta(U)} x_e \ge 1$ for all odd sets $U \subseteq V$ with $|U| \ge 3$ (16)

where $|E| = m$, the number of classifiers, and $x_e = 1$ means that classifier e is chosen to be used in the sequence. Therefore, the reordering problem is to solve the following linear program:

$$\min \sum_{e\in E} c_e x_e \qquad (17)$$

where x satisfies "(a),(b) and (c)" or "(a),(b) and (d)".

Currently, the minimum-weight perfect matching algorithm runs in time bounded by $O(n(m + n \log n))$ [3], where n is the number of nodes (classes) in the graph and m is the number of edges (binary classifiers). Hence the reordering algorithm can reorder the sequence in that polynomial time.

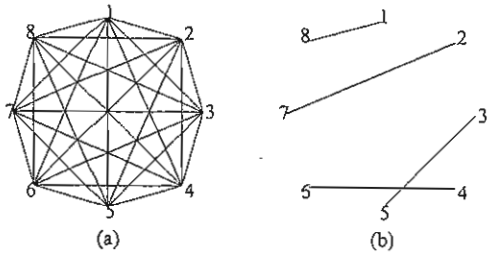


Figure 5. (a) A graph for an 8-class problem. (b) An example of the output of the reordering algorithm.

4. Numerical experiments

In this section, we present experimental results on several datasets from the UCI Repository of machine learning databases [2] including glass, satimage, segment, shuttle, vowel, soybean, letter and isolet (see Table 1). These datasets are different in the number of classes, the number of dimensions, and sizes. For the glass and segment problems, there is no test data so we used 5-fold cross validation. For the soybean problem, we discarded the last four classes because of missing values.

Table 1. Description of the datasets used in the experiments

Dataset	#training data	#test data	#class	#dimension
Glass	214	5-fold	6	9
Satimage	4,435	2,000	6	36
Segment	2,310	5-fold	7	18
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35
Letter	15,963	4,037	26	16
Isolet	6,238	1,559	26	617

In these experiments we scaled both training data and test data to be in [-1,1] and employed Polynomial and RBF kernels. In the experiments, we compare three algorithms, i.e., the original ADAG, the RADAG, and the Max Wins algorithm. For the ADAG, we examined all possible sequences for datasets having not more than 7 classes, whereas we randomly selected 50,000 sequences for datasets having more than 7 classes. Table 2 and 3 present the results of comparing these methods for Polynomial and RBF kernels, respectively. We present the optimal parameters (d and c in Equations (11) and (12)) of the kernels and the corresponding accuracies. The best accuracy among three methods is illustrated in bold-face.

The results show that our method yields highest accuracy in almost all of datasets. Another advantage of our method compared to the DDAG and the original ADAG is that our method always provides one best accuracy for each dataset using the reordering algorithm, whereas, depending on the sequence of classes, the DDAG and the original ADAG may give low accuracies. This shows the effectiveness of the RADAG.

Table 2. A comparison of the accuracy of classification using the Polynomial kernel

Dataset	d	ADAG	d	Max Wins	d	RADAG
Glass	2	**71.135**	2	71.078	2	71.063
Satimage	6	88.430	6	88.453	6	**88.900**
Segment	8	97.408	8	97.379	8	**97.489**
Shuttle	8	**99.924**	8	**99.924**	8	**99.924**
Vowel	3	64.293	3	64.329	2	**64.502**
Soybean	5	90.446	3	90.471	3	**91.176**
Letter	3	95.984	3	**96.125**	4	96.111
Isolet	3	97.030	3	97.040	3	**97.049**

Table 3. A comparison of the accuracy of classification using the RBF kernel

Dataset	c	ADAG	c	Max Wins	c	RADAG
Glass	0.08	72.759	0.09	73.238	0.09	**74.319**
Satimage	3.0	91.968	3.0	**91.984**	3.0	91.950
Segment	0.7	97.282	0.7	**97.298**	0.7	97.273
Shuttle	3.0	**99.897**	3.0	**99.897**	3.0	**99.897**
Vowel	0.2	65.589	0.2	65.340	0.2	**67.100**
Soybean	0.08	90.412	0.08	90.468	0.07	**90.882**
Letter	3.0	97.909	3.0	97.918	3.0	**97.969**
Isolet	0.01	96.932	0.01	96.916	0.01	**96.985**

Table 4. A comparison of the computational time using the Polynomial kernel

Dataset	d	RADAG			Max Wins
		Classifying (seconds)	Reordering (seconds)	Total (seconds)	Classifying (seconds)
Satimage	6	1.90	0.50	2.40	9.47
Segment	8	0.11	0.08	0.19	0.41
Shuttle	8	1.75	3.38	5.13	5.15
Vowel	2	0.12	0.25	0.37	0.61
Soybean	3	0.30	0.25	0.55	1.86
Letter	4	8.48	4.20	12.68	125.58
Isolet	3	116.02	1.48	117.50	1671.98

Table 5. A comparison of the computational time using the RBF kernel

Dataset	c	RADAG			Max Wins
		Classifying (seconds)	Reordering (seconds)	Total (seconds)	Classifying (seconds)
Satimage	3.0	11.75	0.51	12.26	37.13
Segment	0.7	0.24	0.10	0.34	0.82
Shuttle	3.0	3.36	0.63	3.99	9.27
Vowel	0.2	0.10	0.27	0.37	0.61
Soybean	0.07	0.32	0.45	0.77	2.20
Letter	3.0	62.27	3.69	65.96	802.85
Isolet	0.01	100.42	1.60	102.02	1369.11

Table 4 and 5 present the comparison of the computational time between the RADAG and the Max Wins for Polynomial and RBF kernels by using a 400 MHz Pentium II processor. There is no computational time of the glass dataset because it has too little test examples to measure the time. The results show that our method requires low computational time in all datasets, especially when the number of classes and/or the number of dimensions are relatively large. For a k class problem, the Max Wins requires $k(k\text{-}1)/2$ classifiers for the classification whereas the RADAG requires only k-1 classifiers. Hence the larger the number of classes the more running time the Max Wins requires than the RADAG. Moreover, the number of dimensions affects the running time of each classifier. For the RADAG, the number of classes affects the running time for reordering. However, it takes a little time even when there are many classes.

5. Conclusions

In this paper, we have presented a new approach for multiclass SVMs, called Reordering Adaptive Directed Acyclic Graph (RADAG), which is the modification of the original ADAG. Our approach eliminates the dependency of the sequence of binary classifiers in nodes in the original ADAG by selecting an appropriate sequence from all possible sequences which consists of classifiers with small generalization error. By the use of minimum-weight perfect matching, the RADAG can reorder the sequence in polynomial time. The experimental results show that our new approach yields higher accuracy than the original ADAG and even Max Wins which is probably the currently most accurate method for multiclass SVMs. Moreover the running time used by the RADAG is less than Max Wins, especially when the number of classes and/or the number of dimensions are relatively large. Our future work is to test the method on datasets with a very large number of classes and dimensions.

6. Acknowledgment

This work was supported by The Thailand Research Fund.

7. References

[1] P. L. Bartlett, J. Shawe-Taylor, "Generalization performance of support vector machines and other pattern classifiers", in Advances in Kernel Methods - Support Vector Learning, B. Schoelkopf, C. J. C. Burges, A. J. Smola (eds), pp. 43-54, MIT Press, Cambridge, USA, 1999.

[2] C. Blake, E. Keogh and C. Merz, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, 1998. http://www.ics.uci.edu/~mlearn/MLRepository. html

[3] W. Cook, and A. Rohe, "Computing minimum-weight perfect matchings", Technical Report 97863, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 1997.

[4] J. H. Friedman, "Another approach to Polychotomous classification", Technical report, Department of Statistics, Stanford University, 1996.

[5] C.-W. Hsu, and C.-J. Lin, "A comparison of methods for multiclass Support Vector Machines", IEEE Trans.on Neural Networks, Vol. 13, pp. 415-425, March, 2002.

[6] J. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin DAGs for multiclass classification", in Advances in Neural Information Processing Systems, MIT Press, Vol. 12, pp. 547-553, 2000.

[7] N. Ussivakul, and B. Kijsirikul, "Multiclass support vector machines using adaptive directed acyclic graph," in IEEE/INNS' Int. Joint Conf. On Neural Networks (IJCNN-2002), 2002.

A NEW FRAMEWORK FOR LEARNING FIRST-ORDER REPRESENTATION

Thanupol Leardlumnouchai[1] and Boonserm Kijsirikul[1]

ABSTRACT: First-order logic rules are one of the most expressive and human readable representations for learning a hypothesis in form of a set of production rules (if-then rules). However, the first-order rules can be learned only by Inductive Logic Programming (ILP) systems, such as PROGOL, FOIL. Other machine learning techniques, such as Neural Networks, Bayesian Networks and Decision Tree Learning, cannot directly learn this kind of rules because these techniques could not select the appropriate values to substitute in variables of the first-order rules. In this paper, we propose the method that makes use of Multiple-Instance Learning (MIL) as a new framework for learning first-order representation. MIL is employed for determining the appropriate values for the substitution. Experimental results show that the proposed method effectively learns first-order representation and is comparable to ILP systems.

KEYWORDS: Inductive Logic Programming, First-Order Logic, Multiple-Instance Learning

1. INTRODUCTION

Knowledge based learning is one technique of machine learning (Mitchell 1997). Informally knowledge is a set of sentences that describe known logical facts. In learning, the knowledge is generally called background knowledge. Knowledge-based learning method receives background knowledge and positive and negative examples as inputs and outputs a learned hypothesis represented in form of a set of production rules (if-then rules). A prominent advantage of this representation is human readable. Also, there are two types of logic for representing learned rules. The first one is propositional logic which is simpler than the other one, first-order logic. Propositional rules have no variable in their rules. In contrast, first-order rules contain variables so they are more expressive than the propositional rules. Consider the difference between these two logical rules in **Figure 1**.

Propositional logic:	IF (Name1=Jannifer)∧(Name2=Andrew)∧(Father1 = Andrew)∧(Female1=True) THEN Daughter1,2 = True
First-order logic:	IF Father(y,x) Female(y) THEN Daughter(x,y)

Figure 1. The propositional logic and first-order logic rule.

The propositional rule is specific only for the first and the second persons being Jannifer and Andrew, respectively. So it is rarely useful for unseen pairs of people. While first-order rules have variables that make the rule much more expressive and general. However, only Inductive Logic Programming (ILP) systems such as PROGOL (Mitchell 1997), FOIL (Mitchell 1997) can learn the first-order rules. These first-order rule learners have ability to select the appropriate values to substitute in variables, while other machine learning method cannot, such as Neural Networks, Bayesian Networks and Decision Tree Learning. Therefore the advantages of these methods, such as robustness to noise, could not be applied in first-order rule learning to increase the efficiency in real world usage.

[1] *Department of Computer Engineering, Chulalongkorn University, Pathumwan, Bangkok, Thailand*

In this paper, we are interested in solving this challenge. We propose the method that can use other techniques to learn first-order rules by using a framework called Multiple-Instance Learning (MIL) (Huang, Chen et al. 2003). MIL is employed for determining the appropriate values for the variable substitutions. We evaluate the proposed method by learning first-order concept *mother(x,y)* with Backpropagation Neural Network (BNN) on MIL framework. The results show that our technique competently learns first-order representation.

2. FRAMEWORK

ILP is only one of machine learning which adapts the logical concept for hypothesis learning and represents the learned rules in first-order logic form. Other learning techniques cannot directly learn this kind of rules because of difficulty in selecting the appropriate values to substitute in variables of first-order rules. To illustrate this problem, consider the task of learning rules for classifying the rich person (*rich(x)*). Background knowledge, the positive and negative examples are given as follow.

- **background knowledge:** genius(Alan), diligent(Alan), genius(Bob), strong(Bob), dilligent(Chris), strong(Chris)
- **positive example:** rich(Alan)
- **negative example:** rich(Bob), rich(Chris)

Figure 2. Input for learning the concept *rich(x)*.

From these inputs, the ILP system could provide the learned rule as: *rich(x):- genius(x), diligent(x)*. The meaning of the above rule is if x is genius and diligent, then x will be a rich person. For comparing to ILP, we use the BNN as the representative of indirectly learning methods. BNN also learns the rich concept with the same inputs as ILP. First, we create the initial network that is equivalent to the background knowledge. The network has 3 input nodes, in this case. We then assign some small random value to initialize each weight. The network constructed from the background knowledge is shown in **Figure 3**.

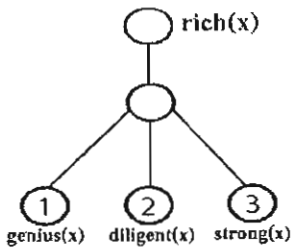


Figure 3. The constructed network.

Next, we feed an example to the network one by one. The input value for an input unit will be 1 if the literal of that unit is true in background knowledge when variables of the literal are substituted to some constants. Otherwise the input value for that unit is 0. For instance, if the fed example is *rich(Alan)*, then the value for each unit is 1, 1 and 0, respectively. Because literals *genius(Alan)* and *diligent(Alan)* are true in background knowledge, while *strong(Alan)* is not true. The target value for the output unit is 1 because *rich(Alan)* is a positive example. The target output is 0 in case of negative examples. Then we apply the Backpropagation algorithm to adjust the network weights to fit training examples. After training, the weight values of *genius(x)* and *diligent(x)* units are higher than the *strong(x)* unit. This is because the two previous literals are more significant than literal *strong(x)* and this is comparable to the induced rule from ILP.

However, when inputs are relational examples and background knowledge, neural network cannot easily learn as the previous example. As there are many constants that can be mapped to relational variables, the correct inputs for the neural network cannot be easily determined. Consider the following example.

- **background knowledge:** diligent(Alan), parent(Bob,Alan), genius(Bob), diligent(Bob), dilligent(Chris), strong(Chris)
- **positive example:** rich(Alan), rich(Bob)
- **negative example:** rich(Chris)

Figure 4. Inputs that contain relational predicate.

In this example, an additional relational literal *parent(Bob,Alan)* is given. This literal means Bob is Alan's parent. The construction of the network also adds nodes *parent(x,y)* and *parent(y,x)* in the input layer, so there are 5 input nodes in this case. When we feed positive example *rich(Alan)* to the network, the first three nodes which are *genius(x), diligent(x)* and *strong(x)* receive 0,1 and 0 as its input respectively. For node *parent(y,x),* the variable x is replaced by Alan. But we have to determine to which term (Alan, Bob or Chris) variable y should be replaced. If we select Bob for substitution, the truth value for this input unit will be 1. The other substitutions will give 0 for this unit. While the truth value for *parent(x,y)* is 0 for any substitution. This learning problem may occur when background knowledge contains relational data and the learner cannot determine the appropriate value for the variable substitution.

To solve this problem, we use the power of Multiple-Instance Learning (MIL) to provide input data for relational first-order rule learning. In MIL framework, the training set is composed of a set of bags, each of which is a collection of different number of instances. A bag is labeled as a negative bag if all the instances in it are negative. On the other hand, if a bag contains at least one positive instance then it is labeled as a positive bag. With this concept, we define a set of training examples as $\{B_1, B_2, \ldots B_n\}$, where n is a number of examples including positive and negative ones. A bag is labeled as a positive bag if an example is positive, and negative otherwise. Each bag contains m_i instances $\{B_{i1}, B_{i2}, \ldots, B_{imi}\}$ where each is one possible binding (substitution). Therefore the appropriate value selection would not be a problem because in one bag there are all cases of variable substitutions and the learning algorithm are designed for this kind of MIL problem. We can use these transformed data for learning a hypothesis. Consider an example of positive bag *rich(Alan)* as input data (see **Table 1**).

Table 1. Input data of the network for bag *rich(Alan)*.

Bag of example rich(Alan)	genius(x)	diligent(x)	strong(x)	parent(x,y)	parent(y,x)
Replace x by Alan, and y by Alan	0	1	0	0	0
Replace x by Alan, and y by Bob	0	1	0	0	1
Replace x by Alan, and y by Chris	0	1	0	0	0

As shown in **Table 1,** the bag *rich(Alan)* has 3 instances. Positive bag *rich(Bob)* and negative bag *rich(Chris)* also have 3 instances as same as bag *rich(Alan)*. For training, these 3 bags are fed to the network one by one. The network weights are adapted by the backpropagation algorithm for MIL (Zhou and Zhang 2002).

3. EXPERIMENT

In this section, we evaluate our proposed technique on learning first-order rules by BNN. The target concept that we try to learn is *mother(x,y)*. A family relationship as shown in **Figure 5** describes a set of training examples. Additionally, four predicates which are *father(x,y), husband(x,y), grandmother(x,y)* and *sister(x,y)* are given as background knowledge.

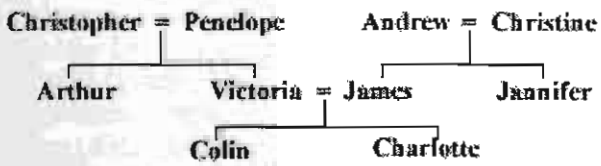


Figure 5. A family relationship where A=B means A marries B.

With these input data, we get 6 positive bags that are *mother(Penelope,Arthur), mother(Penelope,Victoria), mother(Christine,James), mother(Christine,Jannifer), mother(Victoria,Colin),* and *mother(Victoria,Charlotte).* Each positive bag is composed of 10 instances each of which is one case of relational variable *z* replacement. For negative bag, we take one person as one bag so there are 10 negative bags. Each bag has 10 cases of persons for substitution in variable *y* such as *mother(Christopher,Christopher), mother(Christopher,Penelope),* ... and each case contains 10 bindings. So each negative bag contains 90 instances, except for the 3 bags for Penelope, Christine and Victoria which contain only 70 instances as some of them are already used as positive bags. Moreover, each predicate from background knowledge is expanded to 6 literals which are for *(x,y), (x,z), (y,x), (y,z), (z,x),* and *(z,y).* Variable *z* is used for making a connection between literals. The network must be consistent with background knowledge, so we create a network with 24 input nodes and 1 output node. We set the number of hidden nodes to 1 node and 4 nodes for the first and second experiments, respectively. Then we train the network for 2000 epochs by using the backpropagation algorithm. The obtained networks are shown in **Figure 6**, with dark solid lines indicating the largest positive weights, and light lines indicating negligible weights. Both networks perfectly classified all of the training examples and the networks are almost equivalent. Furthermore from learned network weights, the network can be mapped to a rule mother(x,y) ← father(z,y), husband(z,x) (Towell and Shavlik 1993), which is the same as the rule learned by ILP.

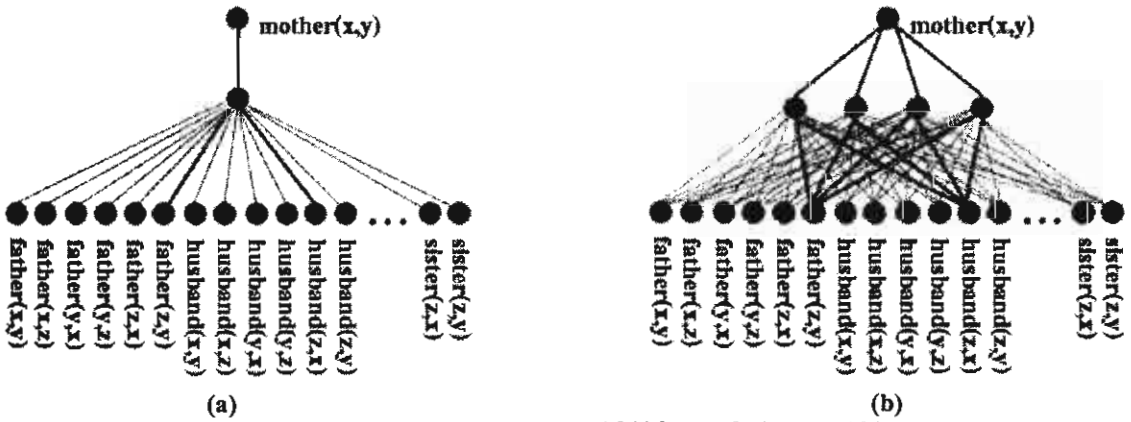


Figure 6. The complete trained networks with 1 hidden node (a) and 4 hidden nodes (b).

4. CONCLUSION

This paper presents a new framework for learning first-order rules. The objective is to solve the problem that other machine learning techniques except for ILP cannot select the appropriate values for variable substitution. So we applied MIL to provide certain input data from first-order logic input. The experimental results show that our proposed method is able to learn first-order representation. The refined network can be mapped to the rule which is comparable to one obtained by ILP. Consequently, as we can employ other techniques for learning first-order logic, the advantage of these methods can alleviate the weakness of ILP such as sensitivity to noise.

5. REFERENCES

Huang, X., S.-C. Chen, et al. (2003). An Open Multiple Instance Learning Framework and Its Application in Drug Activity Prediction Problems. Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering (BIBE'03), Bethesda, Maryland.

Mitchell, T. M. (1997). Machine Learning, The McGraw-Hill Companies Inc.

Towell, G. G. and J. W. Shavlik (1993). "The Extraction of Refined Rules from Knowledge-Based Neural Networks." Machine Learning Journal 13(1): 71-101.

Zhou, Z.-H. and M.-L. Zhang (2002). Neural Network for Multi-Instance Learning. Proceedings of the International Conference on Intelligent Information Technology, Beijing, China.

Multiclass Support Vector Machines Using Balanced Dichotomization

Boonserm Kijsirikul, Narong Boonsirisumpun, and Yachai Limpiyakorn

Department of Computer Engineering, Chulalongkorn University, Thailand
{Boonserm.K,Yachai.L}@chula.ac.th
Narong.Bo@student.chula.ac.th

The Support Vector Machine (SVM) has been introduced as a technique for solving a variety of learning and function estimation problems. The technique was originally designed for binary classification learning with its outstanding performance. However, many real world applications involve multiclass classification. Typical SVM solutions to N-class problems are to construct and combine several two-class classifiers into an N-class classifier such as the one-against-the-rest approach (1-v-r) and the one-against-one approach (1-v-1). The one-against-one methods solve $N(N-1)/2$ binary classifiers where each one is trained on data from two classes. There are different methods for the evaluation of the correct class after all $N(N-1)/2$ classifiers have been constructed. The Max Wins method takes the majority vote of a certain class as the final output [3]. A drawback of the 1-v-1 SVMs is their inefficiency of classifying data as the number of SVMs grows superlinearly with the number of classes. To improve the efficiency in classifying data, Platt et al. [5] proposed the Decision Directed Acyclic Graph (DDAG) with $N(N-1)/2$ internal nodes and N leaves. Only $N-1$ decision nodes will be evaluated in order to derive an answer, that is lower than $N(N-1)/2$ decisions required by Max Wins. To reduce the unnecessarily high number of node evaluations for the correct class, Kijsirikul, et al. [4] proposed the Adaptive Directed Acyclic Graph (ADAG) method, which is a modification of the DDAG. Like the DDAG, the ADAG requires $N-1$ decisions in order to derive an answer. However, using the reversed triangular structure reduces the number of evaluations the correct class is tested against other classes to $\lceil \log_2 N \rceil$ times or less, which is considerably lower than that of $N-1$ times required by the DDAG.

In this paper, we introduce a new method for constructing multiclass SVMs using binary classifiers, called *Balanced Dichotomization*. For an N-class problem, the system constructs $N(N-1)/2$ binary classifiers during its training phase like other one-against-one methods. Among those binary hyperplanes having been constructed, the system searches for the hyperplane at the most balanced position among all candidate classes, called *balanced dichotomization classifier* that separates the data classes into half-and-half on each side. Using a balanced dichotomization classifier can thus remove half of the candidate classes during each evaluation for the correct class, that is a higher number of elimination compared to other methods, such as the DDAG, the ADAG, which eliminate only one class using an ordinary binary classifier. As a result, the technique can optimally reduce the number of decisions in order to derive an answer to $\lceil \log_2 N \rceil$ times, rather than $N-1$ times in the DDAG and the ADAG.

The basic idea of the primary SVM classification is to find the optimal hyperplane separating the two classes of data as illustrated in Figure 2 (a). The hyperplane maximizes the margin between the data in class 1 and class 2. However, the hyperplane in Figure 2 (a) is not a balanced dichotomization classifier because when considering the positions of all candidate classes, it is not at the most balanced position as depicted in

C. Zhang, H.W. Guesgen, W.K. Yeap (Eds.): PRICAI 2004, LNAI 3157, pp. 973–974, 2004.

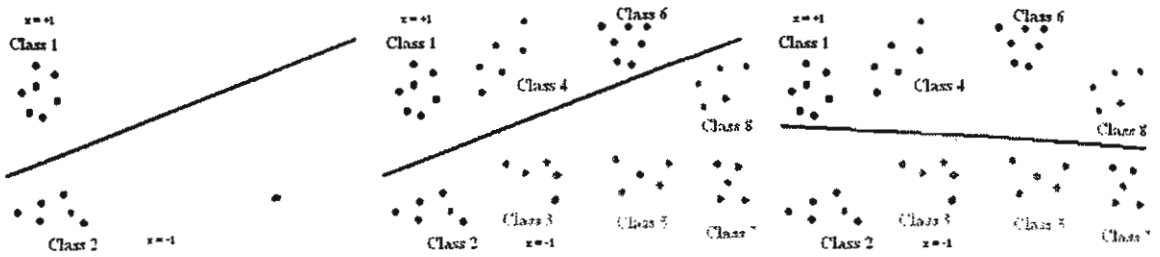


Fig. 2. (a) The optimal hyperplane for classes 1 and 2, (b) the hyperplane is not a balanced dichotomization classifier when considering other classes, and (c) an optimal balanced hyperplane.

Figure 2 (b). The hyperplane shown in Figure 2 (c) is an example of the balanced dichotomization hyperplane. It is posed at the optimal balanced position that separates candidate classes into half-and-half on each side.

Since Balanced Dichotomization requires considering positions of all candidate classes to arrive at a balanced hyperplane, there may be cases where a hyperplane in consideration is posed in between data of certain classes. To deal with these cases, two parameters are introduced in our approach, i.e. the optimal range of generalization error and the optimal pruning percentage. *Pruning percentage* is used as the threshold for the removal of data on either side of the hyperplane in consideration. The strategy of pruning is to achieve the balanced dichotomization that provides the minimum number of evaluations for the correct class while maintaining the accuracy within the range of generalization performance [1]. If the ratio between data of a class on one side and all data of the class is less than pruning percentage, the data on that side will be ignored. Moreover, using *the optimal range of generalization error*, only hyperplanes with the generalization error within the range will be considered.

We evaluate the performance of our method on sèveral datasets from the UCI Repository of machine learning databases [2]: Glass, Satimage, Segment, Shuttle, Vowel, Soybean, Letter, and Isolet. The experimental results show that Balanced Dichotomization runs faster and maintains accuracy comparable to Max Wins and better than the ADAG and the DDAG methods.

References

1. Bartlett, P. L. and Shawe-Taylor, J. (1999) *Generalization performance of support vector machines and other pattern classifiers*, In B.Schölkopf, C. Burges, & A. Smola (Eds.), Advances in Kernel Methods – Support Vector Learning, pp. 43-54, MIT Press, USA.
2. Blake, C., Keogh, E., and Merz, C. (1998) *UCI Repository of Machine Learning Databases*, Department of Information and Computer Science, University of California, Irvine. http://www.ics.uci.edu/~mlearn/MLSummary.html
3. Friedman, J. H. (1996) *Another Approach to Polychotomous classification*, Technical report, Department of Statistics, Stanford University.
4. Kijsirikul, B., Ussivakul, N., and Meknavin, S. (2002) *Adaptive Directed Acyclic Graphs for Multiclass Classification*, The Seventh Pacific Rim International Conference on Artificial Intelligence.
5. Platt, J., Cristianini, N. and Shawe-Taylor, J. (2000) *Large Margin DAGs for Multiclass Classification*, Advances in Neural Information Processing Systems, MIT Press, 12, 547-553.

First-Order Logical Neural Networks

Thanupol Lerdlamnaochai and Boonserm Kijsirikul
Department of Computer Engineering, Chulalongkorn University,
Pathumwan, Bangkok, 10330, Thailand
g46tll@cp.eng.chula.ac.th, boonserm.k@chula.ac.th

Abstract

Inductive Logic Programming (ILP) is a well known machine learning technique in learning concepts from relational data. Nevertheless, ILP systems are not robust enough to noisy or unseen data in real world domains. Furthermore, in multi-class problems, if the example is not matched with any learned rules, it cannot be classified. This paper presents a novel hybrid learning method to alleviate this restriction by enabling Neural Networks to handle first-order logic programs directly. The proposed method, called First-Order Logical Neural Network (FOLNN), is based on feedforward neural networks and integrates inductive learning from examples and background knowledge. We also propose a method for determining the appropriate variable substitution in FOLNN learning by using Multiple-Instance Learning (MIL). In the experiments, the proposed method has been evaluated on two first-order learning problems, i.e., the Finite Element Mesh Design and Mutagenesis and compared with the state-of-the-art, the PROGOL system. The experimental results show that the proposed method performs better than PROGOL.

1. Introduction

Inductive Logic Programming (ILP) [1, 2] is only one of machine learning techniques which adapts the first-order logical concepts for hypothesis learning. The advantages of ILP are the ability of employing background knowledge and the expressive representation of first-order logic. However, first-order rules learned by ILP have the restriction to handle imperfect data in real-world domains such as noisy unseen data. This problem noticeably occurs especially in multi-class classification. In multi-class classification, if an example is not covered any learned rule, it could not be classified. The simple solution is assigning the majority class recorded from training examples to the unable labeled test data [3]. Also, there is more efficient method to solve this problem by using the concept of intelligent hybrid systems [4].

Artificial Neural Networks (ANNs) [5] claim to avoid the restrictions of symbolic rule-based systems described above. Neural networks contain the ability of processing inconsistent and noisy data. Moreover, they compute the most reasonable output for each input. Neural networks, because of their potential for noise tolerance and multi-class classification, offer an attractiveness for combining with symbolic components. Although the ability of neural networks could alleviate the problem in symbolic rule-based systems, learned hypothesis from neural networks is not available in a form that is legible for humans. Therefore neural networks significantly require an interpretation by rule-based systems [4]. Several works show that the integration between robust neural networks and symbolic knowledge representation can improve classification accuracy such as Towell and Shavlik's KBANN [6], Mahoney and Mooney's RAPTURE [7], the works proposed by Rajesh Parekh and Vasant Honavar [8] and d'Avila Garcez et al. [9]. Nevertheless, these researches have been restricted to propositional theory refinement. Some models have been proposed for first-order theory. SHRUTI [10] employed a model making a restricted form of unification—actually this system only propagates bindings—. The work proposed by Botta et al. [11] created a network consisting of restricted form of learning first-order logic. Kijsirikul et al. [12] proposed a feature generation method and a partial matching technique for first-order logic but their method still uses an ILP system in its first-step learning and cannot select the appropriate values to substitute in variables.

In this paper, we are interested in direct learning of first-order logic programs by neural networks, called First-Order Logical Neural Network (FOLNN). FOLNN is a neural-symbolic learning system based on the feedforward neural network that integrates

0-7695-2291-2/05 $20.00 © 2005 IEEE

inductive learning from examples and background knowledge. We also propose the method that makes use of Multiple-Instance Learning (MIL) [13, 14] for determining the variable substitution in our model. Our proposed method has been evaluated on two standard first-order learning datasets i.e., the Finite Element Mesh Design [15] and Mutagenesis [16]. The results show that the proposed method provides more accurate result than the original ILP system.

The rest of this paper is organized as follows. Section 2 outlines the processes of first-order learning by FOLNN, splitting into three subsections. The experimental results on first-order problems are shown in Section 3. Finally the conclusions are given in Section 4.

2. First-Order Logical Neural Network (FOLNN)

Commonly the main reason for integrating robust neural networks and symbolic components is to reduce the weakness of the rule-based system. Combining these two techniques together is normally known as neural-symbolic learning system [9]. Our proposed method, FOLNN is also this type of learning system. FOLNN structure is based on the feedforward neural network and can receive examples and background knowledge in form of first-order logic programs as the inputs. FOLNN weight adaptation is based on the Backpropagation (BP) algorithm [17].

The following subsections explain the FOLNN algorithm composed of creating an initial network, feeding examples to the network and training the network.

2.1. Creating an initial network

In this subsection, we present the first step of the FOLNN algorithm, creating an initial network from background knowledge. A three layers feedforward network, composed of one input layer, one output layer and one hidden layer [18], is employed for FOLNN structure. We define the functionality of each layer as follows.

- Input layer: Input layer is the first layer that receives input data, computes received data, and then transmits processed data to the hidden layer. This layer represents the literals for describing the target rule. The number of units in this layer depends on the number of predicates in background knowledge. One predicate is represented by one unit in the input layer if that predicate contains only arity one. Otherwise, the number of units for a predicate equals to the number of all possible combinations of variables of that predicate.
- Hidden layer: This layer connects between the input layer and the output layer. The hidden layer helps the network to learn the complex classification. The number of units in this layer depends on the complication of the learning concept. The number of units in the hidden layer is determined from the experiments.
- Output layer: This layer is the last layer of the network producing the output for the classification. The target concept is represented in this layer so that the number of units in the output layer equals to the number of concepts to be learned or the number of classes.

An initial network is created by using the above definition. To illustrate the construction of the network, consider the task of learning rules for classifying the rich person (*rich(x)*). Background knowledge, the positive and negative examples are given as follows.

Figure 1. Inputs for learning the concept *rich(x)*.

As shown in Figure 1, background knowledge contains three predicates with arity one which are *genius(x)*, *diligent(x)* and *strong(x)* and one predicate having arity two which is *parent(x,y)*. Each predicate of arity one is represented by one unit in the input layer, so three units are created. Predicate *parent* is represented by two input units for literals *parent(x,y)* and *parent(y,x)*. Furthermore, the output layer, because of only one target concept *(rich(x))*, has only one unit. Therefore in this case, the constructed network will have five input units and one output unit. The created network from the inputs in Figure 1 is shown in Figure 2. In addition, all network weights are initialized to small random numbers.

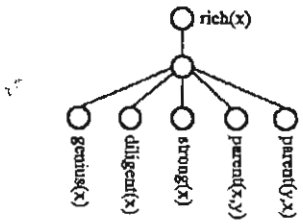


Figure 2. The created network with one hidden unit.

The completely constructed network then receives examples for refining the network. The process for feeding examples to the network is described in the next subsection.

2.2. Feeding examples to the network

In general, neural networks receive inputs in real value form. However, inputs of the ILP system (background knowledge and examples) are in logical form. So we change the logical inputs to the form that can be learned by neural networks. The examples are fed to the network one by one and independently transformed to the network input for each unit. The value for each unit is defined as follows.

$$X_{ij} = \begin{cases} 1 & \text{if } L_i\theta_j \text{ is true in background knowledge} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where X_{ij}, L_i, and θ_j are input value for input unit i when feeding example j, literal represented by input unit i, and variable binding with constants in example j, respectively.

The input value for an input unit will be 1 if there exists substitution that makes the truth value of the literal true in background knowledge. Otherwise the input value for that unit is 0. In addition, the target value for output unit is defined as follows.

$$T_{kj} = \begin{cases} 1 & \text{if } L_k\theta_j \text{ is positive example} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where T_{kj} and L_k are target value for output unit k when feeding example j, and literal represented by output unit k, respectively.

For instance, with the same inputs in Figure 1, if the fed example is *rich(Alan)* then the first three units i.e., *genius(x)*, *diligent(x)* and *strong(x)*, receive 0,1 and 0 as their inputs respectively, because literal *diligent(Alan)* is true in background knowledge, while literals *genius(Alan)* and *strong(Alan)* are false. The target value for the output unit is 1 because *rich(Alan)* is a positive example. However, the input value for literals *parent(x,y)* and *parent(y,x)* cannot be easily determined since there are many possible constants that can be mapped to relational variables. For unit *parent(y,x)*, the variable x is certainly replaced by *Alan*. However, it is quite ambiguous by which term (*Alan*, *Bob* or *Chris*) variable y should be replaced. If we select *Bob* for substitution, the truth value for this input unit will be 1. The other substitutions will give 0 for this unit (see Table 1). The truth value for *parent(x,y)* is 0 for any substitution. From the above example, the input value for unit *parent(y,x)* is not certain and cannot be easily determined for network training. This problem may occur when background knowledge contains relational data and the learner cannot determine the appropriate value for the variable substitution.

Table 1. Input value of unit *parent(y,x)* for each constant replacement.

Unit *parent (y,x)*	Input value
Replace x by *Alan*, and y by *Alan*	0
Replace x by *Alan*, and y by *Bob*	1
Replace x by *Alan*, and y by *Chris*	0

To solve this problem, we use the power of Multiple-Instance Learning (MIL) to provide input data for our network. In MIL framework [13, 14], the training set is composed of a set of bags, each of which is a collection of different number of instances. A bag is labeled as a negative bag if all the instances in it are negative. On the other hand, if a bag contains at least one positive instance then it is labeled as a positive bag. With this concept, we define FOLNN training data as a set of training examples $\{B_1, B_2, \ldots B_n\}$, where n is the number of examples including positive and negative ones. A bag is labeled as a positive bag if an example is positive, and negative otherwise (in multi-class classification, all bags are labeled as positive of their classes). The positive bag is given 1 as its target value and the negative bag is assigned 0, as defined in Equation (2). Each bag contains m_i instances $\{B_{i1}, B_{i2}, \ldots, B_{imi}\}$ where B_{ij} is one possible binding (substitution). This is a very important key because now we can use all cases of variable substitutions as one bag for learning; therefore the appropriate value selection would not be a problem. Consider an example of positive bag *rich(Alan)* as input data (see Table 2).

Table 2. Transformation of example *rich(Alan)* into input data of FOLNN

Positive bag of example *rich(Alan)*	*genius(x)*	*diligent(x)*	*strong(x)*	*parent(x,y)*	*parent(y,x)*
Replace x by *Alan*, and y by *Alan*	0	1	0	0	0
Replace x by *Alan*, and y by *Bob*	0	1	0	0	1
Replace x by *Alan*, and y by *Chris*	0	1	0	0	0

As shown in Table 2, the bag *rich(Alan)* has 3 instances each of which is one case of substitution. Also, positive bag *rich(Bob)* and negative bag *rich(Chris)* have 3 instances as same as positive bag *rich(Alan)*. For training, these 3 bags are fed to the network one by one and the network weights are adapted by the Backpropagation (BP) algorithm for MIL [19] as described in the next subsection.

2.3. Training the network

To train the network, training bags are fed to the network for adapting network weights. Weight adaptation is based on the BP algorithm and the activation function is Sigmoid function. Suppose the network has p input units, o output units, and one hidden layer. The global error function *(E)* of the network is defined as follows.

$$E = \sum_{i=1}^{n} E_i \qquad (3)$$

where E_i is the error on bag i. E_i is defined according to the type of the bag i as:

$$E_i = \begin{cases} \min\limits_{1 \le j \le m_i} \sum\limits_{k=1}^{o} E_{ijk} & \text{if } B_i = + \\ \max\limits_{1 \le j \le m_i} \sum\limits_{k=1}^{o} E_{ijk} & \text{if } B_i = - \end{cases} \qquad (4)$$

$$E_{ijk} = \begin{cases} 0 & \text{if } (B_i = +) \text{ and } (0.5 \le o_{ijk}), l_{ik} = 1 \\ 0 & \text{if } (B_i = -) \text{ and } (o_{ijk} < 0.5), \text{for all } k \\ \frac{1}{2}(l_{ik} - o_{ijk})^2 & \text{otherwise} \end{cases} \qquad (5)$$

where

- E_{ijk} is error of output unit k on instance j in bag example i
- B_i=+ is positive bag example
- B_i=– is negative bag example
- o_{ijk} is actual output of output unit k from bag example i, instance j, and
- l_{ik} is target output of output unit k from bag example i

With the defined error function above, the error BP algorithm is simply adapted for training FOLNN. In each training epoch, the training bags are fed to the network one by one. Then the error E_{ijk} is computed according to Equation (5). For a positive bag B_i, if E_{ijk} is 0 then all the rest of instances of this bag are disregarded, and the weights are not changed for this epoch. Otherwise the process continues and when all the instance of B_i are fed, E_i is computed by using Equation (4) and the weights in the network are changed according to the weight update rule of BP [17]. Then the next bag for training is fed to the network and the training process is repeated until the number of training iterations increases to some predefined threshold or the global error E in Equation (3) is decreased to some predefined threshold. After having been trained, the network can be used to classify unseen data.

3. Results

In the previous section, the three steps of learning FOLNN algorithm were described. In this section, we evaluate FOLNN by performing experiments on the finite element mesh design and the mutagenesis datasets, the well-known ILP problems. We also compare the results obtained by FOLNN with those obtained by an ILP system.

3.1. Datasets

3.1.1. Finite Element Mesh Design. The dataset for the finite element mesh design [15] consists of 5 structures and has 13 classes (13 possible number of partitions for an edge in a structure). Additionally, there are 278 examples each of which has the form *mesh(Edge,Number_of_elements)* where Edge is an edge label (unique for each edge) and *Number_of_elements* indicates the number of partitions. The background knowledge contains relations describing the types of an edge (e.g. *circuit, short*), boundary conditions (e.g. *free, fixed*), loadings (e.g. *not_loaded, one_side_loaded*) and the relations describing the structure of the object (e.g. *neighbour, opposite*). The goal of finite element mesh design is to learn general rules describing how many elements should be used to model each edge of a structure.

3.1.2. Mutagenesis. The dataset for the mutagenesis [16] consists of 188 molecules, of which 125 are mutagenic (active) and 63 are non-mutagenic (inactive). A molecule is described by listing its atoms as *atom(AtomID,Element,Type,Charge)* and the bonds between atoms as *bond(Atom1,Atom2,BondType)*. This problem is a two-class learning problem for predicting the mutagenicity of the molecules, whether a molecule is active or inactive in terms of mutagenicity.

3.2. Experiments

For the finite element mesh design dataset, we create the network containing 130 units in the input

layer (determined by predicates in background knowledge), 13 output units (as the number of classes) and one hidden layer with 80 hidden units (determined by the experiment). For the mutagenesis dataset, the constructed network has 235 input units, 100 hidden units and 2 output units. The weights of two networks are randomly initialized and then adapted by using the BP algorithm with sigmoid activation function. We performed three-fold cross validation [20] on each dataset. The dataset is partitioned into three roughly equal-sized subsets with roughly same proportion of each class as that of the original dataset. Each subset is used as a test set once, and the remaining subsets are used as the training set. The final result is the average result over three-fold data. For each fold, of both datasets, we trained FOLNN with learning rate 0.0001 and momentum 0.97.

Table 3. The percent accuracies of FOLNN and PROGOL on first-order datasets; FEM – Finite Element Mesh Design, MUTA – Mutagenesis.

Dataset	FOLNN	PROGOL
FEM	59.18	57.80
MUTA	88.27	84.58

The average results over three-fold data on FEM and MUTA datasets are summarized in Table 3. PROGOL [21], the state-of-the-art ILP system, has been used to compare the performance with our proposed method, FOLNN. The experimental results show that the accuracies of our proposed method, FOLNN are better than PROGOL in both datasets. The better results are according to the weakness of learned rules generated by PROGOL.

In addition to the results on the original dataset, to see how well our learner handles noisy data, we also evaluate FOLNN on noisy domain. The mutagenesis dataset is selected for this task. Using the three-fold data of the mutagenesis dataset in the last experiment, 10% and 15% class noise is randomly added into the training set, and no noise is added into the test set. In our case, adding x% of noise means that the class value is replaced with the wrong value in x out of 100 data by random selection. The accuracies of PROGOL and FOLNN on noisy data are shown in Table 4.

Table 4. Performance comparison on the noisy mutagenesis dataset.

Noise level in dataset	PROGOL 0% noise setting	PROGOL 10% noise setting	PROGOL 15% noise setting	FOLNN
10%	64.23	69.72	71.29	84.01
15%	60.56	61.54	65.31	81.28

Since PROGOL has an ability to handle noise in data as its option, "x% noise setting" in the table specifies that noise option of PROGOL is set to x%. As can be seen in the Table 4, our proposed algorithm still provides average accuracies higher than PROGOL. When 10% and 15% noise is added into the dataset, the PROGOL performance significantly drops due to its sensitivity to noise which is the main disadvantage of first-order rules directly induced by the ILP system. However, accuracy of our method decreased much slower and is much higher than that of PROGOL. FOLNN, because of the ability of noise tolerance by combining with neural networks, is more robust against noise than the original first-order rules. FOLNN prevents overfitting noisy data by employing neural networks to give higher weights to important features and give less attention to unimportant ones.

4. Conclusions

Learning first-order logic programs by using neural networks is still an open problem. This paper presents a novel hybrid connectionist symbolic system based on the feedforward neural network that incorporates inductive learning from examples and background knowledge, called FOLNN (First-Order Logical Neural Network). FOLNN alleviates the problem of first-order rules induced by the ILP system which are not robust enough to noisy or unseen data. The prominent advantage of FOLNN is that it can learn from inputs provided in form of first-order logic programs directly. Other learners cannot directly learn this kind of programs because they cannot select the appropriate values for variable substitution, but our method can solve this problem by applying the MIL concept to provide certain input data from first-order logic input.

The experimental results show that FOLNN presents the ability of noise tolerance and produces the better performance than PROGOL. This is because of the ability of neural networks that can select important attributes and then gives higher weights to these attributes and vice versa.

Although our main objective is to learn the first-order logic, FOLNN can be applied to other tasks such as learning from propositional datasets containing missing values in some attributes.

One interesting issue is knowledge extraction. Knowledge extraction from a trained network is one phase of the neural-symbolic learning system [9] and is of significant interest in data mining and knowledge discovery applications such as medical diagnosis. However, this phase is not included in this work and we have not yet explored rule extraction from trained networks. Nevertheless, we surmise that many researches [9, 22-24] can be adapted to extract rules from our networks.

5. Acknowledgement

This work was supported by the Thailand Research Fund.

6. References

[1] N. Lavrac and S. Dzeroski, *Inductive Logic Programming Techniques and Applications*, Ellis Horwood, New York, 1994.

[2] S.-H. Nienhuys-Cheng and R. d. Wolf, *Foundation of Inductive Logic Programming*, Springer-Verlag, New York, 1997.

[3] S. Dzeroski, S. Schulze-Kremer, K. R. Heidtke, K. Siems, and D. Wettschereck, "Applying ILP to Diterpene Structure Elucidation from 13C NMR Spectra", *Proceedings of the Sixth International Workshop on Inductive Logic Programming*, 1996.

[4] S. Wermter and R. Sun, "An Overview of Hybrid Neural Systems," *Hybrid Neural Systems, number 1778 in Lecture Notes in Artificial Intelligence*, S. Wermter and R. Sun, Eds., Springer, 2000, pp. 1-13.

[5] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[6] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks", *Artificial Intelligence*, vol. 70(1-2), 1994, pp. 119-165.

[7] J. J. Mahoney and R. J. Mooney, "Combining connectionist and symbolic learning to refine certainty-factor rule-bases", *Connection Science*, vol. 5, 1993, pp. 339-364.

[8] R. Parekh and V. Honavar, "Constructive Theory Refinement in Knowledge Based Neural Networks", *Proceedings of the International Joint Conference on Neural Networks*, Anchorage, Alaska, 1998.

[9] A. S. d. A. Garcez, K. B. Broda, and D. M. Gabbay, *Neural-Symbolic Learning Systems*, Springer-Verlag, 2002.

[10] L. Shastri and V. Ajjanagadde, "From simple associations to systematic reasoning", *Behavioral and Brain Sciences*, vol. 16, 1993, pp. 417-494.

[11] M. Botta, A. Giordana, and R. Piola, "FONN: Combining First Order Logic with Connectionist Learning", *Proceedings of the 14th International Conference on Machine Learning*, Nashville, TN, 1997.

[12] B. Kijsirikul, S. Sinthupinyo, and K. Chongkasemwongse, "Approximate Match of Rules Using Backpropagation Neural Networks", *Machine Learning Journal*, vol. 44, pp. 273-299, 2001.

[13] Y. Chevaleyre and J.-D. Zucker, "A Framework for Learning Rules from Multiple Instance Data", *12th European Conference on Machine Learning*, Freiburg, Germany, 2001.

[14] X. Huang, S.-C. Chen, and M.-L. Shyu, "An Open Multiple Instance Learning Framework and Its Application in Drug Activity Prediction Problems", *Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering (BIBE'03)*. Bethesda, Maryland, 2003.

[15] B. Dolsak and S. Muggleton, "The Application of Inductive Logic Programming to Finite Element Mesh Design", *Inductive Logic Programming*, S. Muggleton. Ed., Academic Press, 1992, pp. 453--472.

[16] A. Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King, "Theories for mutagenicity: a study in first-order and feature-based induction", *Artificial Intelligence*, vol. 85, Elsevier Science Publishers Ltd., 1996, pp. 277-299.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation", *Parallel Distributed Processing*, vol. 1, D. E. Rumelhart and J. L. McClelland, Eds., The MIT Press, Cambridge, MA, 1986.

[18] S. Holldobler and Y. Kalinke, "Towards a massively parallel computational model for logic programming", *Proceedings of the ECAI94 Workshop on Combining Symbolic and Connectionist Processing, ECAI94*, 1994, pp. 68-77.

[19] Z.-H. Zhou and M.-L. Zhang, "Neural Network for Multi-Instance Learning", *Proceedings of the International Conference on Intelligent Information Technology*, Beijing, China, 2002.

[20] T. M. Mitchell, *Machine Learning*, The McGraw-Hill Companies Inc, New York, 1997.

[21] S. Roberts, *An Introduction to Progol. Technical Manual*, University of York, 1997.

[22] R. Andrew, J. Diederich, and A. B. Tickle, "Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks", *Knowledge-Based Systems*, vol. 8, 1995, pp. 373-389.

[23] M. W. Craven, "Extracting Comprehensible Models from Trained Neural Networks", *Department of Computer Science*: University of Wisconsin-Madison, 1996.

[24] G. G. Towell and J. W. Shavlik, "The Extraction of Refined Rules from Knowledge-Based Neural Networks", *Machine Learning Journal*, vol. 13, pp. 71-101, 1993.

การเรียนรู้เน็ตเวิร์กเบย์ลำดับที่หนึ่ง
Learning First-order Bayesian Networks

รัฐฉัตร ฉัตรพัฒนศิริ บุญเสริม กิจศิริกุล

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย ถ.พญาไท ปทุมวัน กรุงเทพ 10330

Email:Ratthachat.C@Student.chula.ac.th, Boonserm.K@Chula.ac.th

บทคัดย่อ

ตัวจำแนกประเภทส่วนใหญ่มักมีข้อจำกัดที่สำคัญมากสองข้อคือ ตัวจำแนกประเภทมักแบ่งข้อมูลชนิดที่ข้อมูลแต่ละหน่วยขึ้นต่อกันได้ไม่ดี และข้อจำกัดข้อที่สองคือตัวจำแนกประเภทมักไม่สามารถแบ่งข้อมูลที่มีสัญญาณรบกวนได้อย่างมีประสิทธิภาพ "เน็ตเวิร์กเบย์ลำดับที่หนึ่ง" เป็นตัวจำแนกประเภทข้อมูลที่สามารถแก้ไขข้อจำกัดสองประการนี้ได้ อย่างไรก็ตามเนื่องจากเน็ตเวิร์กเบย์ลำดับที่หนึ่งมีความซับซ้อนมาก ระบบการเรียนรู้เพื่อสร้างเน็ตเวิร์กเบย์ลำดับที่หนึ่งจากข้อมูลดิบจึงพัฒนาได้ยากยิ่ง งานวิจัยชิ้นนี้เสนอระบบต้นแบบในการเรียนรู้เน็ตเวิร์กเบย์ลำดับที่หนึ่งจากข้อมูลดิบโดยการประยุกต์นำการโปรแกรมตรรกะเชิงอุปนัย และระบบเรียนรู้เน็ตเวิร์กเบย์เข้าไว้ด้วยกัน

Abstract

Most classifiers often struggle with two main problems when (1) there are some relations among the input data and, (2) the input data is imperfect or has some noise. A first-order bayesian network (FOBN) is a powerful classifier that can cope with those two problems. Because of its complication, however, it is very difficult to develop an efficient algorithm for constructing FOBNs. This paper proposes a new framework for constructing FOBNs by combining two algorithms, namely, inductive logic programming and a bayesian network learning algorithm.

1. บทนำ

ตัวจำแนกประเภท (classifier) ส่วนใหญ่มักมีข้อจำกัดที่สำคัญมากสองข้อ ข้อจำกัดแรกคือตัวจำแนกประเภทมักแบ่งข้อมูลชนิดที่ข้อมูลแต่ละหน่วยขึ้นต่อกันได้ไม่ดี และข้อจำกัดข้อที่สองคือตัวจำแนกประเภทมักไม่สามารถแบ่งข้อมูลที่มีสัญญาณรบกวนได้อย่างมีประสิทธิภาพ ในช่วงสิบปีที่ผ่านมานี้ได้มีการวิจัยอย่างจริงจังในการพัฒนาตัวจำแนกประเภทข้อมูลที่สามารถเอาชนะข้อจำกัดทั้งสองประเภทดังกล่าวซึ่งได้แก่*เน็ตเวิร์กเบย์ลำดับที่หนึ่ง (First-order Bayesian Network : FOBN)* [6,7] โดย FOBN ได้รวมข้อดีของตัวจำแนกประเภทข้อมูล 2 ชนิด คือ*ตรรกศาสตร์ลำดับที่หนึ่ง (First-order Logic : FOL)* [11] และตัวจำแนกประเภทข้อมูลชนิด*เน็ตเวิร์กเบย์ (Bayesian Network : BN)* [11,13] เพื่อความเข้าใจในขีดจำกัดของตัวจำแนกประเภท พิจารณาข้อมูลในตารางที่ 1

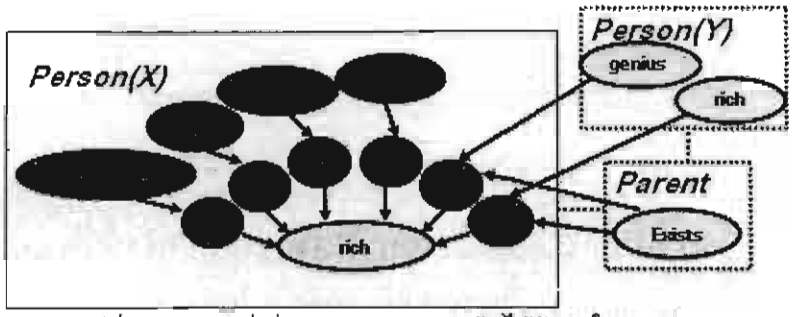
ตารางที่ 1 a. ฐานข้อมูลความรวยและคุณสมบัติของบุคคล

case	rich	genius	artist	diligent	save_money
b	+	1	0	0	0
c	+	0	1	0	1
d	+	1	1	1	1
e	+	0	0	0	0
f	-	0	1	0	0
g	-	0	0	1	0

b. ฐานข้อมูลพ่อแม่และลูก (สัมพันธ์กับตาราง a.)

parent	child
b	e
e	f
f	g

เราสามารถสร้างตัวจำแนกประเภทในรูปกฎตรรกศาสตร์ประพจน์(Propositional Logic) [11] ได้สองข้อคือ "`rich ← genius.`" และ "`rich ← artist,save_money.`" ซึ่งกฎสองข้อนี้อธิบายกรณีของนาย `e` ไม่ได้ (ในความเป็นจริง `e` รวยเพราะพ่อหรือแม่คือ `b` รวย) เนื่องจากตรรกศาสตร์ประพจน์มีความกำกวมในการแสดงกฎที่มีความสัมพันธ์ระหว่างข้อมูล เช่นกฎ "`rich ← rich.`" ไม่สามารถบอกได้ว่าความรวยของบุคคลใดส่งผลต่อบุคคลใด ในงานวิจัยนี้เราเรียกข้อมูลดิบประเภทที่ข้อมูลในแต่ละหน่วยสามารถมีผลกระทบต่อหน่วยอื่นได้ว่า*ข้อมูลเชิงสัมพันธ์ (relational data)* การมีความสัมพันธ์กันระหว่างข้อมูลทำให้การเรียนรู้ทำได้ยาก



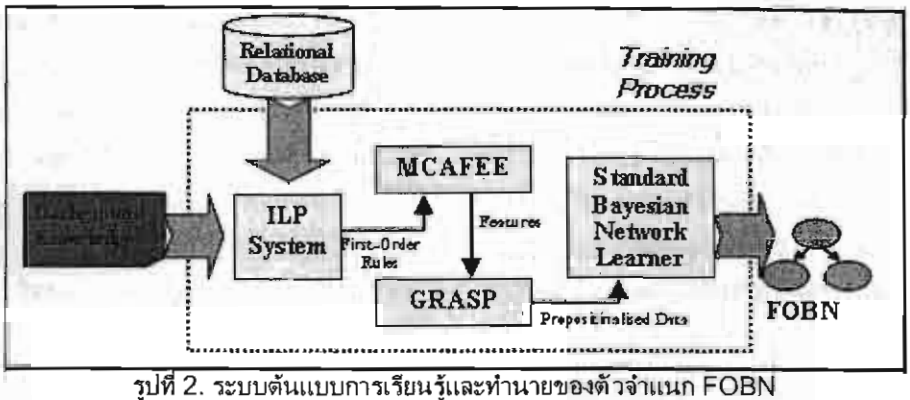
รูปที่ 1. FOBN ที่เพิ่มความสามารถจากต้นไม้ตัดสินใจและ FOL

ยิ่งขึ้นดังกรณีนาย e อย่างไรก็ตาม FOL ซึ่งถูกพัฒนาจากตรรกศาสตร์ประพจน์สามารถเขียนกฎสำหรับข้อมูลที่มีความสัมพันธ์ต่อกันได้อย่างชัดเจนโดยมีการนำตัวแปรทางตรรกศาสตร์มาใช้ดังนี้ "`rich(X) ← artist(X), save_money(X).`", "`rich(X) ← genius(X).`" และ "`rich(X) ← parent(Y,X),rich(Y),genius(Y).`" โดยในกรณีนี้ตัวแปรแต่ละตัวแสดงถึงบุคคลและกฎข้อสามครอบคลุมกรณีนาย e เนื่องจากเราสามารถสร้างกฎตรรกศาสตร์ประพจน์ด้วย FOL ได้เสมอทำให้การใช้ FOL เป็นระบบแบ่งประเภทข้อมูลช่วยลดขีดจำกัดของตัวจำแนกประเภทได้อย่างมาก สำหรับวิธีมาตรฐานในการเรียนรู้ FOL จากข้อมูลดิบนั้นถูกเรียกว่า*การโปรแกรมตรรกะเชิงอุปนัย (Inductive Logic Programming : ILP)* [10,11,12] อย่างไรก็ตามระบบ ILP ยังไม่สามารถจัดการข้อมูลที่ไม่สามารถแบ่งกลุ่มได้แน่นอนได้ดีนัก เมื่อจำเป็นต้องเรียนรู้กฎจากข้อมูลดิบที่ไม่ดีเหล่านี้ ระบบ ILP จะเผชิญกับปัญหาโอเวอร์ฟิต (overfitting problem) [10,11] ซึ่งจะทำให้ FOL ที่เรียนรู้ได้มีลักษณะเฉพาะเจาะจง (specific) กับข้อมูลอินพุตมากเกินไป (ทำงานได้ถูกต้องเฉพาะข้อมูลอินพุตเท่านั้น) พิจารณา FOL ทั้งสามข้อในตัวอย่างที่ผ่านมาพบว่าในบางกรณีกฎเหล่านี้มีความเฉพาะเจาะจงมากเกินไปเช่นกัน ตัวอย่างเช่นถ้าเพิ่มนาย a ลงไปในตารางที่ 1 โดยกำหนดคุณสมบัติให้นาย a ดังต่อไปนี้

`genius(a). save_money(a). parent(e,a).`

จะเห็นได้ว่าคุณสมบัติต่าง ๆ ของนาย a ไม่ตรงกับกฎข้อใดเลยทำให้เราสรุปจากกฎได้ว่านาย a เป็นบุคคลที่ไม่รวย อย่างไรก็ตามสังเกตว่าการสรุปแบบนี้เป็นการสรุปที่ไม่มีประสิทธิภาพนักเนื่องจากคุณสมบัติต่าง ๆ ของนาย a *ตรงบางส่วน (partially match) กับกฎ FOL ในตัวอย่างที่แล้วทั้งสามข้อ* คุณสมบัติการตรงบางส่วนของกฎนี้เองเป็นที่มาของการใช้ FOBN เป็นระบบจำแนกประเภทข้อมูล โดยเมื่อกำหนดคุณสมบัติต่าง ๆ ของนาย a ให้กับ FOBN แล้ว FOBN จะสามารถคำนวณหาค่าความจะเป็นภายหลัง (posterior probability) ที่นาย a จะรวยว่ามากน้อยเพียงใดด้วยเทคนิคการอนุมาน (inference) [13] ได้อย่างมีประสิทธิภาพ ทำให้การใช้ FOBN เป็นระบบจำแนกประเภทข้อมูลมีความยืดหยุ่นเป็นอย่างมากเนื่องจากรองรับทั้งคุณสมบัติความไม่แน่นอนของข้อมูลได้ด้วยการใช้ทฤษฎีความน่าจะเป็น (แทนที่จะสรุปว่า *ใช่* หรือ *ไม่ใช่* ซึ่งเป็นวิธีที่ระบบจำแนกข้อมูลทั่วไปใช้) นอกจากนี้ FOBN ยังมีคุณสมบัติรองรับความสัมพันธ์ระหว่างข้อมูลเช่นเดียวกับ FOL อีกด้วย รูปที่ 1 แสดง FOBN ที่ใช้แก้ปัญหาความรวยในตัวอย่างนี้

FOBN ประกอบไปด้วยสองส่วนหลักเช่นเดียวกับ BN [6,7,13] คือส่วน*โครงสร้าง (structure)* โดยหมายถึงโนดต่าง ๆ และเส้นเชื่อมแบบมีทิศทาง (directed link) ทั้งหมดที่เชื่อมโนดแต่ละโนดเข้าไว้ด้วยกัน โดยโนดแต่ละโนดแทนคุณสมบัติหรือคุณลักษณะ (attribute) การมีเส้นเชื่อมจากโนด A ไปยังโนด B หมายถึงโนด A ส่งผลกระทบโดยตรงต่อโนด B ส่วนประกอบส่วนที่สองของ FOBN คือ*ตารางความน่าจะเป็นมีเงื่อนไข (conditional probability table : CPT)* โดยใช้เพื่อคำนวณความน่าจะเป็นภายหลังเพื่อทำนายคุณสมบัติที่ต้องการ โดยทั่วไปการเรียนรู้ FOBN จึงประกอบไปด้วยสองส่วนหลักคือเรียนรู้โครงสร้างและ CPT อย่างไรก็ตามการเรียนรู้ FOBN ไม่สามารถทำได้โดยง่ายนักเนื่องจาก [1] และ [2] ได้พิสูจน์ว่าเพียงแค่การเรียนรู้ FOL หรือ BN เพียงลำพังก็ยังมีความซับซ้อนของอัลกอริทึมที่ใช้ในการเรียนรู้เป็นแบบ NP งานวิจัยนี้จึงได้เสนอแนวคิดใหม่โดยการนำระบบ ILP และ*ระบบเรียนรู้ BN (Bayesian Network Learner : BNL)* [3,13] สองระบบมาทำงานต่อเนื่องกันเพื่อลดความซับซ้อนของระบบเรียนรู้ FOBN ในการใช้ BNL เพื่อเรียนรู้ FOBN นั้น BNL ต้องการข้อมูลอินพุตประเภทฐานข้อมูลตารางเดี่ยวดังเช่น ตารางที่ 1 a. (ไม่รวมตารางที่ 1 b.) นั่นคืออินพุตประกอบไปด้วยตัวอย่าง *n* ตัวโดยตัวอย่างแต่ละตัวประกอบไปด้วยค่าของคุณสมบัติ *m* ค่า โดยตัวอย่างหนึ่งตัวแทนด้วยแถวนอนหนึ่งแถวในตารางอินพุต และหนึ่งหลักหรือแถวตั้ง (column) แทนคุณสมบัติหนึ่งตัว จากนั้นในขั้นตอน



รูปที่ 2. ระบบต้นแบบการเรียนรู้และทำนายของตัวจำแนก FOBN

การเรียนรู้ BNL จะสร้างเส้นเชื่อมที่เหมาะสมรวมทั้ง CPT ของโนดเหล่านั้นให้ ดังนั้นถ้าเราต้องการใช้ BNL เรียน FOBN ในลักษณะเดียวกับการเรียน BN เราจึงจำเป็นต้องสร้าง ฐานข้อมูลตารางเดี่ยวจากข้อมูลอินพุตประเภทข้อมูลเชิงสัมพันธ์ แต่ทว่าการสร้างฐานข้อมูลตารางเดี่ยวจากข้อมูลลักษณะนี้มีปัญหาตรงที่ไม่สามารถกำหนดคุณสมบัติที่ต้องการให้แน่ชัดได้เพราะคุณสมบัติที่เป็นไปได้ทั้งหมดในอินพุตประเภทนี้สามารถมีได้มากมายมหาศาล [4,9] ตัวอย่างเช่น อาจกำหนดให้ความรวยของพ่อแม่เป็นคุณสมบัติหนึ่ง ให้ความรวยของปู่ย่าเป็นคุณสมบัติหนึ่ง (ด้วยการเพิ่มตัวแปรเช่น `parent(Z,Y), parent(Y,X)` หมายถึง Z เป็นปู่ของ X) เห็นได้ว่าวิธีนี้สามารถเพิ่มคุณสมบัติของฐานข้อมูลตารางเดี่ยวที่ต้องการได้อย่างไม่มีที่สิ้นสุด อย่างไรก็ตาม*การแปลงหรือลดรูปปัญหาจากข้อมูลเชิงสัมพันธ์ไปเป็นฐานข้อมูลตารางเดี่ยว (propositionalization)* นั้นสามารถทำได้โดยการเลือกเฉพาะคุณสมบัติที่สำคัญหรือ*ลักษณะสำคัญ (feature)* ของข้อมูลมาเป็นแถวตั้งของฐานข้อมูลตารางเดี่ยวก็เพียงพอไม่จำเป็นต้องนำทุก ๆ คุณสมบัติที่เป็นไปได้มาสร้างเป็นแถวตั้งทั้งหมด อย่างไรก็ตามปัญหาก็คือเราไม่อาจทราบล่วงหน้าว่าต้องกำหนดความสัมพันธ์ลงไปลึกกี่ขั้นถึงจะได้ลักษณะสำคัญที่ต้องการ (ดังเช่นความสัมพันธ์แบบบรรพบุรุษในตัวอย่างที่แล้ว) ซึ่งปัญหานี้เป็นปัญหาเดียวกับการเรียนรู้ FOL ของ ILP นั่นเอง ดังนั้นงานวิจัยชิ้นนี้จึงเสนอการสร้างลักษณะสำคัญจำนวนหนึ่งที่สำคัญจริง โดยสร้างลักษณะสำคัญเหล่านั้นจาก FOL ที่ได้จาก ILP ดังเช่นจากตัวอย่างที่ผ่านมาเราสามารถบอกได้ว่าลักษณะสำคัญของข้อมูลในแง่ความสัมพันธ์ทางบรรพบุรุษจะจำกัดไม่เกินรุ่นพ่อแม่ (กฎข้อที่สาม)

ระบบต้นแบบในการเรียนรู้ FOBN ที่งานวิจัยนี้เสนอดูได้ที่รูปที่ 2 จากรูปที่ 2 อธิบายได้ดังนี้ ในขั้นแรกใช้ระบบ ILP สร้าง FOL ตามปกติ หลังจากนั้นนำ FOL ที่ได้มาสร้างลักษณะสำคัญ และนำลักษณะสำคัญเหล่านั้นมาสร้าง FOBN โดยใช้ BNL ต่อไป โดยงานวิจัยนี้เสนออัลกอริทึมสองชิ้นคือ *MCAFEE* เพื่อใช้ในการค้นหาลักษณะสำคัญต่าง ๆ จาก FOL เพื่อนำมาเป็นคุณสมบัติ (แถวตั้ง) ของฐานข้อมูลตารางเดี่ยวที่ต้องการ และอัลกอริทึมชิ้นที่สองที่เสนอ คือ *GRASP* เป็นอัลกอริทึมที่จะสร้างแถวนอนหรือตัวอย่าง (training examples) ในฐานข้อมูลตารางเดี่ยวที่ต้องการ ในงานวิจัยนี้จะเรียก FOBN ที่ถูกเรียนรู้เพื่องานทำนายข้อมูลโดยเฉพาะว่าตัว*จำแนก FOBN (FOBN classifier)* ซึ่งมีนิยามดังนี้

นิยามที่ 1 (FOBN Classifier). FOBN ใด ๆ จะถูกเรียกว่าเป็น ตัวจำแนก FOBN เมื่อมีคุณสมบัติสองข้อดังต่อไปนี้

(1) โนดของ FOBN ที่เราสนใจจะทำนายคุณสมบัติ (target node หรือ class node) จะต้องไม่มีโนดลูก (children node) ใด ๆ เลย

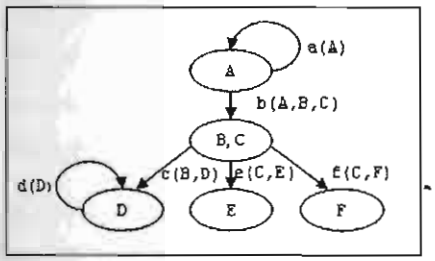
(2) โนดพ่อแม่ (parent node) ของโนดที่เราจะทำนายคุณสมบัติจะต้องเป็นลักษณะสำคัญต่าง ๆ เท่านั้น □

จุดมุ่งหมายของ (1) คือการป้องกันไม่ให้โนดที่เราต้องการทำนายไปทำนายคุณสมบัติของโนดอื่นซึ่งจะทำให้เราไม่ได้ตัวจำแนก FOBN ที่จะทำนายโนดที่ต้องการ ส่วน (2) มีจุดประสงค์ในการนำเทคนิคการตรงบางส่วนของกฎมาใช้ทำนายโนดที่ต้องการ (ดังได้แสดงข้อดีของวิธีนี้ไปแล้ว) รูปที่ 1 เป็นตัวจำแนก FOBN ที่ตรงตามนิยามที่ 1 โดยโนด `f1-f6` หมายถึงลักษณะสำคัญทั้ง 6 ตัว (ดูรายละเอียดในหัวข้อถัดไป)

2. อัลกอริทึมในการเรียนรู้ FOBN

MCAFEE (Minimal ChAin FEature Extraction) เป็นอัลกอริทึมที่ใช้ค้นหาลักษณะสำคัญจาก FOL ที่ได้จาก ILP โดยในที่นี้จะเรียกส่วนของ FOL ว่า*เชน (chain)* เราอาจเรียกลักษณะสำคัญที่งานวิจัยนี้ต้องการได้อีกอย่างว่า *เชนที่สำคัญ (significant chain)* โดยคุณสมบัติแรกของเชนที่สำคัญในงานวิจัยนี้คือต้องไม่ใช่ *เชนที่ไม่มีความหมาย (meaningless chain)* ในงานวิจัยนี้เชนที่ไม่มีความหมายคือการมีตัวแปรบางตัวในเชนถูกสร้างขึ้นมาโดยไม่ได้สัมพันธ์กับตัวแปรอื่นใน

เชน (ดู [8,9] เพิ่มเติม) เช่น พิจารณากฎ "`rich(X) :- dad(Y,X), rich(Y), good(W).`" สังเกตว่าตัวแปร `Y` ถูกสร้างขึ้นมาโดยสัมพันธ์กับตัวแปร `X` ทำให้สามารถทำความเข้าใจ `rich(Y)` ได้ ขณะที่ตัวแปร `W` ถูกสร้างขึ้นมาอย่างไม่สัมพันธ์กับตัวแปรอื่นทำให้ไม่สามารถตีความหมายได้ชัดเจน ฉะนั้นจากกฎข้อนี้เชนใด ๆ ที่มี `good(W)` อยู่จะไม่ถูกเลือกให้เป็นลักษณะสำคัญ วิธีในการสร้างเชนที่สำคัญจาก FOL ของ MCAFEE ได้แนวคิดจาก [8] โดยจะมองกฎ FOL เสมือนเป็นกราฟแบบระบุทิศทาง (directed graph) เช่นจากกฎ "`r(A) :- a(A),b(A,B,C),c(B,D),d(,D),e(C,E), f(C,F).`" สามารถสร้างกราฟแบบระบุทิศทางได้ดังรูปที่ 3



รูปที่ 3. กราฟแบบระบุทิศทาง

งานวิจัยนี้เรียกเชนที่มีความหมายอีกชื่อหนึ่งว่า *เชนที่ถูกต้อง (valid chain)* โดยนิยามได้ดังนี้

นิยามที่ 2 (valid chain). เมื่อมองกฎ FOL หนึ่ง ๆ เสมือนเป็นกราฟแบบระบุทิศทางแล้ว เส้นทาง (path) จากโนดรากไปยังโนดใด ๆ คือเชนที่ถูกต้องก็ต่อเมื่อเส้นทางนั้นมีคุณสมบัติใดคุณสมบัติหนึ่งต่อไปนี้ (1) เส้นทางนั้นมีวงวน (loop) เกิดขึ้น หรือ (2) เส้นทางนั้นมีโนดที่เป็นโนดใบ (leaf node) อยู่ในเส้นทาง □

โดยโนดใบในงานวิจัยนี้หมายถึงโนดที่ไม่มีเส้นเชื่อมออกจากตัวมันเอง เงื่อนไข (1) และ (2) ถูกสร้างขึ้นเพื่อต้องการให้แต่ละเชนที่ถูกต้องมีข้อมูลที่ครบถ้วนจากกฎ FOL เดิมให้มากที่สุดเท่าที่เป็นไปได้นั่นเอง ทั้งนี้เพื่อต้องการสร้างลักษณะที่สำคัญจริง ๆ จากกฎของ FOL (ดู [8] เพิ่มเติม) อย่างไรก็ตามในนิยามที่ 1 ได้กำหนดว่าลักษณะสำคัญทั้งหมดจะเป็นโนดพ่อแม่ของโนดที่ต้องการทำนาย ซึ่งหมายความว่าต้องมีการคำนวณค่าความน่าจะเป็นภายหลังทั้งหมดทุกรูปแบบที่เป็นไปได้ของลักษณะสำคัญหรือโนดพ่อแม่ทั้งหมดใน CPT ของโนดที่ต้องการทำนาย ดังนั้นถ้าเลือกทุก ๆ เชนที่ถูกต้องเป็นลักษณะสำคัญของโนดที่ต้องการทำนายอาจทำให้เกิดความซ้ำซ้อน (redundant) ได้โดยไม่จำเป็น ตัวอย่างเช่นจากรูปที่ 3 "`b(A,B,C), e(C,E), f(C,F)`" เป็นเชนที่ถูกต้องแต่ซ้ำซ้อนเนื่องจากสามารถเกิดจากการรวมกัน (combination) ของเชนที่ถูกต้องสองเชนคือ "`b(A,B,C), e(C,E)`" กับ "`b(A,B,C), f(C,F)`" ดังนั้นในการสร้างลักษณะสำคัญ MCAFEE จะสร้างเฉพาะเชนที่ถูกต้อง*และ*ไม่ซ้ำซ้อนเท่านั้น โดยจะเรียกเชนประเภทนี้ว่า *เชนที่ถูกต้องเล็กสุด (minimal valid chain)* และนิยามดังนี้

นิยามที่ 3 (minimal valid chain). เชนที่ถูกต้อง `r` ใด ๆ เล็กที่สุดก็ต่อเมื่อไม่มีเชนที่ถูกต้องอื่น ๆ `r'` ที่มีคุณสมบัติ $r' \subseteq r$

ตัวอย่างของเชนที่ถูกต้องเล็กสุดหรือลักษณะสำคัญจากรูปที่ 3 ที่สร้างโดย MCAFEE แสดงได้ในรูปที่ 4 และเชนที่ถูกต้องเล็กสุดทั้งหมดของ FOL สามกฎในตัวอย่างของหัวข้อที่ 1 แสดงในรูปที่ 5

```
f1(A) :- a(A).
f2(A,B,C,D) :- b(A,B,C),c(B,D),d(D).
f3(A,B,C,E) :- b(A,B,C),e(C,E).
f4(A,B,C,F) :- b(A,B,C),f(C,F).
```

รูปที่ 4. ลักษณะสำคัญจากรูปที่ 4 ที่สร้างโดย MCAFEE

```
f1(A) :- genius(A).
f2(A) :- diligent(A).
f3(A) :- artist(A).
f4(A) :- save_money(A).
f5(A,B) :- parent(B,A),rich(B).
f6(A,B) :- parent(B,A),genius(B).
```

รูปที่ 5. ลักษณะสำคัญจากตัวอย่างในหัวข้อที่ 1

อัลกอริทึม GRASP (GRound substitution AS example Propositionalization) เป็นอัลกอริทึมในการสร้างแถวนอนหรือตัวอย่างของฐานข้อมูลตารางเดี่ยว ทั้งนี้เนื่องจากบางกรณีตัวอย่างของข้อมูลที่มีความสัมพันธ์ต่อกันเมื่อแปลงรูปไปเป็นฐานข้อมูลตารางเดี่ยวตามแถวตั้งที่กำหนดแล้ว สามารถเปลี่ยนเป็นแถวใหม่มากกว่าหนึ่งได้เช่น พิจารณาตัวอย่างในหัวข้อที่ 1 อีกครั้ง สมมุติให้นาย `a` เป็นตัวอย่างหนึ่งในข้อมูลอินพุตเชิงสัมพันธ์และให้นาย `a` มีพ่อบุญธรรมอยู่อีกหนึ่งคน (นาย `h`) ซึ่งมีคุณสมบัติดังนี้ `parent(h,a). genius(h).` จากตัวอย่างนี้ เมื่อมีการแปลงข้อมูลให้ไปอยู่ในรูปฐานข้อมูลตารางเดี่ยวแล้วจะเปลี่ยนเป็นสองตัวอย่างใหม่ดังแสดงในตารางที่ 2 (ดู `f1-f6` ในรูปที่ 5)

ตารางที่ 2. ข้อมูลของนาย `a` (หัวข้อที่ 1) เมื่อแปลงให้อยู่ในรูปฐานข้อมูลตารางเดี่ยว

case	binding	f1	f2	f3	f4	f5	f6
a	A/a,B/e	1	0	0	1	1	0
	A/a,B/h	1	0	0	1	0	1

โดยกรณีแรกนาย a มีพ่อที่รวย (นาย e) ส่วนในกรณีที่สองนาย a มีพ่อที่ฉลาด (นาย h) แต่ว่าไม่ใช่คนเดียวกัน โดย [5] เสนอให้ใช้ *การแทนค่าพื้นฐาน (ground substitution : GS หรือ variable binding)* ทั้งหมดทุกกรณีเป็นตัวอย่างใหม่ทั้งหมดในฐานข้อมูลตารางเดี่ยว (โดยถือว่าตัวอย่างใหม่ทั้งหมดที่เกิดจากตัวอย่างเดิมนั้นไม่มีความข้องเกี่ยวต่อกันและกันเลย) ดังเช่น GS สองกรณีของนาย a ในตารางที่ 2 เกิดเป็นตัวอย่างใหม่สองตัว โดยวิธีนี้มีข้อดีสองข้อคือ ข้อแรกการแปลงข้อมูลประเภทนี้ทำให้เกิดข้อมูลใหม่เป็นฐานข้อมูลตารางเดี่ยวที่แท้จริงทำให้สามารถใช้ BNL เรียนรู้ได้ทันทีไม่ต้องปรับแต่งข้อมูลอีก และข้อดีข้อที่สองคือความสัมพันธ์ระหว่างโนดที่สร้างได้จากวิธีนี้จะมีความสมบูรณ์สูงกว่า (strong completeness) บางงานวิจัยเช่น [8,9] ที่เสนอให้นำเฉพาะ GS เดียวที่คิดว่าดีที่สุดมาเป็นตัวอย่างใหม่ (เนื่องจากวิธีเหล่านั้นไม่ใช้ข้อมูลทั้งหมดที่มีให้เกิดประโยชน์สูงสุด) พิจารณาตารางที่ 3

ตารางที่ 3. ตัวอย่างของความสมบูรณ์สูงกว่าใน GS

case	class	binding	f1	f2	f3	f4
E1	+	θ_{E11}	1	0	0	1
		θ_{E12}	0	1	0	0
E2	+	θ_{E21}	0	1	0	0
		θ_{E22}	1	0	0	1
E3	+	θ_{E31}	0	0	1	1
		θ_{E32}	1	0	1	0
E4	-	θ_{E41}	0	0	1	0
		θ_{E42}	1	0	0	0
E5	-	θ_{E51}	1	0	1	0
		θ_{E52}	1	0	0	0

จากตารางที่ 3 เห็นได้ว่าเนื่องจากความน่าจะเป็นภายหลัง $p(+|f2)$ และ $p(+|f4)$ มีค่า 1.0 ในขณะที่ $p(+)$ คำนวณได้ 0.6 ฉะนั้นโนด `class` จึงขึ้นกับ (depend on) โนด `f2` และ `f4` แต่ถ้าเลือกเพียงหนึ่ง GS ในตัวอย่างเดิมเป็นตัวอย่างใหม่จะทำให้คำนวณได้ว่าโนด `class` ขึ้นกับโนด `f2` หรือ `f4` โนดใดโนดหนึ่งเท่านั้นซึ่งไม่ครบถ้วน

อย่างไรก็ตามการเลือกทุก ๆ GS เป็นตัวอย่างใหม่อาจทำให้เกิดปัญหาสองอย่างคือ ปัญหาข้อแรกคือจำนวนตัวอย่างใหม่ที่ได้อาจมีมหาศาลทำให้หน่วยความจำในการเก็บ ฐานข้อมูลตารางเดี่ยวไม่เพียงพอ ปัญหาข้อที่สองคือวิธีนี้อาจทำให้เกิดความผิดพลาดในการเรียนรู้ได้เช่น กรณีในตารางที่ 4

จากตารางที่ 4 เห็นได้ว่าความน่าจะเป็นภายหลังของ $p(+|f4)$ คำนวณได้ 0.5 ในขณะที่ $p(+)$ คำนวณได้ 0.4 เนื่องจากค่าความน่าจะเป็นทั้งสองค่าใกล้เคียงกันอาจทำให้ BNL สรุปว่าโนด `class` เป็นอิสระต่อโนด `f4` ซึ่งไม่ถูกต้องเนื่อง

ตารางที่ 4. กรณีตัวอย่างเจ็ดตัวเดิมถูกเปลี่ยนเป็นสิบตัว

case	class	binding	f1	f2	f3	f4
E1	+	θ_{E11}	1	0	0	1
E2	+	θ_{E21}	1	0	0	1
E3	+	θ_{E31}	0	0	1	1
E4	+	θ_{E41}	0	0	0	1
E5	-	θ_{E51}	1	0	1	0
E6	-	θ_{E61}	1	0	1	0
E7	-	θ_{E71}	0	0	0	1
		θ_{E72}	0	0	0	1
		θ_{E73}	0	0	0	1
		θ_{E74}	0	0	0	1

จากในความเป็นจริงมีตัวอย่างลบเพียงตัวอย่างเดียว (`E7`) ที่ขัดแย้งกับตัวอย่างบวกสี่ตัวอย่าง (หรือ $p(+|f4) = 0.8$)

GRASP แก้ปัญหาทั้งสองข้อข้างต้นโดยการเลือกเพียงบาง GS เป็นตัวอย่างใหม่เท่านั้นโดยจะเลือกเฉพาะ *GS ที่ไม่ซ้ำ (non-duplicate) และมีความเฉพาะเจาะจงมากที่สุด (maximal specific binding : MSB)* โดย MSB มีนิยามดังนี้

นิยามที่ 4 (maximal specific binding). เมื่อกำหนด ω คือ GS ใด ๆ กำหนดให้ $f(\omega)$ คือเซตของลักษณะสำคัญทั้งหมดที่เป็นจริงของ ω พิจารณา GS θ ของแต่ละตัวอย่างเดิม e θ เป็น MSB ก็ต่อเมื่อไม่มี α ซึ่งเป็น GS อื่นของ e ที่ $f(\theta) \subseteq f(\alpha)$ □

จากนิยามที่ 4 พิจารณาตารางที่ 5 MSB ที่ไม่ซ้ำ (non-duplicate MSB) จากตารางมีสี่กรณีคือ θ_{E11} θ_{E13} θ_{E21} และ θ_{E22} สังเกตว่า θ_{E14} ก็เป็น MSB แต่ซ้ำกับ θ_{E11} เช่นเดียวกับ θ_{E23} ซึ่งซ้ำกับ θ_{E22} ฉะนั้นตัวอย่างใหม่ที่สร้างโดย GRASP มีเพียง GS สี่กรณีข้างต้นเท่านั้น

ตารางที่ 5. กรณีตัวอย่างสองตัวเดิมเปลี่ยนเป็นแปดตัวใหม่

case	class	θ	f1	f2	f3	f4
E1	+	θ_{E11}	1	1	0	1
		θ_{E12}	0	1	0	1
		θ_{E13}	0	1	1	0
		θ_{E14}	1	1	0	1
E2	-	θ_{E21}	1	1	0	1
		θ_{E22}	0	0	1	0
		θ_{E23}	0	0	1	0
		θ_{E24}	1	1	0	0

3. ผลการทดลองเบื้องต้น

การทดลองเบื้องต้นนี้ใช้ชุดข้อมูล (dataset) ของ*ความสามารถในการก่อกลายพันธุ์ของโมเลกุล (mutagenesis)* [8] ในการทดลองนี้ระบบต้นแบบของเราได้เลือกใช้ระบบ ILP คือ

PROGOL (เวอร์ชัน CProgol4.2) [12] ในการสร้าง FOL และใช้โปรแกรม WinMine [3] เป็น BNL ในขั้นตอนการทำนายของการทดลองครั้งนี้ใช้เทคนิค 3CV (3-fold cross-validation) [11] นอกจากนี้การทดลองนี้ยังได้เพิ่มสัญญาณรบกวนอย่างสุ่มจำนวน 10% และ 15% ในชุดข้อมูลนี้อีกด้วย เพื่อทดสอบประสิทธิภาพในการรับมือของสัญญาณรบกวนของ FOBN เนื่องจากระบบ PROGOL อนุญาตให้ผู้ใช้ปรับเปลี่ยนตัวเลือก (option) เพื่อรับมือกับสัญญาณรบกวนได้อยู่แล้ว การทดลองครั้งนี้จึงได้ลองปรับค่าตัวเลือกต่าง ๆ ของ PROGOL (ดังแสดงใน "x% noise setting") แล้วนำผลที่ได้มาเปรียบเทียบกับผลการทดลองจาก FOBN ผลการทดลองที่ได้แสดงในตารางที่ 6 เห็นได้ว่าเมื่อไม่มีสัญญาณรบกวนภายในชุดข้อมูล ผลการทำงานของ FOBN มีประสิทธิภาพใกล้เคียงกับ PROGOL แต่ว่าผลการทำงานของ FOBN ในชุดข้อมูลที่มีสัญญาณรบกวนให้ความถูกต้องสูงกว่า PROGOL อย่างมากในทุกกรณี ความถูกต้องที่สูงกว่านี้เมื่อวิเคราะห์แล้วเกิดจากสาเหตุใหญ่สองประการคือ 1) ลักษณะสำคัญที่ได้จาก MCAFEE ช่วยให้การใช้เทคนิคการตรงบางส่วนของกฎได้อย่างมีประสิทธิภาพ (ดูบทวิเคราะห์เพิ่มใน [8]) 2) การใช้ทฤษฎีความน่าจะเป็นของ FOBN ช่วยให้การใช้งานลักษณะสำคัญมีความยืดหยุ่นมากยิ่งขึ้น

4. สรุป

งานวิจัยนี้เสนอการเรียนรู้ตัวจำแนกข้อมูลชนิด FOBN ซึ่งเป็นตัวจำแนกประเภทข้อมูลที่มีความสามารถสูง เนื่องจากรวมความสามารถของ FOL และ BN ไว้ด้วยกันทำให้สามารถเรียนรู้จากข้อมูลเชิงสัมพันธ์และข้อมูลที่ไม่สามารถแบ่งกลุ่มได้แน่นอนได้ดี

กิตติกรรมประกาศ

ผู้เขียนขอขอบคุณดร.สุกรี สินธุภิญโญที่ได้แลกเปลี่ยนความคิดเห็นในงานวิจัยชิ้นนี้ ขอขอบคุณ Dr. David Maxwell Chickering และ Daniel Lowd สำหรับการช่วยเหลือในปัญหาต่าง ๆ ของโปรแกรม WinMine งานวิจัยชิ้นนี้ได้รับการสนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย (สกว.)

Reference

1. M. Botta and A. Giordana and L. Saitta and M. Sebag. Relational learning: Hard Problems and Phase transition. Selected papers from AIIA'99, Springer-Verlag, 2000.
2. D. M. Chickering. Learning Bayesian Networks is NP-Complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics* V, 1996.
3. D. M. Chickering. The WinMine Toolkit. *Technical Report MSR-TR-2002-103*, Microsoft, 2002
4. L. De Raedt. Attribute value learning versus inductive logic programming: The missing links (extended abstract). In D. Page, editor, *Proc. of the 8th Int. Conference on Inductive Logic Programming*, pages 1-8. Springer-Verlag, 1998.
5. D. Fensel, M.Zickwolff, and M. Weise. Are substitutions the better examples ? In L. De Raedt, editor, *Proc. of the 5th International Workshop on ILP*, 1995.
6. L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. *Relational Data Mining*, S. Dzeroski and N. Lavrac, editors, 2001
7. K. Kersting, L. De Raedt. Basic Principles of Learning Bayesian Logic Programs. *Technical Report No. 174*, University of Freiburg, Germany, June 2002
8. B. Kijsirikul, S. Sinthupinyo, and K. Chongkasemwongse. Approximate Match of Rules Using Backpropagation Neural Networks. *Machine Learning Journal*, 2001
9. S. Kramer, N. Lavrac and P. Flach. Propositionalization Approaches to Relational Data Mining, in: Dzeroski S., Lavrac N, editors, *Relational Data Mining*, 2001.
10. N. Lavrac and S. Dzeroski. *Inductive Logic Programming : Techniques and Applications*. Ellis Horwood, 1994
11. T. Mitchel. *Machine Learning*. McGraw-Hill, 1997.
12. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, Special issue on Inductive Logic Programming, 13(3-4):245–286, 1995.
13. J. Pearl. *Causality*. Addison Wesley, 2001.

ตารางที่ 6. เปอร์เซ็นต์ความถูกต้องในปัญหาความสามารถในการก่อกลายพันธุ์ของโมเลกุล

Noise Level in Dataset	PROGOL 0% noise setting	PROGOL 5% noise setting	PROGOL 10% noise setting	PROGOL 15% noise setting	PROGOL +FOBN
0%	84.58	82.99	77.14	77.14	84.34
10%	64.23	65.42	69.72	71.29	78.67
15%	60.56	59.02	61.54	65.31	74.33

การเรียนรู้โปรแกรมตรรกะโดยนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง
Learning Logic Programs by First-Order Neural Networks

ธนุพล เลิศลำเนาชัย และ บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย ถนนพญาไท ปทุมวัน กรุงเทพฯ 10500
Email: g46tll@cp.eng.chula.ac.th, boonserm.k@chula.ac.th

Abstract

The advantages of Inductive Logic Programming (ILP) are the ability of employing background knowledge in form of first-order logic and its highly expressive representation. Nevertheless, an ILP system is not robust enough to noisy or unseen data. Moreover in multi-class classification, if the noisy example is not matched with any learned rules, it cannot be classified. In this paper we present a new learning method that alleviates this problem by enabling Neural Networks to handle first-order logic programs directly. The proposed method is called the First-Order Neural Network (FONN). FONNs can receive First-Order Logic programs as the input of the networks. Our proposed method has been evaluated on the Finite Element Mesh Design, a first-order learning dataset. The experimental results show that the proposed method provides more accurate result than the ILP system.

บทคัดย่อ

การโปรแกรมตรรกะเชิงอุปนัยหรือระบบไอแอลพี เป็นการเรียนรู้ของเครื่องรูปแบบหนึ่งซึ่งมีข้อดีที่สามารถนำความรู้ภูมิหลังมาใช้ในการเรียนรู้ได้และกฎที่ได้สามารถอธิบายความรู้ได้อย่างกว้างขวาง อย่างไรก็ตามระบบไอแอลพีเป็นระบบที่ไม่ทนทานต่อข้อมูลที่มีสัญญาณรบกวน เมื่อนำระบบไอแอลพีมาจำแนกข้อมูลลักษณะนี้อาจทำให้การจำแนกผิดพลาดได้ เนื่องจากไม่มีกฎที่ตรงกับตัวอย่างนั้น นอกจากนี้ในกรณีที่นำกฎไปใช้จำแนกตัวอย่างแบบหลายประเภทและไม่มีกฎที่ตรงกับตัวอย่างจะทำให้ตัวอย่างนั้นไม่สามารถถูกจำแนกได้ งานวิจัยนี้นำเสนอระบบการเรียนรู้วิธีใหม่ที่นำนิวรอลเน็ตเวิร์กมาประยุกต์เข้ากับระบบไอแอลพี ทำให้ระบบยืดหยุ่นขึ้น และสามารถรับอินพุตในรูปแบบของตรรกะอันดับที่หนึ่งมาทำการเรียนรู้ได้โดยตรง โดยเรียกระบบนี้ว่า นิวรอลเน็ตเวิร์กอันดับที่หนึ่ง และจากการทดสอบกับข้อมูลการวิเคราะห์ไฟไนต์เอลิเมนต์ พบว่าผลการทดลองที่ได้จากนิวรอลเน็ตเวิร์กอันดับที่หนึ่งมีเปอร์เซ็นต์ความถูกต้องสูงกว่าระบบไอแอลพี

Key-words: Inductive Logic Programming, First-Order Logic, Neural Networks

1. บทนำ

การเรียนรู้ของเครื่อง (Machine learning)[1] เป็นแนวคิดที่ต้องการสร้างโปรแกรมคอมพิวเตอร์ให้สามารถเรียนรู้จากตัวอย่างที่ผู้สอนได้จัดเตรียมไว้ให้ โดยโปรแกรมที่สร้างขึ้นนั้นต้องสามารถสร้างแนวคิด (concept)ที่สอดคล้องกับตัวอย่างที่ได้รับจากผู้สอนและสามารถนำแนวคิดที่ได้นั้นไปใช้ในการจำแนกตัวอย่างใหม่ในอนาคตได้อย่างถูกต้อง การเรียนรู้ของเครื่องมีหลายประเภทด้วยกัน เช่น การโปรแกรมตรรกะเชิงอุปนัย (Inductive Logic Programming: ILP)[1-3] นิวรอลเน็ตเวิร์ก (Neural Networks)[1, 4] เน็ตเวิร์กเบย์ (Bayesian Networks)[1] ต้นไม้ตัดสินใจ (Decision Tree)[1] เป็นต้น การเรียนรู้ของเครื่องแต่ละประเภทก็เหมาะที่จะนำไปใช้กับงานในลักษณะต่างๆกัน

การโปรแกรมตรรกะเชิงอุปนัยหรือไอแอลพีเป็นวิธีการเรียนรู้ของเครื่องประเภทหนึ่งซึ่งนำหลักการทางตรรกะ

มาประยุกต์ใช้ ทำให้มีข้อดีที่แตกต่างจากการเรียนรู้ของเครื่องแบบอื่นๆ เนื่องจากแนวคิดที่ได้จากการเรียนจะอยู่ในรูปแบบของตรรกะอันดับที่หนึ่ง (First-Order Logic) ซึ่งเป็นลักษณะของการอธิบายความรู้ของมนุษย์ ทำให้เข้าใจได้ง่ายกว่าแนวคิดที่ได้จากวิธีอื่น นอกจากนี้ตรรกะอันดับที่หนึ่ง ยังใช้อธิบายแนวคิดที่ซับซ้อนได้ดีกว่าตรรกศาสตร์ประพจน์ (Propositional Logic) อีกด้วย ปกติแล้วระบบไอแอลพีมักจะถูกนำมาใช้ในการจำแนกตัวอย่างที่มี 2 ประเภท (class) โดยไอแอลพีจะรับอินพุตเป็นความรู้ภูมิหลัง (Background Knowledge) ตัวอย่างบวก (positive example) และตัวอย่างลบ (negative example) แล้วจะพยายามสร้างกฎหรือแนวคิดที่ครอบคลุมตัวอย่างบวกแต่ไม่ครอบคลุมตัวอย่างลบ โดยเมื่อนำกฎที่ได้ไปใช้จำแนกตัวอย่างทดสอบ ตัวอย่างนั้นจะถูกจำแนกว่าเป็นตัวอย่างบวกถ้าตรงกับกฎ แต่ถ้าไม่ตรงก็จะจำแนกตัวอย่างนั้นเป็นตัวอย่างลบ อย่างไรก็ตามถ้าตัวอย่างที่นำมาจำแนกมีความผิดพลาด เช่น ถูกสัญญาณรบกวนทำให้ข้อมูลบางส่วนหายไป ก็จะทำให้การจำแนกข้อมูลมีความผิดพลาดไปด้วย นอกจากนี้ถ้าเรานำระบบไอแอลพีไปใช้ในการจำแนกตัวอย่างแบบหลายประเภท (Multiclass Classification) ตัวอย่างที่ทำการจำแนกต้องสอดคล้องกับกฎจึงจะสามารถจำแนกได้ว่าตัวอย่างนั้นอยู่ประเภทใด ซึ่งการที่ข้อมูลมีสัญญาณรบกวนอาจทำให้ตัวอย่างนั้นไม่สามารถถูกจำแนกได้ว่าอยู่ประเภทใดและทำให้ความแม่นยำในการจำแนกตัวอย่างลดลง

เนื่องด้วยข้อจำกัดที่ไม่สามารถจำแนกตัวอย่างในลักษณะนี้ได้ จึงมีงานวิจัยที่นำเสนอแนวทางเพื่อเพิ่มความแม่นยำของการจำแนกตัวอย่าง โดยParekhและHonavar[5] ได้ประยุกต์การเรียนรู้ทางตรรกะเข้ากับนิวรอลเน็ตเวิร์กทำให้ทนทานต่อข้อมูลที่มีสัญญาณรบกวนได้ดีขึ้น แต่ว่าตรรกะที่ใช้ยังเป็นลักษณะของตรรกศาสตร์ประพจน์อยู่ Kijsirikulและคณะ[6]ได้เสนอวิธีที่สามารถใช้ตรรกะอันดับที่หนึ่งร่วมกับนิวรอลเน็ตเวิร์กได้ ทำให้มีความยืดหยุ่นมากกว่าระบบไอแอลพีและทำให้ค่าความแม่นยำสูงขึ้น แต่ว่าวิธีที่ใช้นั้นในขั้นตอนแรกจำเป็นต้องใช้ระบบไอแอลพีมาทำการสร้างกฎขึ้นมาก่อน แล้วจึงนำลักษณะสำคัญของกฎที่ได้มาเรียนรู้ด้วยนิวรอลเน็ตเวิร์กอีกครั้งหนึ่ง วิธีนี้แม้ว่าจะทำให้ผลในการจำแนกตัวอย่างดีขึ้น แต่ว่ายังจำเป็นต้องพึ่งระบบไอแอลพีอยู่

ในงานวิจัยนี้จะนำเสนอการเรียนรู้แบบใหม่ที่ยืดหยุ่นขึ้น โดยนำนิวรอลเน็ตเวิร์กมาประยุกต์เข้ากับการโปรแกรมตรรกะเชิงอุปนัย แต่ว่าสามารถรับอินพุตในรูปแบบของตรรกะอันดับที่หนึ่งมาทำการเรียนรู้ได้โดยตรง ไม่ต้องใช้ระบบไอแอลพีมาช่วยในการเรียนรู้ วิธีการที่นำเสนอนี้สามารถจัดการกับข้อจำกัดของระบบไอแอลพีได้และให้ผลที่แม่นยำมากขึ้น โดยเรียกวิธีการที่พัฒนาขึ้นนี้ว่า นิวรอลเน็ตเวิร์กอันดับที่หนึ่ง (First-Order Neural Networks: FONNs)

2. นิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

เนื้อหาในส่วนนี้จะอธิบายถึงการเรียนรู้ของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง โดยจะแบ่งอธิบายออกเป็นส่วนๆ คือ โครงสร้างของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง อินพุตของนิวรอลเน็ตเวิร์กอันดับที่หนึ่งโดยหลักการเรียนรู้แบบหลายตัวอย่าง (Multiple-Instance Learning: MIL)[7, 8] และในส่วนสุดท้ายเป็นวิธีการปรับค่าน้ำหนักของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

2.1 โครงสร้างของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

เนื่องจากนิวรอลเน็ตเวิร์กอันดับที่หนึ่งนั้นเป็นระบบการเรียนรู้ที่เราพัฒนามาจากนิวรอลเน็ตเวิร์ก ดังนั้นโครงสร้างก็จะมีพื้นฐานมาจากโครงสร้างของนิวรอลเน็ตเวิร์ก แต่ประยุกต์ให้รับตัวอย่างและความรู้ภูมิหลังในรูปแบบของตรรกะได้ โครงสร้างของนิวรอลเน็ตเวิร์กอันดับที่หนึ่งจะแบ่งออกเป็น 3 ชั้นด้วยกัน คือ ชั้นนำเข้า (input layer) ชั้นซ่อน (hidden layer) และชั้นผลลัพธ์ (output layer) ในแต่ละชั้นจะมีความหมายและหน้าที่ดังต่อไปนี้

1) ชั้นนำเข้า เป็นชั้นที่แสดงถึงสัญพจน์ (predicate) ต่างๆที่จะนำมาใช้เป็นตัวแทนของกฎหรือแนวคิดในการเรียน โดยจะเป็นชั้นแรกในการรับข้อมูลก่อนที่จะผ่านไปยังชั้นต่อๆ ไป จำนวนนิวรอนในชั้นนี้จะขึ้นอยู่กับจำนวนสัญพจน์ของความรู้ภูมิหลัง

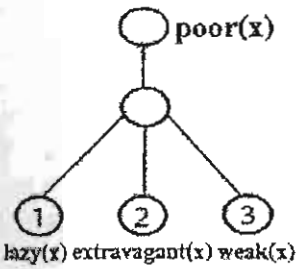
2) ชั้นซ่อน เป็นชั้นที่เชื่อมต่อระหว่างชั้นนำเข้าและชั้นผลลัพธ์ ช่วยเพิ่มความสามารถในการเรียนกฎที่มีความซับซ้อนได้ จำนวนนิวรอนในชั้นนี้มักจะกำหนดโดยดูจากความซับซ้อนของแนวคิดที่ต้องการเรียนรู้

3) ชั้นผลลัพธ์ เป็นชั้นที่แสดงถึงแนวคิดที่ต้องการให้ระบบทำการเรียนรู้ โดยชั้นนี้จะคำนวณหาผลลัพธ์สุดท้ายของเน็ตเวิร์ก จำนวนนิวรอนในชั้นผลลัพธ์จะขึ้นอยู่กับจำนวนแนวคิดทั้งหมดที่ต้องการเรียน

ตัวอย่าง เช่น การเรียนแนวคิดของ poor(x) ซึ่งมีอินพุตที่ใช้ในการเรียนเป็นดังรูปที่ 1

รูปที่ 1. อินพุตสำหรับการเรียนแนวคิด poor(x)

ตัวอย่างที่ได้รับประกอบไปด้วยตัวอย่างบวก 1 ตัว คือ poor(John) ซึ่ง John เป็นคนเกียจคร้าน (lazy(John))และขี้เหนียว (extravagant(John)) มีตัวอย่างลบ 2 ตัว คือ poor(Peter) และ poor(Bob) โดยที่ Peter เป็นคนเกียจคร้านและไม่แข็งแรง (weak(Peter)) Bob เป็นคนฟุ่มเฟือยและไม่แข็งแรง อินพุตที่ได้รับมีจำนวนสัญพจน์ที่ปรากฏในความรู้ภูมิหลัง 3 ตัวด้วยกัน คือ lazy(x), extravagant(x) และ weak(x)



รูปที่ 2. ลักษณะโครงสร้างของเน็ตเวิร์กโดยกำหนดให้ในชั้นซ่อนมี 1 นิวรอน

ดังนั้นในชั้นนำเข้าจะมีอยู่ 3 นิวรอน เพื่อแทนลักษณะทั้ง 3 สัญพจน์นั้น ส่วนจำนวนนิวรอนในชั้นผลลัพธ์จะมีเพียง 1 นิวรอนเท่านั้น เนื่องจากแนวคิดที่ต้องการเรียนมีเพียง 1 แนวคิดเท่านั้น คือ poor(x) ดังแสดงในรูปที่ 2

2.2 อินพุตของนิวรอลเน็ตเวิร์กอันดับที่หนึ่งโดยหลักการเรียนรู้แบบหลายตัวอย่างย่อย

2.2.1 อินพุตของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

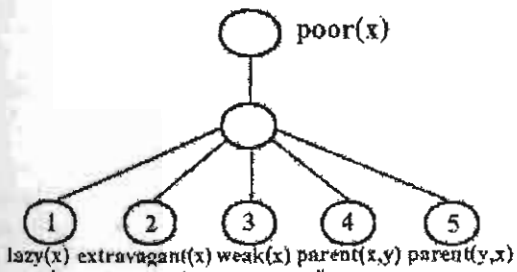
โดยปกติแล้วอินพุตของนิวรอลเน็ตเวิร์กจะอยู่ในรูปของจำนวนจริง แต่ว่าอินพุตของระบบไอแอลพีจะอยู่ในรูปของตรรกะ (ความรู้ภูมิหลังและตัวอย่าง) ดังนั้นจึงต้องเปลี่ยนอินพุตที่เป็นลักษณะของตรรกะไปเป็นอินพุตที่สามารถเรียนรู้ได้โดยนิวรอลเน็ตเวิร์กเสียก่อน การเปลี่ยนจะทำทีละ 1 ตัวอย่าง ทีละ 1 นิวรอน โดยตัวอย่าง 1 ตัวจะถูกเปลี่ยนให้เป็นค่า 1 เพื่อเป็นอินพุตของนิวรอน ถ้าแทนค่าตัวแปรในสัญพจน์ประจำนิวรอนด้วยค่าคงที่ที่ปรากฏในตัวอย่างแล้วมีค่าความจริงเป็นจริงในความรู้ภูมิหลัง แต่ถ้าไม่เป็นจริงในความรู้ภูมิหลังก็จะกำหนดให้อินพุตของนิวรอนมีค่าเป็น 0 ตัวอย่างเช่น จากตัวอย่างอินพุตในรูปที่ 1 เมื่อเปลี่ยนตัวอย่าง poor(John) ไปเป็นอินพุตของเน็ตเวิร์กแล้ว จะได้ค่าอินพุตของนิวรอน lazy(x), extravagant(x) และ weak(x) เป็น 1, 1 และ 0 ตามลำดับ เนื่องจากเมื่อแทนตัวแปร x ในสัญพจน์ของแต่ละนิวรอนด้วย John ซึ่งเป็นค่าคงที่ของตัวอย่าง poor(John) แล้วจะได้เป็น lazy(John), extravagant(John) และ weak(John) ซึ่ง lazy(John) และ extravagant(John) เป็นจริง ในความรู้ภูมิหลังจึงให้ค่าอินพุตของนิวรอนเป็น 1 ในขณะที่ weak(John) ไม่เป็นจริงในความรู้ภูมิหลัง จึงมีค่าเป็น 0 สำหรับตัวอย่าง poor(Peter) และ poor(Bob) จะได้ค่าอินพุตของนิวรอนเป็น 1,0,1 และ 0,1,1 ตามลำดับ

อย่างไรก็ตามเนื่องจากแนวคิดที่ได้จากระบบไอแอลพีอาจมีการสร้างตัวแปรใหม่ขึ้นในสัญพจน์โดยที่ตัวแปรนั้นไม่ปรากฏอยู่ในส่วนหัวของกฎ ตัวแปรใหม่ที่สร้างขึ้นนั้นเพื่อนำมาใช้อธิบายความสัมพันธ์ที่เกี่ยวข้อง (relation) กันของแต่ละสัญพจน์ อย่างเช่น poor(x) ← parent(y,x), poor(y), lazy(x) ซึ่งมีการสร้างตัวแปร y ขึ้นมาเพื่อแสดงความสัมพันธ์

ระหว่างสัญพจน์ parent(y,x) และ poor(y) แต่ว่าการสร้างตัวแปรขึ้นใหม่ในลักษณะนี้จะทำให้การสร้างอินพุตสำหรับนิวรอนเกิดปัญหาความไม่แน่นอนขึ้น เช่น ถ้าความรู้ภูมิหลังและตัวอย่างที่รับเข้ามามีลักษณะเป็นดังรูปที่ 3

รูปที่ 3. อินพุตที่เพิ่มสัญพจน์ parent(x,y) ในความรู้ภูมิหลัง

ในกรณีนี้โครงสร้างของเน็ตเวิร์กจะมีความซับซ้อนมากขึ้นเนื่องจากในความรู้ภูมิหลังมีสัญพจน์ parent(x,y) เพิ่มเข้ามา และเนื่องจากสัญพจน์นี้มีอาร์กิวเมนต์ 2 ตัวและเราไม่สามารถรู้ได้ว่าอาร์กิวเมนต์ 2 ตัวนี้ควรมีการวางตำแหน่งอย่างไร จึงได้ทำการสร้างนิวรอนขึ้นมาสำหรับทั้ง 2 กรณี คือ parent(x,y) และ parent(y,x) ทำให้จำนวนนิวรอนในชั้นนำเข้าเพิ่มขึ้นอีก 2 นิวรอน ดังรูปที่ 4 แต่ว่านิวรอนที่เพิ่มขึ้นมานี้มีตัวแปรใหม่อยู่ด้วย ทำให้มีปัญหาในการกำหนดค่าอินพุตให้กับนิวรอน เช่น ตัวอย่าง poor(John) เมื่อจะกำหนดค่าอินพุตให้กับนิวรอน parent(y,x) ก็จะแทนที่ตัวแปร x ด้วยค่าคงที่ John ส่วนตัวแปร y ซึ่งไม่มีการกำหนดค่ามาจากตัวอย่าง ทำให้ไม่สามารถระบุได้ว่าต้องแทนตัวแปร y ด้วยค่าคงที่ใด และการแทนด้วยค่าคงที่แต่ละแบบนั้นก็จะให้ค่าอินพุตของนิวรอนแตกต่างกันดังแสดงในตารางที่ 1



รูปที่ 4. เน็ตเวิร์กที่มีนิวรอนในชั้นนำเข้า 5 นิวรอน

ส่วนกรณีของ parent(x,y) ไม่เกิดปัญหาเนื่องจากเมื่อแทนค่าตัวแปร x ด้วย John แล้วไม่ว่าจะแทนค่าตัวแปร y ด้วยค่าคงที่ใด ก็ไม่เป็นจริงในความรู้ภูมิหลัง ทำให้ค่าอินพุตของนิวรอน parent(x,y) เป็น 0 ในทุกกรณี

ตารางที่ 1. ค่าอินพุตของนิวรอน parent(y,x) เมื่อแทนตัวแปรด้วยค่าคงที่ต่างๆ

parent(y,x)	ค่าอินพุตของนิวรอน
แทน x ด้วย John แทน y ด้วย John	0
แทน x ด้วย John แทน y ด้วย Peter	1
แทน x ด้วย John แทน y ด้วย Bob	0

จากตัวอย่างข้างต้นแสดงให้เห็นว่าการกำหนดอินพุตให้กับนิวรอนในกรณีที่สัญพจน์ประจำนิวรอนนั้นมีตัวแปรที่ใช้แสดงความสัมพันธ์กับสัญพจน์อื่น โดยตัวแปรนั้นปรากฏอยู่ในส่วนหัวของกฎด้วย ในบางกรณีจะทำให้สามารถกำหนดค่าที่แน่นอนให้กับนิวรอนได้ เนื่องจากไม่ทราบว่าควรแทนค่าให้กับตัวแปรที่แสดงความสัมพันธ์ด้วยค่าใด ได้นำหลักการของการเรียนรู้แบบหลายตัวอย่างย่อย (Multiple Instance Learning: MIL)[7, 8]มาประยุกต์เพื่อทำการสร้างอินพุตให้กับนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง ดังกล่าวในหัวข้อ 2.2.2 ด้านล่างนี้

2.2.2 หลักการเรียนรู้แบบหลายตัวอย่างย่อย

การเรียนรู้แบบหลายตัวอย่างย่อยจะนิยามตัวอย่างการเรียนรู้เป็นถุง (bag) $\{B_1, B_2, \ldots, B_n\}$ โดยที่ n เป็นจำนวนของถุง แต่ละถุง (B_i) จะประกอบไปด้วยตัวอย่าง (instance) m_i ตัว $\{B_{i1}, B_{i2}, \ldots, B_{imi}\}$ โดยที่ถุงหนึ่งๆจะกำหนดให้เป็นถุงตัวอย่างบวก (positive bag) ก็ต่อเมื่อในนั้นมีตัวอย่างย่อยอย่างน้อย 1 ตัวที่เป็นบวก และจะกำหนดเป็นถุงตัวอย่างลบ (negative bag) ก็ต่อเมื่อตัวอย่างย่อยทุกตัวในถุงนั้นเป็นตัวอย่างลบทั้งหมด โดยที่ถุงตัวอย่างบวกจะกำหนดให้มีค่าเป้าหมาย (label) เป็น 1 ส่วนถุงตัวอย่างลบจะมีค่าเป็น 0

การแปลงตัวอย่างไปเป็นอินพุตให้กับเน็ตเวิร์ก จะแทนตัวอย่างบวก 1 ตัว แทนด้วยถุงตัวอย่างบวก 1 ถุง และแทนตัวอย่างลบ 1 ตัวด้วยถุงตัวอย่างลบ 1 ถุง (ในกรณีที่เป็นค

ย่างแบบหลายประเภทจะถือว่าทุกถุงเป็นถุงตัวอย่าง
ประเภทนั้นๆ) ในถุงจะประกอบไปด้วยตัวอย่างย่อยซึ่ง
มาจากการแทนค่าตัวแปรด้วยค่าคงที่แต่ละแบบ
จากตัวอย่างอินพุตตามรูปที่ 3 ในการเรียนแนวคิดของ
จะมีถุงตัวอย่างบวกทั้งหมด 2 ถุง คือ poor(John) และ poor(Peter) ถุงตัวอย่างลบ 1 ถุง คือ poor(Bob) และในแต่ละถุงจะประกอบไปด้วยตัวอย่างย่อยทั้งหมด 3 ตัว จากการแทนค่าตัวแปร y ด้วยค่า John, Peter และ Bob ดังแสดงตัวอย่างในตารางที่ 2

ตารางที่ 2. การแปลงตัวอย่างของ poor(John) ไปเป็นอินพุตให้กับนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

ถุงตัวอย่าง ของ poor(John)	lazy(x)	extravagant(x)	weak(x)	parent(x,y)	parent(y,x)
แทน x ด้วย John แทน y ด้วย John	1	0	0	0	0
แทน x ด้วย John แทน y ด้วย Peter	1	0	0	0	1
แทน x ด้วย John แทน y ด้วย Bob	1	0	0	0	0

จากนั้นจึงนำถุงตัวอย่างที่ได้ไปใส่เป็นอินพุต
เน็ตเวิร์กและทำการเรียนรู้โดยอาศัยอัลกอริทึมแบ็ค
พรอพาเกชันอัลกอริทึม (Backpropagation Algorithm)
การเรียนรู้แบบหลายตัวอย่างย่อย[9, 10]

การปรับค่าน้ำหนักของนิวรอลเน็ตเวิร์กอันดับที่หนึ่ง

การปรับค่าน้ำหนักของเส้นเชื่อมนั้นจะทำทั้งส่วนที่
ระหว่างชั้นนำเข้ากับชั้นซ่อน และระหว่างชั้นซ่อนกับชั้น
การปรับค่าน้ำหนักจะนำหลักการของแบ็กพรอพาเกชัน
มาใช้ โดยนิยามค่าความผิดพลาดรวม (Global Error:

$$E = \sum_{i=1}^{n} E_i \qquad (1)$$

เป็นค่าความผิดพลาดของถุงตัวอย่างแต่ละถุง ซึ่งนิยาม
ขึ้นกับประเภทของถุงตัวอย่างนั้นๆ คือ

$$E_i = \begin{cases} \min\limits_{1\le j\le m_i} E_{ij} & \text{if } B_i = + \\ \max\limits_{1\le j\le m_i} E_{ij} & \text{if } B_i = - \end{cases} \qquad (2)$$

ความผิดพลาดของตัวอย่างย่อยนิยามดังนี้

$$E_{ij} = \begin{cases} 0 & \text{if } (B_i = +) \text{ and } (0.5 \le o_{ij}) \\ 0 & \text{if } (B_i = -) \text{ and } (o_{ij} < 0.5) \\ \frac{1}{2}(o_{ij} - l_i)^2 & \text{otherwise} \end{cases} \qquad (3)$$

$B_i = +$ หมายถึง ถุงตัวอย่างบวก

$B_i = -$ หมายถึง ถุงตัวอย่างลบ

o_{ij} หมายถึง ผลลัพธ์ที่ได้จากถุงตัวอย่างที่ i ตัวอย่างย่อยที่ j (B_{ij})

l_i หมายถึง ค่าเป้าหมายของถุงตัวอย่างที่ i

ในการเรียนแต่ละรอบ ตัวอย่างที่ใช้ในการสอนจะถูกป้อนให้กับเน็ตเวิร์กทีละถุงทีละตัวอย่างย่อย จากนั้นจะทำการคำนวณหาค่าความผิดพลาดของแต่ละตัวอย่างย่อยในถุงนั้นๆ ตามสมการ (3) ถ้ากรณีที่ถุงตัวอย่างที่ป้อนให้กับเน็ตเวิร์กเป็นถุงตัวอย่างบวกและพบว่ามีตัวอย่างย่อยที่ให้ค่าความผิดพลาดเป็น 0 แล้ว ตัวอย่างย่อยที่เหลือไม่จำเป็นต้องป้อนให้กับเน็ตเวิร์กและไม่ต้องทำการปรับค่าน้ำหนักเส้นเชื่อมใดๆ เพราะถือว่าตัวอย่างย่อยตัวนั้นเป็นตัวอย่างที่ทำให้ถุงตัวอย่างเป็นบวกและเน็ตเวิร์กจำแนกได้ถูกต้องแล้ว แต่ถ้าเป็นกรณีอื่นๆค่าความผิดพลาดของถุงตัวอย่างก็จะเป็นไปตามสมการ (2) จากนั้นเมื่อป้อนครบทุกตัวอย่างย่อยในถุงแล้วก็จะนำค่าความผิดพลาดของถุงที่ได้ไปทำการปรับค่าน้ำหนักของเส้นเชื่อมตามวิธีของแบ็กพรอพาเกชัน แล้วจึงทำการป้อนถุงตัวอย่างใหม่เข้าไปและทำตามลำดับขั้นตอนเดิม โดยจะหยุดกระบวนการเรียนรู้เมื่อค่าความผิดพลาดรวมในสมการ (1) มีค่าลดลงจนถึงจุดที่กำหนดไว้หรือเรียนรู้จนครบจำนวนรอบที่ได้กำหนดไว้

3. การทดลองและผลการทดลอง

ในการทดลองเรานำชุดข้อมูล (Dataset) การวิเคราะห์ไฟไนต์เอลิเมนต์ (Finite Element Mesh Design: FEM)[11] มาใช้ทำการทดลอง ปัญหา FEM มีจุดมุ่งหมายเพื่อหากฎที่ใช้

ในการวิเคราะห์ไฟไนต์เอลิเมนต์ในโครงสร้างสำหรับงานทางด้านวิศวกรรม มีกลุ่มความรู้ภูมิหลังเป็นลักษณะต่างๆของโครงสร้าง ประกอบด้วยลักษณะของเส้นเชื่อม เช่น long และ circuit ฯลฯ เงื่อนไขขอบเขต เช่น free และ fixed ฯลฯ โลด เช่น not_loaded และ one_side_loaded ฯลฯ และตัวอย่าง 12 ประเภทด้วยกัน โดยจะแบ่งประเภทตามจำนวนองค์ประกอบ (Element) ที่เหมาะสมของโครงสร้างนั้น ตัวอย่างแต่ละตัวถูกจัดอยู่ในรูปแบบ mesh(Edge, Element) เมื่อ Edge คือชื่อโครงสร้างและ Element คือ จำนวนองค์ประกอบภายในโครงสร้างนั้น รวมจำนวนตัวอย่างทั้งหมด 278 ตัวอย่าง ในการทดลองนั้นใช้วิธี 4-fold cross validation [1] ทำโดยแบ่งข้อมูลทั้งหมดออกเป็น 4 ส่วนเท่าๆกัน ทำการทดลองทั้งหมด 4 ครั้ง ในแต่ละครั้งจะเลือกส่วนหนึ่งใดๆเป็นชุดทดสอบและส่วนที่เหลือ 3 ส่วนจะใช้เป็นชุดสอน

โครงสร้างของเน็ตเวิร์กที่ใช้จะมีจำนวนนิวรอนในชั้นนำเข้าทั้งหมด 130 นิวรอน กำหนดจากความรู้ภูมิหลังที่ได้รับเข้ามา นิวรอนในชั้นผลลัพธ์ 12 นิวรอน กำหนดจากประเภทของตัวอย่าง และใช้จำนวนนิวรอนในชั้นซ่อน 80 นิวรอน(จากการทดลอง) มีค่าอัตราการเรียนรู้เป็น 0.0001 และค่าโมเมนตัมเป็น 0.97 โดยกระบวนการเรียนรู้จะหยุดเมื่อครบ 6000 รอบ หรือค่าความผิดพลาดรวมมีค่าน้อยกว่า 0.05

ตารางที่ 3. ผลเปรียบเทียบเปอร์เซ็นต์ความถูกต้องในการทดสอบกับชุดข้อมูล FEM

อัลกอริทึม	เปอร์เซ็นต์ความถูกต้อง
FONN	59.18
PROGOL	57.80 [6]

ตารางที่ 3 แสดงผลการทดลองที่ได้ในการทดสอบกับชุดข้อมูล FEM เปรียบเทียบระหว่าง FONN (วิธีการที่นำเสนอ) กับ PROGOL ซึ่งเป็นระบบไอแอลพีที่มีประสิทธิภาพมากสุดระบบหนึ่งในปัจจุบัน จะเห็นว่า FONN สามารถเรียนแนวคิดจากชุดข้อมูลตรรกะอันดับที่หนึ่งได้และให้เปอร์เซ็นต์ความถูกต้องในการจำแนกสูงกว่า PROGOL[12]

4. สรุป

งานวิจัยนี้ได้นำเสนอนิวรอลเน็ตเวิร์กแบบไ
ประยุกต์เข้ากับแนวคิดของการเรียนรู้เชิงตรรกะ เรียกว่า นิ
เน็ตเวิร์กอันดับที่หนึ่ง เพื่อเป็นแนวทางหนึ่งในการนำ
จัดการกับข้อจำกัดของการโปรแกรมตรรกะเชิงอุปนัย
นิวรอลเน็ตเวิร์กอันดับที่หนึ่งมีข้อดีที่สามารถรับอิน
รูปแบบของตรรกะอันดับที่หนึ่งได้โดยตรงและมีข้อ
นิวรอลเน็ตเวิร์กซึ่งทนทานต่อข้อมูลที่มีสัญญาณรบกวนได้
ด้วย

5. รายการอ้างอิง

[1] T. M. Mitchell, *Machine Learning*: The McGraw
Companies Inc., 1997.

[2] N. Lavrac and S. Dzeroski, *Inductive L
Programming Techniques and Applications*:
Horwood, New York, 1994.

[3] S.-H. Nienhuys-Cheng and R. d. Wolf, *Foundati
Inductive Logic Programming*: Springer-Verlag
York, Inc., 1997.

[4] C. M. Bishop, *Neural Networks for Pa
Recognition*: Oxford University Press, 1995.

[5] R. Parekh and V. Honavar, "Constructive Th
Refinement in Knowledge Based Neural Networ
Proceedings of the International Joint Conferenc
Neural Networks, Anchorage, Alaska, 1998.

[6] B. Kijsirikul, S. Sinthupinyo, and
Chongkasemwongse, "Approximate Match of
Using Backpropagation Neural Networks," *Mac
Learning Journal*, vol. 44, pp. 273-299, 2001.

[7] Y. Chevaleyre and J.-D. Zucker, "A Framewor
Learning Rules from Multiple Instance Data,"
European Conference on Machine Learning, Frei
Germany, 2001.

[8] X. Huang, S.-C. Chen, and M.-L. Shyu, "An O
Multiple Instance Learning Framework and
Application in Drug Activity Prediction Proble
Proceedings of the Third IEEE Symposium
BioInformatics and BioEngineering (BIBE
Bethesda, Maryland, 2003.

[9] P. Wattuya, A. Rungsawang, and B. Kijsir
"Multiple-Instance Neural Network with Str
Boundary Criteria," The 7th National Comp
Science and Engineering Conference, Chon
Thailand, 2003.

[10] Z.-H. Zhou and M.-L. Zhang, "Neural Network
Multi-Instance Learning," Proceedings of
International Conference on Intelligent Informati
Technology, Beijing, China, 2002.

[11] B. Dolsak and S. Muggleton, "The Application
Inductive Logic Programming to Finite Element M
Design," in *Inductive Logic Programming*,
Muggleton, Ed.: Academic Press, 1992, pp. 453–47

[12] S. Roberts, *An Introduction to Progol. Techni
Manual*: University of York, 1997.

เทคนิคการแตกครึ่งตามสารสนเทศ สำหรับซัพพอร์ตเวกเตอร์แมชชีนแบบหลายประเภท

An Information-Based Dichotomization Technique for Multiclass Support Vector Machines

ทมศิริ สงศิริ และ บุญเสริม กิจศิริกุล
ภาควิชาวิศวกรรมคอมพิวเตอร์
วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ถนนพญาไท ปทุมวัน กรุงเทพฯ 10330
boonsiri@hotmail.com and boonserm.k@chula.ac.th

ฐิมาพร เพชรแก้ว
สำนักวิชาสารสนเทศศาสตร์
มหาวิทยาลัยวลัยลักษณ์ เลขที่ 222 ต.ไทรบุรี
อ.ท่าศาลา จ.นครศรีธรรมราช 80160
Email: pthimapo@wu.ac.th

บทคัดย่อ

การแก้ปัญหาการจำแนกแบบหลายประเภทด้วยซัพพอร์ตเวกเตอร์แมชชีน โดยส่วนใหญ่จะพิจารณาเป็นปัญหาของการนำตัวจำแนกแบบสองประเภทหลายตัวมาใช้ร่วมกัน ซึ่งการดังกล่าวยังมีข้อจำกัดในเรื่องของความถูกต้องและระยะเวลาครั้งในการจำแนก บทความนี้นำเสนอวิธีการใหม่ในการจำแนกข้อมูลด้วยเทคนิคการแตกครึ่งตามสารสนเทศโดยสร้างต้นไม้สำหรับการจำแนกแบบไบนารีจากตัวจำแนกแบบสองประเภท ซึ่งแต่ละโนดของต้นไม้จะเป็นตัวจำแนกแบบสองประเภทที่มีค่าเอนโทรปีต่ำที่สุด วิธีนี้สามารถลดจำนวนตัวจำแนกแบบสองประเภทที่ใช้ในการจำแนกลงได้ในอัลกอริทึมของจำนวนประเภทซึ่งต่ำกว่าวิธีอื่น จากผลการทดลองแสดงให้เห็นว่าวิธีนี้สามารถลดจำนวนครั้งของการจำแนกลงได้ และยังคงให้ค่าความถูกต้องใกล้เคียงกับวิธีอื่น

คำสำคัญ: การแตกครึ่งตามสารสนเทศ, ซัพพอร์ตเวกเตอร์แมชชีนแบบหลายประเภท, เอนโทรปี

Abstract

Approaches for solving a multiclass classification problem by Support Vector Machines (SVMs) are typically to consider the problem as combination of two-class classification problems. Previous approaches have some limitations in classification accuracy and evaluation time. This paper proposes a novel method that employs information-based dichotomization for constructing a binary classification tree. Each node of the tree is a binary SVM with the minimum entropy. Our method can reduce the number of binary SVMs used in the classification to the logarithm of the number of classes which is lower than previous methods. The experimental results show that the proposed method takes lower evaluation time while it maintains accuracy compared to other methods.

Key Words: Information-Based Dichotomization, Multiclass Support Vector Machines, Entropy

1. บทนำ

ซัพพอร์ตเวกเตอร์แมชชีนเป็นเทคนิคการเรียนรู้ที่ใช้คุณสมบัติเชิงเรขาคณิตในการคำนวณหาระนาบหลายมิติ (Hyperplane) ที่ดีที่สุดในการแยกข้อมูลออกจากกัน โดยในช่วงแรกเริ่มนั้นเทคนิคนี้มีข้อจำกัดอยู่ที่สามารถจำแนกข้อมูลได้เพียงสองประเภทเท่านั้น แต่ในปัญหาการจำแนกส่วนใหญ่มักเป็นแบบหลายประเภท ต่อมาจึงมีผู้พัฒนาเทคนิคนี้ให้สามารถใช้ได้กับการจำแนกแบบหลายประเภท วิธีการส่วนใหญ่มักจะเป็นการนำฟังก์ชันที่จำแนกแบบสองประเภทหลายฟังก์ชันมาใช้ร่วมกัน เช่น การจำแนกแบบหนึ่งต่อที่เหลือ (One-against-the-rest) เป็นการเปรียบเทียบประเภทข้อมูลหนึ่งกับประเภทอื่นที่เหลือทั้งหมด และการจำแนกแบบ

หนึ่งต่อหนึ่ง (One-against-one) ซึ่งเป็นการเปรียบเทียบประเภทข้อมูลหนึ่งกับประเภทข้อมูลอื่นทีละประเภท [6]

อัลกอริทึมแมกซ์วินเป็นหนึ่งในวิธีการจำแนกแบบหนึ่งต่อหนึ่งซึ่งให้ค่าความถูกต้องสูงกว่าการจำแนกแบบหนึ่งต่อที่เหลือ แต่ใช้เวลาในการจำแนกนานกว่า [5] ต่อมา Phetkaew, et al. ได้เสนอวิธีอาร์เอดีเอจี (Reordering Adaptive Directed Acyclic Graph (RADAG)) [8] ซึ่งวิธีนี้จะมีการเลือกลำดับของโนดที่เหมาะสมซึ่งมีโอกาสที่จะจำแนกผิดพลาดน้อย โดยใช้อัลกอริทึมของการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุด (Minimum-Weight Perfect Matching) [4] และมีการจัดเรียงโนดใหม่ในทุกชั้นของการจำแนก ทำให้มีค่าความผิดพลาดลดลงจากวิธีเอดีเอจี [9] อย่างไรก็ตามในวิธีที่กล่าวมาข้างต้นสำหรับการจำแนกข้อมูล k ประเภทใดๆ วิธีอาร์เอดีเอจีต้องใช้จำนวนครั้งของการจำแนกข้อมูลเป็น k-1 ครั้ง เนื่องจากในการจำแนกข้อมูลแต่ละครั้งสามารถตัดข้อมูลที่ไม่ถูกต้องออกไปได้เพียง 1 ประเภทต่อครั้งเท่านั้น ต่อมา Kijsirikul & Boonsirisumpun ได้เสนอการสร้างต้นไม้สำหรับการจำแนกแบบหลายประเภทด้วยวิธีการแตกครึ่งแบบสมดุล (Balanced Dichotomization Classification) [7] ที่ทำการจำแนกข้อมูลโดยค้นหาระนาบที่แบ่งข้อมูลในตำแหน่งที่สมดุลที่สุดของข้อมูลทั้งหมดในแต่ละรอบมาจำแนก เพื่อให้การจำแนกข้อมูลแต่ละครั้งสามารถตัดประเภทที่ไม่ถูกต้องออกไปได้มากกว่า 1 ประเภท ทั้งนี้ในการค้นหาระนาบนั้นจะเลือกระนาบที่แบ่งข้อมูลแล้วให้ค่าจำนวนประเภทซึ่งตกอยู่ในแต่ละด้านของระนาบมีค่าใกล้เคียงกันมากที่สุด ภายใต้สมมติฐานที่ว่าความน่าจะเป็นในการเกิดข้อมูลในแต่ละประเภทของข้อมูลสอนมีค่าใกล้เคียงกับข้อมูลทดสอบ วิธีนี้จะมีความเหมาะสมกรณีที่ข้อมูลแต่ละประเภทมีความน่าจะเป็นในการเกิดที่เท่ากัน โดยวิธีนี้จะพยายามสร้างต้นไม้ให้ทั้งสองกิ่งมีความสมดุลที่สุด ซึ่งส่งผลให้ค่าจำนวนครั้งในการจำแนกข้อมูลแต่ละประเภทจะมีค่าใกล้เคียงกัน ในความเป็นจริงข้อมูลที่นำมาจำแนกอาจไม่ได้มีความน่าจะเป็นในการเกิดเท่ากัน หากยึดวิธีสร้างต้นไม้ที่ให้จำนวนครั้งในการจำแนกข้อมูลแต่ละประเภทที่ใกล้เคียงกันโดยไม่พิจารณาความน่าจะเป็นในการเกิดข้อมูลแต่ละประเภทประกอบอาจไม่ได้ต้นไม้สำหรับจำแนกที่ให้จำนวนครั้งในการจำแนกเฉลี่ยต่ำที่สุด ซึ่งเป็นการสะสมความผิดพลาดในการจำแนกโดยไม่จำเป็น

บทความนี้จะนำเสนอวิธีการใหม่ในการจำแนกข้อมูลโดยสร้างต้นไม้สำหรับการจำแนกแบบหลายประเภทด้วยเทคนิคการแตกครึ่งตามสารสนเทศโดยการนำความน่าจะเป็นในการเกิดข้อมูลแต่ละประเภทมาใช้ในการค้นหาระนาบที่แบ่งข้อมูล ในทางทฤษฎีวิธีนี้จะสามารถลดจำนวนครั้งในการจำแนกลงเหลือประมาณ $\log k$ ครั้ง ซึ่งเป็นการลดเวลาของการจำแนกเฉลี่ยลงได้โดยเฉพาะอย่างยิ่งกรณีปัญหาการจำแนกที่มีจำนวนประเภทเป็นจำนวนมาก

2. ซัพพอร์ตเวกเตอร์แมชชีนและทฤษฎีสารสนเทศ (Support Vector Machines & Information Theory)

เทคนิคการแตกครึ่งตามสารสนเทศสำหรับซัพพอร์ตเวกเตอร์แมชชีนแบบหลายประเภทมีทฤษฎีที่เกี่ยวข้องดังนี้

2.1 แนวคิดพื้นฐานของซัพพอร์ตเวกเตอร์แมชชีน

แนวคิดหลักในการจำแนกข้อมูล ของซัพพอร์ตเวกเตอร์แมชชีนก็คือ การสร้างระนาบหลายมิติที่ใช้ในการแบ่งข้อมูลออกเป็นสองประเภท

2.1.1 ซัพพอร์ตเวกเตอร์แมชชีนแบบเชิงเส้น (Linear support vector machine)

สมมติมีเซตของข้อมูล D ที่ประกอบด้วยตัวอย่างจำนวน l ตัวในปริภูมิอันดับ n ที่มี 2 ประเภท (+1 และ -1)

$$D=\{(\mathbf{x}_k, y_k) \mid k\in\{1,..,l\}, \mathbf{x}_k\in\Re^n, y_k\in\{+1,-1\}\} \quad (1)$$

ระนาบหลายมิติในปริภูมิอันดับ n ถูกกำหนดโดย $(\mathbf{w},b)$ เมื่อ $\mathbf{w}$ คือเวกเตอร์ในปริภูมิอันดับ n ที่ตั้งฉากกับระนาบหลายมิติและ b คือค่าคงที่ระนาบหลายมิติ $(\mathbf{w}\cdot\mathbf{x})+b$ จะแบ่งข้อมูลได้ก็ต่อเมื่อ

$$\begin{aligned}(\mathbf{w}\cdot\mathbf{x}_i)+b>0 \quad &\text{if} \quad y_i=+1\\ (\mathbf{w}\cdot\mathbf{x}_i)+b<0 \quad &\text{if} \quad y_i=-1.\end{aligned} \quad (2)$$

ต้องการค่า w และ b ที่ทำให้จุดที่อยู่ใกล้ระนาบ
มากที่สุดมีระยะห่าง $1/|w|$ แล้วจะได้

$$(w \cdot x_i) + b \geq 1 \quad \text{if} \quad y_i = +1$$
$$(w \cdot x_i) + b \leq -1 \quad \text{if} \quad y_i = -1 \qquad (3)$$

หรือเขียนได้

$$y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \quad \forall i. \qquad (4)$$

ในการค้นหาระนาบหลายมิติที่ใช้แบ่งข้อมูลดีที่สุดจะต้องค้นหาระนาบหลายมิติที่มีระยะห่างระหว่างข้อมูลที่ใช้สอนกับระนาบหลายมิติที่น้อยที่สุดมีค่ามากที่สุด ระยะห่างระหว่างข้อมูลตัวอย่างสองตัวจากประเภทที่แตกต่างกันมีค่าเท่ากับ

$$d(\mathbf{w}, b) = \min_{\{x_i | y_i = +1\}} \frac{(\mathbf{w} \cdot \mathbf{x}_i) + b}{|\mathbf{w}|} - \max_{\{x_i | y_i = -1\}} \frac{(\mathbf{w} \cdot \mathbf{x}_i) + b}{|\mathbf{w}|}. \qquad (5)$$

จากสมการที่ (3) ค่าที่น้อยที่สุดและมากที่สุดที่เป็นไปได้คือ ± 1 ดังนั้นจำเป็นต้องเพิ่มค่าของฟังก์ชัน

$$d(\mathbf{w}, b) = \frac{1}{|\mathbf{w}|} - \frac{-1}{|\mathbf{w}|} = \frac{2}{|\mathbf{w}|}. \qquad (6)$$

ให้สูงที่สุด ดังนั้นปัญหานี้จึงเท่ากับ

- ลดค่าของ $|w|^2/2$ ให้ต่ำที่สุดโดยขึ้นกับเงื่อนไขต่อไปนี้

  (1) $y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 \quad \forall i.$

สำหรับกรณีที่ไม่สามารถแบ่งข้อมูลได้โดยเงื่อนไขข้างต้น จะต้องปรับเงื่อนไขใหม่โดยเพิ่มพจน์ค่าปรับซึ่งประกอบด้วยผลรวมของความคลาดเคลื่อน ξ_i จากขอบเขตเข้าไปในเงื่อนไข ดังนั้นปัญหาตอนนี้คือ

- ลดค่าของ $\frac{|\mathbf{w}|^2}{2} + C\sum_{i=1}^{l} \xi_i$ ให้ต่ำที่สุด โดยขึ้นกับเงื่อนไขต่อไปนี้

  (1) $y_i[(\mathbf{w} \cdot \mathbf{x}_i) + b] \geq 1 - \xi_i,$

  (2) $\xi_i \geq 0 \quad \forall i.$

พจน์ค่าปรับสำหรับตัวอย่างข้อมูลสอนที่ทำนายผิดพลาดถูกเพิ่มน้ำหนักโดยค่าคงที่ C การเลือกค่า C สูงจะมีผลให้เพิ่มค่าของความคลาดเคลื่อน ξ_i และเพิ่มการคำนวณโดยทำให้การค้นหาหนทาง ที่จะลดจำนวนตัวอย่างข้อมูลสอนที่ทำนายผิดพลาดเพิ่มขึ้น นำลากรองเกรียนมาใช้กับปัญหานี้ สามารถแปลงปัญหาเป็น

- ลดค่าของฟังก์ชันนี้ให้ต่ำที่สุด

  $$L(\mathbf{w}, b, \alpha) = \sum_{i=1}^{l} a_i \quad \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \qquad (7)$$

  โดยขึ้นกับเงื่อนไขต่อไปนี้

  (1) $0 \leq \alpha_i \leq C, \forall i$

  (2) $\sum_{i=1}^{l} \alpha_i y_i = 0$

เมื่อ α_i เรียกว่าตัวคูณลากรอง ตัวอย่างข้อมูลสอนแต่ละตัวจะมีตัวคูณลากรองหนึ่งตัว ตัวอย่างที่มีค่า $\alpha_i > 0$ เรียกว่าซัพพอร์ตเวกเตอร์ และเป็นซัพพอร์ตเวกเตอร์ที่ทำให้ค่าทางขวามือของสมการที่ (4) เท่ากับ 1 ส่วนตัวอย่างอื่นๆที่มี $\alpha_i = 0$ สามารถตัดออกไปจากเซตของตัวอย่างข้อมูลสอนได้ โดยไม่มีผลกระทบใดๆ ต่อผลลัพธ์ของระนาบหลายมิติ

ให้ α^0 ซึ่งเป็นเวกเตอร์ในปริภูมิอันดับ 1 แทนค่าต่ำสุดของ $L(\mathbf{w},b,\alpha)$ ถ้า $\alpha_i^0 > 0$ แล้ว $\mathbf{x}_i$ เป็นซัพพอร์ตเวกเตอร์ของระนาบหลายมิติที่แบ่งแยกดีสุด $(\mathbf{w}^0, b^0)$ สามารถเขียนในเทอมของ α^0 และข้อมูลสอน โดยเฉพาะอย่างยิ่งในเทอมของซัพพอร์ตเวกเตอร์ได้ดังนี้

$$\mathbf{w}^0 = \sum_{i=1}^{l} \alpha_i^0 y_i \mathbf{x}_i = \sum_{\text{support vectors}} \alpha_i^0 y_i \mathbf{x}_i. \qquad (8)$$

$$b^0 = 1 - \mathbf{w}^0 \cdot \mathbf{x}_i \text{ for } \mathbf{x}_i \text{ with } y_i = 1 \text{ and } 0 < \alpha_i < C \qquad (9)$$

ระนาบหลายมิติที่แบ่งแยกดีสุดจะจำแนกจุดต่างๆ ตามเครื่องหมายของผลลัพธ์ของฟังก์ชัน $f(x)$

$$f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^0 \cdot \mathbf{x} + b^0\right)$$
$$= \text{sign}\left(\sum_{\text{support vectors}} \alpha_i^0 y_i (\mathbf{x}_i \cdot \mathbf{x}) + b^0\right).$$

ซัพพอร์ตเวกเตอร์ $\mathbf{x}_i$ ที่มี $\alpha_i^0 = C$ อาจจะถูกจำแนกผิดพลาดหรือไม่ก็ได้ แต่ $\mathbf{x}_i$ ตัวอื่นๆ จะจำแนกได้อย่างถูกต้อง

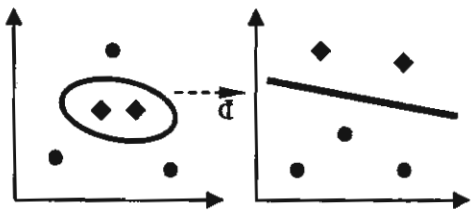
2.1.2 ซัพพอร์ตเวกเตอร์แมชชีนแบบไม่เชิงเส้น (Non-linear Support vector machine)

อัลกอริทึมที่ได้กล่าวมาแล้วใช้ได้กับข้อมูลที่สามารถแบ่งด้วยระนาบหลายมิติแบบเชิงเส้นเท่านั้น แต่กับข้อมูลที่ไม่อาจแบ่งแบบเชิงเส้นได้ ซัพพอร์ตเวกเตอร์แมชชีนแก้ปัญหานี้โดยแมปข้อมูลตัวอย่าง ไปยังปริภูมิอันดับสูง โดยเลือกใช้ฟังก์ชันการแมปที่ไม่เป็นเชิงเส้น นั่นคือ เราเลือกฟังก์ชันในการแมป $\Phi:\Re^n \mapsto \mathrm{H}$ เมื่อปริภูมิของ H มีอันดับสูงกว่าปริภูมิอันดับ n เราสามารถค้นหาระนาบที่แบ่งแยกดีสุดในปริภูมิอันดับสูงที่เทียบเท่ากับการแยกที่ไม่เป็นเชิงเส้นใน $\Re^n$

ตัวอย่างข้อมูลสอนที่ใช้ในสมการที่ (7) อยู่ในรูปของผลคูณเชิงสเกลาร์ระหว่างเวกเตอร์ ดังนั้นในปริภูมิอันดับสูงเราสามารถจัดการกับข้อมูลในรูปของ $\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$. ถ้าปริภูมิของ H มีอันดับสูงจะทำให้ยุ่งยาก หรือใช้การคำนวณมาก อย่างไรก็ตามถ้าเรามีฟังก์ชันเคอร์เนลที่สามารถคำนวณ $k(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$ แล้วเราสามารถใช้ฟังก์ชันนี้แทนที่ $\mathbf{x}_i\cdot\mathbf{x}_j$ ทุกๆที่ในการคำนวณ และไม่จำเป็นต้องรู้ฟังก์ชันที่ใช้แมป Φ จริงๆ ฟังก์ชันเคอร์เนลที่นิยมใช้ได้แก่

Polynomial degree d: $k(\mathbf{x},\mathbf{y})=\left|\mathbf{x}\cdot\mathbf{y}+1\right|^d$ (11)

Radial basis function: $k(\mathbf{x},\mathbf{y})= e^{-\|\mathbf{x}-\mathbf{y}\|^2/c}$ (12)



รูปที่1. แนวคิดการแมปข้อมูลไม่เชิงเส้นไปสู่ปริภูมิอันดับสูง

2.1.3 ประสิทธิภาพโดยนัยทั่วไปของซัพพอร์ต เวกเตอร์แมชชีน

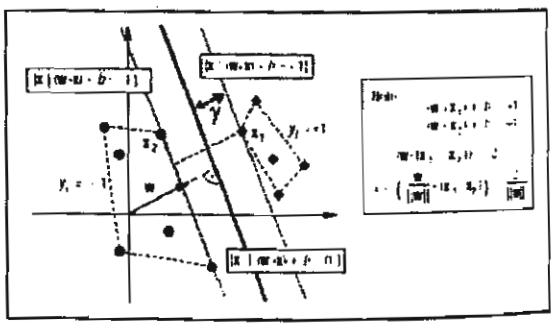
เราสามารถกำหนดขอบเขตความผิดพลาดของซัพพอร์ตเวกเตอร์ได้โดยกำหนดชนิดของฟังก์ชันเป็นค่าตัวเลขจำนวนจริงภายในลูกบอลที่มีรัศมี R ค่ารัศมีไม่เกิน $\Re^n$ เป็นฟังก์ชันดังนี้ $F=\{\mathbf{x}\mapsto \mathbf{w}\cdot\mathbf{x}:\|\mathbf{w}\|\le 1,\|\mathbf{x}\|\le R\}$ จะมีค่าคงที่ c ที่สำหรับการกระจายทุกประเภท ด้วยความน่าจะเป็นอย่างน้อย 1-δ บนตัวอย่าง z ที่เลือกมาอย่างอิสระจากกัน m ตัว ถ้าตัวจำแนก $h=\mathrm{sgn}(f)\in\mathrm{sgn}(F)$ มีค่าระยะห่างระหว่างระนาบหลายมิติ กับข้อมูลที่ใช้สอนอย่างน้อย γ (ดูรูปที่ 2) สำหรับตัวอย่างทุกตัวใน z แล้ว ความผิดพลาดของตัวจำแนก h จะไม่เกินค่า

$$\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m+\log\left(\frac{1}{\delta}\right)\right). \qquad (13)$$

นอกเหนือจากนี้แล้วด้วยความน่าจะเป็นอย่างน้อย 1-δ ทุกตัวจำแนก $h\in\mathrm{sgn}(F)$ จะมีความผิดพลาดไม่เกิน

$$\frac{k}{m}+\sqrt{\frac{c}{m}\left(\frac{R^2}{\gamma^2}\log^2 m+\log\left(\frac{1}{\delta}\right)\right)} \qquad (14)$$

เมื่อ k คือจำนวนตัวอย่างข้อมูลใน z ที่มีระยะห่างระหว่างระนาบหลายมิติกับซัพพอร์ตเวกเตอร์อย่างน้อย γ



รูปที่ 2. ระยะห่างระหว่างระนาบหลายมิติกับข้อมูลที่ใช้สอน

ทฤษฎีสารสนเทศ (Information Theory)

สารสนเทศของข้อมูลขึ้นอยู่กับความน่าจะเป็นของ ซึ่งสามารถวัดอยู่ในรูปของบิตจากสูตร

$$\text{สารสนเทศของข้อมูล} = -\log_2 P \qquad (15)$$

เมื่อ P คือความน่าจะเป็นของข้อมูล

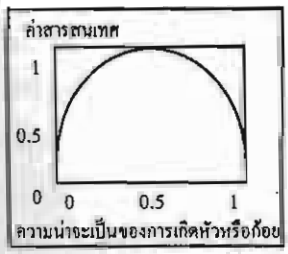
ให้ชุดของข้อมูล M ประกอบด้วยค่าที่เป็นไปได้คือ $(m_1, m_2, \ldots, m_n)$ และให้ความน่าจะเป็นที่จะเกิดค่า m_i มีค่าเท่ากับ $P(m_i)$ จะได้ว่าค่าสารสนเทศของ M หรือค่าเอนโทรปีของ M เขียนแทนด้วย $I(M)$ คำนวณได้จากสูตร

$$I(M) = \sum_{i=1}^{n} -P(m_i)\log_2 P(m_i) \qquad (16)$$

ความหมายของค่าสารสนเทศกรณีมีค่าน้อย หมายถึง ข้อมูลในชุดนั้นมีความแตกต่างกันน้อย หรือเกือบจะเป็นประเภทเดียวกัน แต่กรณีค่าสารสนเทศสูงจะบ่งบอกว่าข้อมูลในชุดนั้นมีความแตกต่างกันมากหรือประกอบด้วยตัวอย่างหลายประเภทที่มีจำนวนใกล้เคียงกัน ตัวอย่างเช่น การโยนหัวโยนก้อย ชุดข้อมูล M จะประกอบด้วยค่าที่เป็นไปได้ {หัว, ก้อย} จะได้ว่าค่าสารสนเทศของการโยนหัวก้อยเท่ากับ

$$-P(\text{หัว})\log_2(P(\text{หัว})) - P(\text{ก้อย})\log_2(P(\text{ก้อย}))$$

เมื่อพิจารณากรณีที่โยนออกหัวหมดหรือก้อยหมด ค่าสารสนเทศจะเป็น 0 และค่าสารสนเทศจะค่อยๆ เพิ่มขึ้นจนสูงที่สุด เมื่อความน่าจะเป็นของการเกิดหัวเท่ากับความน่าจะเป็นของการเกิดก้อยหรือสามารถแสดงได้ดังรูปที่ 3



รูปที่ 3. ความสัมพันธ์ระหว่างความน่าจะเป็นและค่าสารสนเทศ

3. งานวิจัยที่เกี่ยวข้อง

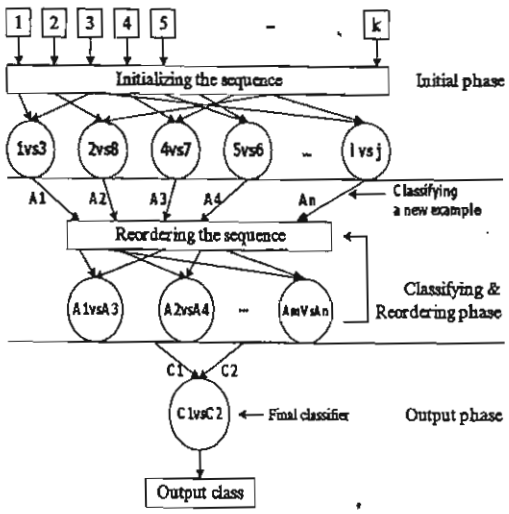
3.1 วิธีแมกซ์วิน (Max Wins)

อัลกอริทึมแมกซ์วินจะทำงานโดยอาศัยฟังก์ชันพื้นฐานของการจำแนกแบบหนึ่งต่อหนึ่ง สร้างตัวจำแนกข้อมูลทั้งหมด จำนวน $k(k\text{-}1)/2$ ตัว โดยที่ k คือจำนวนประเภทข้อมูลทั้งหมด ตัวจำแนกแต่ละตัวจะถูกสอนโดยข้อมูลจาก 2 ประเภท i และ j ใดๆ ตามเงื่อนไขดังนี้

$$\min_{w^{ij}, b^{ij}, \xi^{ij}} \quad \frac{1}{2}(w^{ij})^T w^{ij} + C\sum_t \xi_t^{ij}(w^{ij})^T$$
$$(w^{ij})^T \Phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij}, \quad \text{if } y_t = i$$
$$(w^{ij})^T \Phi(x_t) + b^{ij} \leq -1 + \xi_t^{ij}, \quad \text{if } y_t = j \; \xi_t^{ij} \geq 0. \qquad (17)$$

แล้วใช้ฟังก์ชัน $sign((w^{ij})^T\Phi(x) + b^{ij})$ จำแนกให้คำตอบว่าตัวอย่าง x ใดๆ อยู่ในประเภท i หรือ j หลังจากได้คำตอบทั้งหมดแล้ว วิธีแมกซ์วินจะนำคำตอบที่ได้จากตัวจำแนกทั้ง $k(k\text{-}1)/2$ ตัว มาโหวตรวมกันโดยประเภทที่ได้ผลโหวตสูงที่สุด ก็จะเป็นคำตอบของแมกซ์วิน ส่วนในกรณีที่มีผลโหวตสูงสุดมากกว่า 1 ประเภท ก็จะทำการสุ่มคำตอบจากประเภทเหล่านั้น

3.2 วิธีอาร์เอดีเอจี (Reordering Adaptive Directed Acyclic Graph (RADAG))



รูปที่ 4. โครงสร้างการจำแนกข้อมูลของอาร์เอดีเอจี

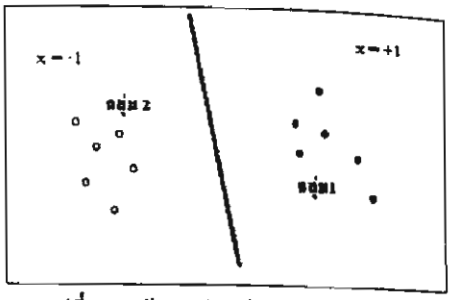
Phetkaew, et al. [8] นำเสนอวิธีการใหม่ เรียกว่า อาร์เอดีเอจี เพื่อปรับปรุงความถูกต้องของวิธีเอดีเอจีเดิม [9] โดยในโครงสร้างของการจำแนกนั้นจะเป็นรูปคล้ายสามเหลี่ยมคว่ำเหมือนวิธีเอดีเอจี แต่จะมีการเลือกลำดับที่ทำให้เกิดความผิดพลาดน้อยที่สุดของการจำแนกในแต่ละชั้นโดยวิธีการเลือกจะใช้อัลกอริทึมของการจับคู่สมบูรณ์แบบน้ำหนักน้อยสุด (Minimum-Weight Perfect Matching) [4] เพื่อจับคู่ให้ค่าขอบเขตความผิดพลาดรวมในแต่ละชั้นมีค่าน้อยที่สุด โดยค่าขอบเขตความผิดพลาดของฟังก์ชันแบบสองประเภทแต่ละตัวหาได้จากค่าประสิทธิภาพโดยนัยทั่วไปของซัพพอร์ตเวกเตอร์แมชชีน (Generalization Performance of Support Vector Machine) [2] และเมื่อมีการจำแนกผ่านไปสู่ชั้นใหม่ก็จะมีการเรียงลำดับการจำแนกใหม่ (Reordering) อีกจนเหลือประเภทเดียวในชั้นสุดท้ายซึ่งเป็นคำตอบของอาร์เอดีเอจี

3.3 วิธีจำแนกข้อมูลแบบแตกครึ่งแบบสมดุล (Balanced Dichotomization Classification)

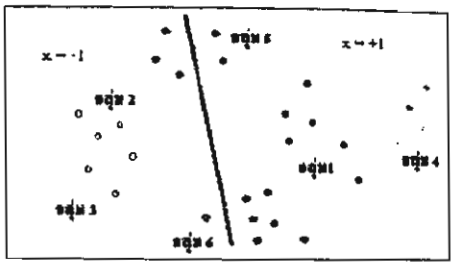
Kijsirikul & Boonsirisumpun [7] เสนอวิธีจำแนกข้อมูลแบบแตกครึ่งแบบสมดุล (Balanced Dichotomization) โดยมีแนวคิดในการจำแนกข้อมูลด้วยระนาบหลายมิติที่อยู่ในตำแหน่งกึ่งกลางที่สุดของข้อมูลมาจำแนก เพื่อลดจำนวนครั้งของการจำแนกลงโดยมีขั้นตอนในการจำแนกข้อมูลดังนี้

3.3.1 การค้นหาตำแหน่งของระนาบหลายมิติ

ในการสร้างฟังก์ชันของระนาบหลายมิติในการจำแนกแบบสองประเภทตามปกตินั้น เราจะได้ตำแหน่งของระนาบอยู่กึ่งกลางระหว่างสองประเภทข้อมูลที่เราใช้สอน โดยที่ตำแหน่งของข้อมูลทั้งสองประเภทจะอยู่คนละด้านของระนาบตัว อย่างเช่น ถ้าสร้างระนาบระหว่าง ประเภท 1 กับ ประเภท 2 ก็จะได้ระนาบหลายมิติอยู่กึ่งกลางระหว่าง 1 กับ 2 และได้ตำแหน่งของประเภท 1 อยู่ในด้าน $x = +1$ และตำแหน่งของประเภท 2 อยู่ในด้าน $x=-1$ เป็นต้น (ดูรูปที่ 5 (ก)) แต่จะไม่รู้ตำแหน่งเทียบกับประเภทข้อมูลอื่น ทำให้การจำแนกข้อมูล 1 ครั้งตัดประเภทที่ไม่ถูกต้องออกไปได้เพียง 1 ประเภท ในวิธีนี้จึงมีการค้นหาตำแหน่งของระนาบเทียบกับประเภทข้อมูลอื่นเพิ่มโดยนำฟังก์ชันของระนาบแบบสองประเภทที่มีอยู่แล้วไปคำนวณหาตำแหน่งของประเภทข้อมูลอื่นเพิ่มเติมก็จะได้ตำแหน่งของระนาบเมื่อเทียบกับข้อมูลทั้งหมด (ดูรูปที่ 5 (ข))



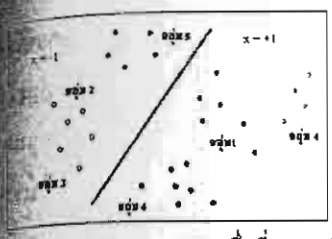
รูปที่ 5 (ก) ข้อมูลเรียนรู้ของระนาบตามปกติ



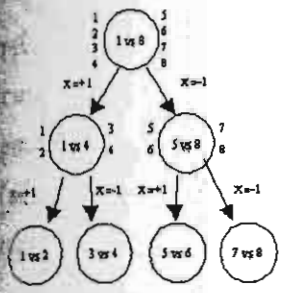
รูปที่ 5 (ข) ข้อมูลเรียนรู้เมื่อมีการค้นหาตำแหน่งเทียบกับประเภทข้อมูลอื่นเพิ่ม

3.3.2 การค้นหาระนาบแตกครึ่งแบบสมดุล

เมื่อทำการคำนวณเพื่อค้นหาตำแหน่งของระนาบหลายมิติเทียบกับประเภทข้อมูลอื่นตามขั้นตอนข้างต้นแล้ว ในการจำแนกข้อมูลแบบแตกครึ่งแบบสมดุลจะทำการค้นหาระนาบที่สามารถแบ่งข้อมูลในตำแหน่งสมดุลที่สุดของข้อมูลทั้งหมดมาจำแนกซึ่งจะเลือกระนาบที่แบ่งข้อมูลในแต่ละด้านของระนาบมีจำนวนประเภทที่ใกล้เคียงกัน เพื่อให้ในการจำแนกข้อมูลแต่ละครั้งสามารถตัดประเภทข้อมูลที่ไม่ถูกต้องออกไปได้มากที่สุด นั่นคือครึ่งหนึ่งของประเภทข้อมูลทั้งหมดในรอบนั้นนั่นเอง



รูปที่ 6 ตัวอย่างระนาบแตกครึ่งที่สมดุลที่สุด



รูปที่ 7 การจำแนกแบบแตกครึ่งแบบสมดุลที่ดีที่สุดสำหรับปัญหา 8 ประเภท

โครงสร้างของการจำแนกโดยใช้ระนาบแตกครึ่งที่สมดุลที่สุดนั้น ถ้าสามารถหาระนาบที่แบ่งข้อมูลเหลือครึ่งหนึ่งได้ทุกครั้งจะทำให้จำนวนครั้งของการจำแนกที่ดีที่สุดลดลงเหลือ $\log_2 k$ ครั้งในปัญหา k ประเภท ดังตัวอย่างในรูปที่ 7

ข้อจำกัดของการควบคุมความผิดพลาด วิธีการแตกครึ่งแบบสมดุลมีการกำหนดขอบเขตความผิดพลาดของการจำแนกข้อมูล โดยการกำหนดให้ระนาบหลายมิติที่จะนำมาหาระนาบสมดุลที่สุดต้องมีขอบเขตความผิดพลาดของช่วงค่าประสิทธิภาพโดยทั่วไปอยู่ในช่วงที่กำหนดในแต่ละชั้น เพื่อควบคุมค่าความถูกต้อง

4. เทคนิคการจำแนกข้อมูลแบบแตกครึ่งตามสารสนเทศ

ในหัวข้อนี้จะนำเสนอแนวคิดที่ทำให้การจำแนกข้อมูลแต่ละครั้งสามารถตัดประเภทข้อมูลที่ไม่ถูกต้องออกไปได้มากที่สุด โดยพิจารณาความน่าจะเป็นในการเกิดข้อมูลแต่ละประเภทประกอบเพื่อให้ได้จำนวนครั้งเฉลี่ยของการจำแนกลดลง

กรณีตัวอย่างจากปัญหาการเข้ารหัสข้อมูลของอักขระ 4 ตัว คือ A, B, C และ D โดยสมมติให้ความน่าจะเป็นของการเกิดข้อมูลแต่ละประเภทมีเท่ากัน จะได้ว่าจำนวนบิตที่น้อยที่สุดในการแทนอักขระเหล่านี้คือ 2 บิต โดยแทนแต่ละอักขระดังนี้

A	แทนด้วย	00
B	แทนด้วย	01
C	แทนด้วย	10
D	แทนด้วย	11

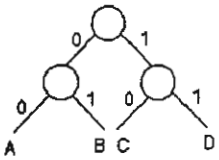
ตัวอย่าง หากต้องการเข้ารหัสข้อมูลของ ABACADAB จะได้ 0001001000110001 จำนวนบิตที่ใช้ คือ 16 เมื่อพิจารณาข้อมูลจากตัวอย่าง ตลอดจนข้อมูลที่เป็นข้อความทั่วไปในชีวิตประจำวันจะมีความน่าจะเป็นในการเกิดข้อมูลของแต่ละอักขระไม่เท่ากัน อาทิ อักขระ A, E, I, O และ U ซึ่งเป็นสระในภาษาอังกฤษจะมีความน่าจะเป็นในการเกิดมากกว่าอักขระอื่นๆ

จากตัวอย่างเดิมสมมติว่าความน่าจะเป็นในการเกิดแต่ละอักขระเป็นดังนี้

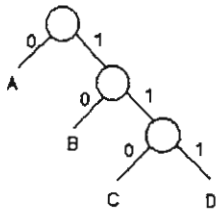
A	มีความน่าจะเป็นคือ 1/2 แทนรหัสเดิมด้วย	0
B	มีความน่าจะเป็นคือ 1/4 แทนรหัสเดิมด้วย	10
C	มีความน่าจะเป็นคือ 1/8 แทนรหัสเดิมด้วย	110
D	มีความน่าจะเป็นคือ 1/8 แทนรหัสเดิมด้วย	111

หากต้องการเข้ารหัสข้อมูลชุดเดิมคือ ABACADAB จะได้ 01001100111010 จำนวนบิตที่ใช้ คือ 14 จะเห็นว่าเมื่อนำความน่าจะเป็นในการเกิดของแต่ละอักขระ มาพิจารณาด้วย จะทำให้จำนวนบิตเฉลี่ยในการเข้ารหัสข้อมูลลดลง โดยอาศัยหลักพื้นฐานที่ว่าหากอักขระใดใช้บ่อย ก็ให้แทนด้วยจำนวนบิตน้อยๆ ส่วนอักขระใดไม่ค่อยใช้ก็ยอมให้แทนด้วยจำนวนบิตที่ยาวกว่าได้

เมื่อพิจารณาในกรณีแรกหากความน่าจะเป็นในการเกิดข้อมูลของแต่ละอักขระมีค่าเท่ากันก็เหมาะสมแล้วที่จะแทนอักขระแต่ละตัวด้วยรหัสไบนารีจำนวน 2 บิต เพื่อให้ง่ายแก่การเข้าใจจึงแสดงวิธีการแทนอักขระกรณีที่ความ น่าจะเป็น ในการเกิดข้อมูลของแต่ละอักขระเท่ากันและไม่เท่ากันด้วยรูปที่ 8 (ก) และ (ข) ตามลำดับ



รูปที่ 8 (ก) การแทนอักขระด้วยต้นไม้แบบไบนารี กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระเท่ากัน



รูปที่ 8 (ข) การแทนอักขระด้วยต้นไม้แบบไบนารี กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระไม่เท่ากัน

จากรูปที่ 8 (ก) และ (ข) ซึ่งแสดงการแทนอักขระด้วยต้นไม้แบบไบนารีจะเห็นว่าในรูปที่ 8 (ก) กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระเท่ากัน ต้นไม้แบบไบนารีที่สมดุลจะให้จำนวนเส้นเชื่อม (edge) จากรากไปถึงแต่ละใบนั้นเท่ากัน แต่ในรูปที่ 8 (ข) กรณีที่ความน่าจะเป็นในการเกิดของแต่ละอักขระไม่เท่ากันจะให้จำนวนเส้นเชื่อมจากรากไปถึงใบแต่ละใบด้วยจำนวนเส้นเชื่อมมากน้อยต่างกันขึ้นกับความน่าจะเป็นในการเกิดของอักขระนั้น เช่น อักขระ A มีความน่าจะเป็นสูงสุดจะให้จำนวนเส้นเชื่อมจากรากไปถึงใบต่ำที่สุด ส่วนอักขระ C และ D จะมีจำนวนเส้นเชื่อมจากรากไปถึงใบมากที่สุดเนื่องจากมีความน่าจะเป็นในการเกิดต่ำสุด

หากพิจารณาในแง่ของการค้นหาข้อมูลของรูปที่ 8 (ข) อักขระ A เกิดบ่อย จึงแทนด้วยจำนวนบิตต่ำ หรือกล่าวในเชิงการค้นหาข้อมูลคือ จำนวนครั้งในการค้นหาต่ำ ส่วนอักขระ C และ D มีโอกาสเกิดน้อย จึงแทนด้วยจำนวนบิตที่ยาวได้ หรือจำนวนครั้งในการค้นหาสูงได้ เมื่อพิจารณาการค้นหาโดยรวมจึงทำให้เวลาการค้นหาเฉลี่ยต่ำสุด

จากข้างต้นการสร้างต้นไม้สำหรับการจำแนกแบบหลายประเภทเทียบได้กับการเข้ารหัสข้อมูลโดยแต่ละโนดจะเป็นตัวจำแนกแบบสองประเภทที่เลือกมา ซึ่งจำนวนครั้งในการจำแนกข้อมูลเฉลี่ยย่อมขึ้นกับความน่าจะเป็นของการเกิดข้อมูลในแต่ละประเภทเช่นเดียวกับความน่าจะเป็นของการเกิดอักขระ

บทความนี้ได้เสนอวิธีการใหม่เรียกว่า เทคนิคการจำแนกข้อมูลแบบแตกครึ่งตามสารสนเทศ ซึ่งมีแนวคิดในการจำแนกข้อมูลด้วยระนาบหลายมิติที่แบ่งข้อมูลออกเป็นสองส่วนได้ดีที่สุดโดยพิจารณาจากความน่าจะเป็นในการเกิดข้อมูลแต่ละประเภท ซึ่งมีขั้นตอนในการจำแนกข้อมูลดังนี้

4.1 การค้นหาตำแหน่งของระนาบหลายมิติ

การสร้างฟังก์ชันของระนาบหลายมิติในการจำแนกแบบสองประเภทตามปกตินั้นจะได้ตำแหน่งของระนาบอยู่กึ่งกลางระหว่างข้อมูลสองประเภทที่เราใช้สอน โดยที่ตำแหน่งของข้อมูลทั้งสองประเภทจะอยู่คนละด้านของระนาบแต่จะไม่รู้ตำแหน่งเทียบกับประเภทข้อมูลอื่น จึงมีการค้นหาตำแหน่งของระนาบเทียบกับประเภทข้อมูลอื่นเพิ่มโดยนำฟังก์ชันของระนาบแบบสองประเภทที่มีอยู่แล้ว ไปคำนวณหาตำแหน่งของประเภทข้อมูลอื่นเพิ่มเติมก็จะได้ตำแหน่งของระนาบ เมื่อเทียบกับข้อมูลทั้งหมด เช่นเดียวกับในวิธีแตกครึ่งแบบสมดุล

4.2 การค้นหาระนาบแตกครึ่งตามสารสนเทศ

เมื่อทำการคำนวณเพื่อค้นหาตำแหน่งของระนาบหลายมิติเทียบกับประเภทข้อมูลอื่นตามขั้นตอนข้างต้นแล้วจะทำการค้นหาระนาบที่สามารถแบ่งข้อมูลได้ดีที่สุดโดยพิจารณาจากค่าเอนโทรปีของข้อมูลโดยเลือกระนาบที่ให้ค่าเอนโทรปีต่ำสุดซึ่งจะเป็นตัวบ่งชี้ว่าหากนำระนาบนั้นไปใช้ในการจำแนกข้อมูลแล้ว ข้อมูลที่ได้ในแต่ละด้านของระนาบจะมีจำนวนประเภทที่ปนกันอยู่น้อยที่สุดตามหลักของทฤษฎีสารสนเทศที่กล่าวข้างต้น เมื่อแต่ละด้านมีจำนวนประเภทของข้อมูลปนกันอยู่น้อยจึงสามารถตัดประเภทที่ไม่ใช่ออกไปได้ดีด้วย

ใช้ในการคำนวณค่าเอนโทรปีเป็นดังนี้

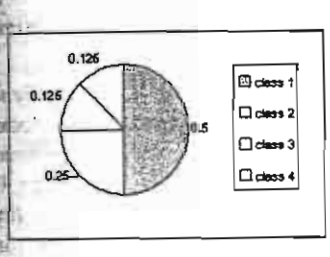
เอนโทรปีของตัวจำแนก $= \sum_{i=1}^{n} \frac{|t_i|}{|T|} I(t_i)$

$$\sum_{i=1}^{k} -P(m_i)\log_2 P(m_i) \qquad (18)$$

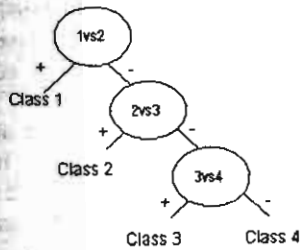
n เป็นจำนวนกิ่งของต้นไม้ในที่นี้จะมีค่าเป็น 2 ซึ่งได้แก่ กิ่งบวกและกิ่งลบ

k เป็นจำนวนประเภทของข้อมูลทั้งหมด

ตัวอย่าง กำหนดให้ชุดข้อมูลหนึ่งมีจำนวนประเภท(Class) 4 ประเภทและมีความน่าจะเป็นในการเกิดข้อมูลของแต่ละประเภทเป็นดังรูปที่ 9 (ก) และต้นไม้สำหรับการจำแนกหลายประเภทที่สร้างจากค่าเอนโทรปีของข้อมูลชุดตัวอย่างที่ดีที่สุดแสดงดังรูปที่ 9 (ข)



รูปที่ 9 (ก) ความน่าจะเป็นในการเกิดข้อมูลแต่ละประเภท



รูปที่ 9 (ข) ต้นไม้สำหรับการจำแนกแบบหลายประเภท

การคำนวณค่าจำนวนครั้งในการจำแนกที่คาดหวังสามารถคำนวณได้จากสูตร

$$-P(m_i)\log_2 P(m_i) \qquad (19)$$

สำหรับ ปัญหาการจำแนก k ประเภท และ m_i คือความน่าจะเป็นในการเกิดข้อมูลของประเภทที่ m_i

กรณีตัวอย่างจากรูปที่ 9 สามารถคำนวณค่าจำนวนครั้งในการจำแนกที่คาดหวังได้ดังนี้

$$\sum_{i=1}^{4} -P(m_i)\log_2 P(m_i)$$

$$=[-(0.5)*\log_2(0.5)]+[-(0.25)*\log_2(0.25)]+$$
$$[-(0.125)*\log_2(0.125)]+[-(0.125)*\log_2(0.125)]$$
$$= 1.75$$

5. ผลการทดลอง

สำหรับการทดลองได้ใช้ข้อมูลทดสอบจาก UCI Machine Learning Repository [3] จำนวน 4 ชุด ดังตารางที่ 1 ข้อมูลแต่ละชุดจะประกอบด้วย ข้อมูลสอนและข้อมูลทดสอบ ในขั้นตอนของการทดลองจะทำการแบ่งชุดข้อมูลสอนออกเป็น ข้อมูลสอนจริงและข้อมูลทดสอบความถูกต้อง(Validation data) เพื่อใช้ในการหาค่าพารามิเตอร์ที่เหมาะสม [7] ซึ่งประกอบด้วยค่า P คือค่าร้อยละในการตัดเล็ม ($0 \le P \le 10$) และ R คือ ค่าขอบเขตความผิดพลาด($x_{min} \le R \le x_{mean+sd}$ เมื่อ x_{min} คือ ค่าต่ำสุดของขอบเขตความผิดพลาดของตัวจำแนกทั้งหมด และ $x_{mean+sd}$ คือผลรวมของค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานของขอบเขตความผิดพลาดของตัวจำแนกทั้งหมด) หลังจากนั้นจึงใช้ค่า P และ R ที่ได้เพื่อสร้างตัวจำแนกแบบหลายประเภทจากชุดข้อมูลสอนเดิมและใช้ชุดข้อมูลทดสอบเพื่อหาค่าความถูกต้อง และค่าจำนวนครั้งเฉลี่ยในการจำแนกข้อมูล ผลการทดลองแสดง ดังตารางที่ 2-3 (IBD คือ Information-Baesd dichotomization, BD คือ Balanced Dichotomization, Expected Value คือ ค่าจำนวนครั้งในการจำแนกที่คาดหวัง, ตัวอักษร c, d คือพารามิเตอร์ของฟังก์ชันเคอร์เนล และตัวอักษรเข้มในตารางแสดงค่าที่ดีที่สุดในแต่ละชุดข้อมูล)

ตารางที่ 1 : ลักษณะของข้อมูลที่นำมาทดสอบ

Dataset	#training set	#test set	#class	#feature
Satimage	4,435	2,000	6	36
Shuttle	43,500	14,500	7	9
Vowel	528	462	11	10
Soybean	290	340	15	35

ตารางที่ 2: เปรียบเทียบจำนวนครั้งของการจำแนก

Dataset	Polynomial Kernel					
	Expected Value of IBD	Output of IBD	Expected Value of BD ($\log_2 k$)	Output of BD	RADAG (k-1)	Maxwins (k(k-1)/2)
Satimage	2.474	4.999	2.585	4.847	5	15
Shuttle	0.965	5.359	2.807	5.378	6	21
Vowel	3.459	3.385	3.459	5.665	10	55
Soybean	3.617	6.971	3.907	6.783	14	105
Dataset	RBF Kernel					
	Expected Value of IBD	Output of IBD	Expected Value of BD ($\log_2 k$)	Output of BD	RADAG (k-1)	Maxwins (k(k-1)/2)
Satimage	2.474	4.734	2.585	4.577	5	15
Shuttle	0.965	4.982	2.807	5.758	6	21
Vowel	3.459	5.628	3.459	5.819	10	55
Soybean	3.617	5.400	3.907	6.591	14	105

ตารางที่ 3: เปรียบเทียบค่าความถูกต้อง

Dataset	Polynomial Kernel							
	d	IBD	d	BD	d	RADAG	d	Max Wins
Satimage	6	88.850	6	87.840	6	88.900	6	88.453
Shuttle	8	99.924	8	99.924	8	99.924	8	99.924
Vowel	3	64.935	2	65.022	3	64.502	3	64.329
Soybean	3	90.882	4	89.529	3	91.176	3	90.471
Dataset	RBF Kernel							
	c	IBD	c	BD	c	RADAG	c	Max Wins
Satimage	0.5	91.950	1.0	91.350	0.5	91.950	0.5	91.984
Shuttle	3.0	99.890	3.0	99.886	3.0	99.897	3.0	99.897
Vowel	0.3	66.450	0.2	62.900	0.2	67.100	0.2	65.340
Soybean	0.07	91.471	0.04	89.118	0.07	90.882	0.08	90.468

จากผลการทดลองการแตกครึ่งตามสารสนเทศให้ค่าความถูกต้องใกล้เคียงกับวิธีอื่น ส่วนจำนวนครั้งในการจำแนกเมื่อเปรียบเทียบกับอาร์เอดีเอจีและแมกซ์วิน พบว่าการแตกครึ่งตามสารสนเทศให้ค่าจำนวนครั้งในการจำแนกต่ำกว่าทุกกรณี และเมื่อเปรียบเทียบกับการแตกครึ่งแบบสมดุล พบว่าการแตกครึ่งตามสารสนเทศให้ค่าจำนวนครั้งในการจำแนกต่ำกว่าถึง 5 ใน 8 กรณี ส่วนอีก 3 ใน 8 กรณี แม้การแตกครึ่งตามสารสนเทศจะให้จำนวนครั้งในการจำแนกที่สูงกว่าแต่ก็ให้ค่าความถูกต้องที่สูงกว่าด้วย

ความสามารถในการลดจำนวนครั้งของการจำแนกจะขึ้นกับระนาบของตัวจำแนกแบบสองประเภทที่สร้างได้จากค่า P และ R ที่ดีที่สุดด้วย โดยหากระนาบที่ได้สามารถแบ่งประเภทของข้อมูลได้ดีนั่นคือ แต่ละด้านของระนาบมีประเภทของข้อมูลปนกันน้อยหรือถ้าหากด้านใดด้านหนึ่งของระนาบมีเพียงประเภทเดียว ค่าจำนวนครั้งเฉลี่ยของการจำแนกที่ได้จริงยิ่งให้ค่าใกล้เคียงกับค่าจำนวนครั้งในการจำแนกที่คาดหวังจากสูตร $\sum_{i=1}^{k} -P(m_i)\log_2 P(m_i)$ ในทางตรงกันข้ามหากระนาบที่ได้ระนาบใดที่แบ่งประเภทข้อมูลได้ดีพอก็ยิ่งส่งผลให้ตัวจำแนกหลายประเภทที่ได้มีจำนวนครั้งในการจำแนกเฉลี่ยสูงกว่าคาดคลื่อนจากค่าจำนวนครั้งในการจำแนกที่คาดหวังมากขึ้นเท่านั้น

6. สรุป

เทคนิคการแตกครึ่งตามสารสนเทศสามารถเพิ่มประสิทธิภาพการจำแนกประเภทของซัพพอร์ตเวกเตอร์แมชชีนแบบหลายประเภทด้านจำนวนครั้งของการจำแนก โดยผลดังกล่าวจะยิ่งปรากฏชัดในกรณีปัญหาที่มีจำนวนประเภทเป็นจำนวนมาก อีกทั้งยังคงให้ค่าความถูกต้องใกล้เคียงกับตัวจำแนกแบบหลายประเภทวิธีอื่น

7. เอกสารอ้างอิง

[1] บุญเสริม กิจศิริกุล., *เอกสารประกอบการเรียนวิชาปัญญาประดิษฐ์* ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2546

[2] Bartlett, P. L. and Shawe-Taylor, J., *Generalization performance of support vector machines and other pattern classifiers*, in Advances in Kernel Methods Support Vector Learning, pp. 43-54, MIT Press, Cambridge, USA, 1999

[3] Blake, C., Keogh, E., and Merz, C, *UCI Repository of Machine Learning Databases*, Department of Information and Computer Science, University of California, Irvine, 1998. http://www.ics.uci.edu/~mlearn/MLSummary. html

[4] Cook, W. and Rohe, A., *Computing Minimum-Weight Perfect Matchings*, Technical Report 97863, Forschungsinstitut für Diskrete Mathematik, Universität Bonn, 1997.

[5] Friedman, J. H., *Another Approach to Polychotomous classification*, Technical report, Department of Statistics Stanford University, 1996.

[6] Hsu , C. and Lin, C., *A Comparison of Methods for Multiclass Support Vector Machines*, IEEE Transactions on Neural Networks, 13, 415-425, March, 2002.

[7] Kijsirikul, B., Boonsirisumpun, N. and Limpiyakorn, Y., *Multiclass Support Vector Machines Using Balanced Dichotomization*, The 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI-2004), 2004.

[8] Phetkaew, T., Kijsirikul, B. and Rivepiboon, W., *Reordering Adaptive Directed Acyclic Graphs: An Improved Algorithm for Multiclass Support Vector Machines*, The 2003 IEEE/INNS International Joint Conference on Neural Networks, Portland, Oregon, July 20-24, 2003.

[9] Ussivakul, N. and Kijsirikul, B., *Adaptive DAG: Another Approach for Multiclass Classification*, International Conference on Intelligent Technologies, 2001.