บทที่ 3

ทฤษฎีที่ใช้ในการวิจัย

การเรียนรู้ของเครื่อง (Machine Learning) [20]

การเรียนรู้ของมนุษย์มีหลายแบบตั้งแต่การเรียนรู้จากคำบอกเล่า (learning by being told) จนถึงการเรียนรู้จากการค้นพบ (learning by discovery) การเรียนรู้จากคำบอกเล่าผู้สอนจะบอกความรู้ทั้งหมดให้ผู้เรียน เปรียบได้กับการเขียนโปรแกรมคอมพิวเตอร์ ซึ่งจะต้องบอกขั้นตอนการทำงานของโปรแกรมให้กับคอมพิวเตอร์ ส่วนการเรียนรู้จากการค้นพบผู้เรียนจะค้นพบความรู้ใหม่จากการสังเกตสิ่งแวดล้อม หรือความสัมพันธ์ของคุณสมบัติของวัตถุ

การเรียนรู้ทั้งสองที่ได้กล่าวมานำไปสู่การเรียนรู้จากตัวอย่าง (learning from examples) ซึ่งผู้เรียนจะสรุปความรู้จากตัวอย่างที่ผู้สอนให้ โดยพิจารณาถึงสิ่งที่เหมือนกันของตัวอย่าง เพื่อใช้อธิบายตัวอย่างอื่นว่าเป็นสิ่งที่เรียนรู้ได้หรือไม่ ความรู้ที่เรียนรู้ได้เป็นความรู้ที่สามารถอธิบายตัวอย่างที่สอดคล้องกับสิ่งที่ต้องการให้เรียนรู้หรือเรียกว่าตัวอย่างบวก และไม่ครอบคลุมตัวอย่างที่ไม่สอดคล้องกับสิ่งที่ต้องการให้เรียนรู้หรือเรียกว่าตัวอย่างลบ การเรียนรู้แบบนี้มีความเหมาะสมเนื่องจากผู้สอนให้ตัวอย่างที่ดีแทนการให้ความรู้หรือทฤษฎีที่มีความซับซ้อน วิธีการดังที่กล่าวมานี้เป็นวิธีการเรียนรู้ของเครื่อง ซึ่งนำไปใช้ประโยชน์ในการเรียนรู้หลายด้าน เช่นการตรวจโรคของผู้ป่วย การทำนายคุณสมบัติของสวนประกอบทางเคมี

การโปรแกรมตรรกะเชิงอุปนัย (Inductive Logic Programming, ILP) [18]

การโปรแกรมตรรกะเชิงอุปนัยหรือเรียกโดยย่อว่า ไอแอลพี (ILP) เป็นงานวิจัยที่ใช้หลักการของการเรียนรู้ของเครื่องกับการโปรแกรมตรรกะ โดยส่วนมากระบบไอแอลพีมีส่วนประกอบ 3 ส่วน คือ ตัวอย่าง (E) ความรู้ภูมิหลัง (B) และสมมติฐาน (H) โดย E B และ H อยู่ในรูปแบบของการโปรแกรมตรรกะ ระบบไอแอลพีเรียนรู้จากตัวอย่างและความรู้ภูมิหลังสร้างสมมติฐานขึ้นมา

ทฤษฎีของไอแอลพีได้พัฒนามาจากทฤษฎีการพิสูจน์ (proof theory) และทฤษฎีโมเดล (model theory) สำหรับเพรดิเคตอันดับที่หนึ่ง การสร้างสมมติฐานโดยการอุปนัยใช้เทคนิคไออาร์ (Inverse Resolution, IR) อาร์แอลจีจี (Relative Least General Generalizations, RLGG) อินเวอร์สอิมพลิเคชัน (Inverse Implication) และไออี (Inverse Entailment, IE) นอกจากนี้การขจัด

ความซ้ำซ้อนและกระบวนการค้นหาเป็นเทคนิคที่มีบทบาทสำคัญ ส่วนการวิเคราะห์ผลลัพธ์ที่เกิดจากการเรียนรู้ใช้ทฤษฎีเรียนรู้เชิงคำนวณ (Computational Learning Theory, COLT)

ระบบไอแอลพีได้ถูกนำไปประยุกต์ใช้ในปัจจุบันได้แก่ การเรียนรู้กฎเกี่ยวกับสตรัคเจอร์แอคติวิตี้ของยา (learning drug structure-activity rules) การเรียนรู้กฎสำหรับการทำนายแบบมิวเทเจ็นนิซิส (learning rules for predicting mutagenesis) การเรียนรู้กฎสำหรับการทำนายโครงสร้างอันดับสองของโปรตีน (learning rules for predicting protein secondary structure) การเรียนรู้กฎจากฐานข้อมูลเกมหมากรุก (learnig rules from chess databases) การเรียนรู้กฎสำหรับการออกแบบไฟไนท์อิลิเมนท์เมช (learning rules for finite element mesh design) การเรียนรู้กฎเพื่อใช้ตรวจสอบคุณลักษณะตัวจ่ายไฟของดาวเทียม (learning diagnosis rules for qualitative models of satellite power supplies) การเรียนรู้คุณลักษณะของยูทูป (learning qualitative models of the U-tube system) และการรู้จำตัวพิมพ์อักษรภาษาไทย [11]

ตัวอย่างระบบไอแอลพีได้แก่ GOLEM PROGOL CIGOL FOIL เป็นต้น

<u>งานวิจัยด้านไอแอลพี</u> [17]

- งานวิจัยของ Plotkin ซึ่งถือเป็นรากฐานของการพัฒนาไอแอลพี โดยแนวคิดหลักๆคือ 1.รีเลทีฟสับซัมชัน (relative subsumption) และ 2.อาร์แอลจีจี (Relative Least General Generalization, RLGG) ที่ประสบความสำเร็จในระดับหนึ่ง แต่ยังมีข้อจำกัดเนื่องจากอาร์แอลจีจีของอนุประโยค 2 อนุประโยคไม่มีจำนวนจำกัด

- Shapiro ได้ใช้หลักการค้นหาจากเจนเนอรัลไปสู่สเปคซิฟิค (general-to-specific) ในรูปแบบของอนุประโยคฮอร์น (Horn Clause) ที่แตกต่างไปจากแนวทางของ Plotkin ที่ค้นหาจากสเปคซิฟิคไปสู่เจนเนอรัล (specific-to-general) นอกจากนี้ Shapiro ได้สร้างเทคนิคขั้นตอนวิธีการแก้จุดบกพร่อง (algorithmic debugging) คือถ้าโปรแกรมภาษาโปรล็อกไม่สมบูรณ์ ไม่ถูกต้อง หรือไม่มีจุดสิ้นสุด ระบบจะตรวจสอบว่าอนุประโยคใดที่ผิด และจะแทนที่อนุประโยคนั้นใหม่

- Sammut and Banerji กล่าวถึงระบบ MARVIN ซึ่งระบบจะทำการเจนเนอรัลไรซ์ตัวอย่าง 1 ตัวอย่าง ณ เวลาหนึ่ง โดยการอ้างอิงจากอนุประโยคภูมิหลัง (background clause)

- Muggleton and Buntine ได้สร้างโปรแกรม CIGOL โดยนำแต่ละตัวอย่างมาวิเคราะห์ร่วมกับความรู้ภูมิหลังในการสร้างเพรดิเคตใหม่

- Banerji, Wirth, Ishizaka, Ling and Dawes, Rouveirol and Puget กล่าวถึงวิธีการในการสร้างเพรดิเคตใหม่ที่เป็นการเรียนรู้เชิงประพจน์ (propositional) ซึ่งยังไม่เป็นระบบการเรียนรู้อันดับที่หนึ่ง (first-order learning systems)

- Quinlan ได้สร้างโปรแกรม FOIL โดยใช้หลักการของอนุประโยคฮอร์นอันดับที่หนึ่ง (first-order Horn clause) และการค้นหาสมมติฐานจะค้นหาจากเจนเนอรัลไปสู่สเปคซิฟิค รวมทั้งใช้วิทยาการศึกษาสำนึกเชิงสารสนเทศ (information based heuristic) ช่วยในการค้นหา รายละเอียดของ FOIL จะกล่าวถึงในส่วนถัดไป

- Buntine พยายามจำกัดรูปแบบของความรู้ภูมิหลัง โดยการสร้างแบบจำลองเหอแบนด์ (Herband model) ซึ่งประสบความสำเร็จในระดับหนึ่ง โดยขึ้นกับความซับซ้อนของอนุประโยคที่สร้างขึ้น

- Muggleton and Feng ประยุกต์ใช้ข้อจำกัดเชิงกำหนด (determinate) กับสมมติฐาน และสร้างโปรแกรม GOLEM ที่มีประสิทธิภาพใกล้เคียงกับโปรแกรม FOIL ของ Quinlan

- Muggleton ทำการสร้างโปรแกรม PROGOL ที่ใช้หลักการของเอ็มดีไออาร์ (Mode-Directed Inverse Resolution, MDIR) และการค้นหาสมมติฐานเป็นแบบเจนเนอรัลไปสู่สเปคซิฟิค

ระบบ FOIL [19]

ระบบไอแอลพีที่ใช้ในการเรียนรู้คำกำกวมในงานวิจัยนี้คือ ระบบ FOIL รุ่น 6.4 ซึ่งพัฒนาโดย Ross Quinlan และ Mike Cameron-Jones เหตุผลที่เลือกใช้ระบบ FOIL เนื่องจากเป็นระบบการเรียนรู้ที่มีประสิทธิภาพ ผลลัพธ์จากการเรียนรู้อยู่ในรูปแบบของกฎที่มนุษย์สามารถเข้าใจและนำไปใช้ประโยชน์ได้ และสามารถใช้ในงานวิจัยทางด้านการศึกษาได้โดยไม่เสียค่าใช้จ่าย ในเนื้อหาต่อไปนี้จะกล่าวถึงความหมายของระบบ FOIL การใช้งานระบบ FOIL พร้อมกับแสดงตัวอย่างการใช้งานและขั้นตอนการทำงานของ FOIL

1. ความหมายของระบบ FOIL

ระบบ FOIL เป็นรูปแบบของการโปรแกรมตรรกะเชิงอุปนัย โดยสมมติฐานที่เกิดจากการเรียนรู้อยู่ในรูปแบบอนุประโยคฟังก์ชันฟรีฮอร์นคลอส (function-free Horn clause) จากตัวอย่างที่เป็นคู่ลำดับ (tuple) ในรูปแบบการประมวลผลแบบกลุ่ม (batch) และใช้วิทยาการศึกษาสำนึกเชิงสารสนเทศ (information-based heuristic หรือ Gain) ในการค้นหา

2. การใช้งานระบบ FOIL

ข้อมูลทั้งหมดที่จะใช้ในการเรียนรู้จัดเก็บในแฟ้มข้อมูลที่ลงท้ายด้วยอะไรก็ได้ เช่น ta_klom.txt แล้วใช้คำสั่งเริ่มทำงานดังนี้

$ foil6 <ชื่อแฟ้มข้อมูลนำเข้า >ชื่อแฟ้มผลลัพธ์

เช่น $ foil6 <ta_klom.txt >ta_klom.out

ข้อมูลที่ใช้ในการเรียนรู้ของ FOIL ประกอบด้วย 3 ส่วนดังนี้

2.1 ลักษณะของชนิด (specification of types) ประกอบด้วยชื่อของชนิดของค่าคงที่ตามด้วยเครื่องหมาย : แล้วตามด้วยค่าคงที่ โดยที่แต่ละค่าคงที่จะคั่นด้วยเครื่องหมาย , และจบชนิดนั้นด้วยเครื่องหมาย .

2.2 การนิยามส่วนเพิ่มเติมของความสัมพันธ์ (extensional definitions of relations) ก่อนที่จะเริ่มเขียนข้อมูลส่วนนี้จะต้องเว้นบรรทัดหนึ่งบรรทัด แต่ละความสัมพันธ์จะประกอบด้วยส่วนหัวและหนึ่งหรือสองกลุ่มของตัวอย่าง ส่วนหัวเขียนได้ดังนี้

name(type, type,..., type)

ถ้าไม่ใช่ความสัมพันธ์ของสิ่งที่ต้องการเรียนรู้ หรือเรียกว่าความรู้ภูมิหลังจะมีเครื่องหมาย * นำหน้าชื่อความสัมพันธ์นั้น นอกจากชื่อความสัมพันธ์แล้วในวงเล็บเป็นชนิดของอาร์กิวเมนต์ นอกจากส่วนหัวแล้วเป็นกลุ่มของตัวอย่างที่มีลักษณะดังนี้

positive tuple

positive tuple

...

;

negative tuple

negative tuple

...

.

โดยที่ positive tuple เป็นตัวอย่างบวก และ negative tuple เป็นตัวอย่างลบ แต่ละตัวอย่างประกอบด้วยค่าคงที่ แต่ละค่าคงที่คั่นด้วยเครื่องหมาย , และในหนึ่งตัวอย่างต้องเขียนภายในบรรทัดเดียวกัน ส่วนที่เหลือเป็นทางเลือกคือ เครื่องหมาย ; ทำหน้าที่แบ่งแยกตัวอย่างบวกออกจากตัวอย่างลบ

2.3 ส่วนทดสอบสำหรับความรู้ที่เรียนรู้ได้ (test case for learned definitions) ส่วนนี้เป็นทางเลือกซึ่งไม่มีก็ได้ มีลักษณะดังนี้

บรรทัดว่างหนึ่งบรรทัด (เป็นการบอกจุดเริ่มต้นของส่วนทดสอบ)

relation name

test tuples

.

แต่ละตัวอย่างทดสอบประกอบด้วยตัวอย่างบวกที่ตามด้วยเครื่องหมาย :+ ถ้าตัวอย่างสอดคล้องกับความสัมพันธ์นั้น หรือตัวอย่างลบที่ตามด้วยเครื่องหมาย :- สำหรับตัวอย่างที่ไม่สอดคล้องกับความสัมพันธ์นั้น

3. ขั้นตอนการทำงานของระบบ FOIL

ขั้นตอนการทำงานของระบบ FOIL แบ่งออกเป็น 2 ขั้นตอนคือ

3.1 ขั้นตอนวงวนนอก (Outermost level) คือเริ่มต้นด้วยการสร้างกลุ่มข้อมูลสอนของสิ่งที่ต้องการเรียนรู้ ที่ประกอบด้วยตัวอย่างบวกและตัวอย่างลบ แล้วทำการหาอนุประโยคจากขั้นตอนวงวนใน และเอาตัวอย่างบวกที่ครอบคลุมด้วยอนุประโยคที่หาได้ออกจากกลุ่มข้อมูลสอน จนกระทั่งไม่มีตัวอย่างบวกในกลุ่มข้อมูลสอน หรือเป็นไปตามเกณฑ์การหยุดที่1 (stopping criterion1) ที่จะกล่าวถึงในเกณฑ์การหยุดในส่วนถัดไป

3.2 ขั้นตอนวงวนใน (Inner loop) เริ่มต้นด้วยการสร้างกลุ่มสอนย่อย (local training set) หรือเรียกว่า T_i ขึ้นจากกลุ่มข้อมูลสอน และกำหนดให้ i มีค่าเท่ากับ 1 แล้วทำการหาสัญพจน์ (literal) ที่มีค่าเกน (Gain) มากที่สุดเพิ่มเข้าไปทางขวามือของอนุประโยค โดยเริ่มจากซ้ายไปขวา จนกระทั่งไม่มีจำนวนตัวอย่างลบในกลุ่มข้อมูลสอนย่อยของสัญพจน์ที่เกิดขึ้น หรือเป็นไปตามเกณฑ์การหยุดที่2 (stopping criterion2) ซึ่งจะกล่าวถึงในเกณฑ์การหยุดในส่วนถัดไป

แต่ละสัญพจน์มีรูปแบบที่เป็นไปได้ดังนี้

$Q(v_1,....v_r)$ โดยที่ Q เป็นความสัมพันธ์ของความรู้ภูมิหลัง และต้องมีอาร์กิวเมนต์อย่างน้อย 1 ตัวใน v_i ที่มีอยู่ในอนุประโยค

$Equal(v_j,v_k)$ โดยที่ v_j และ v_k เป็นอาร์กิวเมนต์ที่มีอยู่ในอนุประโยค

รูปแบบที่เป็นนิเสธของรูปแบบที่1 หรือรูปแบบที่2

ค่าวิทยาการศึกษาสำนึกเชิงสารสนเทศหรือค่าเกน หาได้จาก

$$Gain = T_i^{++} * (I(T_i) - I(T_{i+1}))$$

โดยที่ $I(T_i)$ มีค่าเท่ากับ $- \log2 (T_i^+ / (T_i^+ + T_i^-))$

เมื่อทำงานในขั้นตอนวงวนในเสร็จแล้วจะได้อนุประโยคที่ประกอบด้วยสัญพจน์ที่เพิ่มขึ้นจากซ้ายไปขวา ซึ่งจะทำการตัด (prune) สัญพจน์ออกจากอนุประโยคทีละสัญพจน์ในทางตรงกันข้ามคือเริ่มจากสัญพจน์ตัวสุดท้ายของอนุประโยคออก แล้วทดสอบอนุประโยคที่เหลือนั้นว่าครอบคลุมตัวอย่างบวกทั้งหมดที่ครอบคลุมด้วยอนุประโยคที่ยังไม่ได้ตัดสัญพจน์ออก และต้องไม่ครอบคลุมตัวอย่างลบ ถ้าไม่เป็นไปตามนี้สัญพจน์ที่ถูกตัดออกจากอนุประโยคจะถูกใส่กลับเข้าไปในอนุประโยคเหมือนเดิม ที่ทำเช่นนี้เพื่อไม่ให้มีสัญพจน์ที่ซ้ำกันในอนุประโยคผลลัพธ์

หอสมุดกลาง สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

เมื่อทำงานในขั้นตอนวงวนนอกเสร็จแล้วจะทำการตรวจสอบว่าแต่ละอนุประโยคที่ได้นั้นต้องไม่ครอบคลุมตัวอย่างบวกซ้ำกัน ถ้าซ้ำกันจะทำการตัดอนุประโยคที่ซ้ำกันทิ้ง

ระบบ FOIL ใช้เกณฑ์การหยุด (stopping criteria) ในกรณีที่อนุประโยคที่เกิดขึ้นจากการเรียนรู้ไม่สมบูรณ์ และจะทำการพิมพ์ข้อความเตือน เกณฑ์การหยุดจะใช้วิทยาการศึกษาสำนึกแบบความยาวการเข้ารหัส (encoding-length heuristics) เพื่อจำกัดอนุประโยคที่ไม่ถูก เช่นอนุประโยคที่ครอบคลุมตัวอย่างลบ วิทยาการศึกษาสำนึกที่ใช้คือ จำนวนบิตที่เข้ารหัสอนุประโยคต้องไม่เกินจำนวนบิตที่แสดงตัวอย่างบวกที่ครอบคลุมโดยอนุประโยค

จำนวนบิตที่แสดงตัวอย่างบวก p ตัวออกจาก |T| จำนวนตัวอย่างของกลุ่มข้อมูลสอน ดังนี้

$$EC = \log_2(|T|) + \log_2 \begin{bmatrix} |T| \\ p \end{bmatrix}$$

$B(L_i)$ หรือจำนวนบิตที่เข้ารหัสของหนึ่งสัญพจน์มีค่าเท่ากับ

$1 + \log_2$(จำนวนของความสัมพันธ์) + $\log_2$(จำนวนของอาร์กิวเมนต์)

จำนวนบิตที่เข้ารหัสอนุประโยคหนึ่งอนุประโยคของ n สัญพจน์มีค่าเท่ากับ

$$BC = \text{sum}(B(L_i)) - \log_2(n!)$$

เทอมสุดท้ายเกิดขึ้นเนื่องจากข้อเท็จจริงที่ว่าลำดับของสัญพจน์แต่ละลำดับมีความสำคัญเท่ากัน

เกณฑ์การหยุดที่1 หรือเมื่อไม่มีอนุประโยคใดสามารถเพิ่มเข้าไปในการนิยามของสิ่งที่ต้องการเรียนรู้ได้ภายใต้ข้อจำกัดของวิทยาการศึกษาสำนึกที่ใช้ ถ้าเป็นไปตามเกณฑ์การหยุดที่1ได้ว่าอนุประโยคที่เกิดขึ้นจากการเรียนรู้ไม่สามารถครอบคลุมตัวอย่างบวกทั้งหมดได้ ดังนั้นสมมติฐานที่เกิดขึ้นจึงได้กลุ่มของอนุประโยคที่ไม่สมบูรณ์

เกณฑ์การหยุดที่2 หรือเมื่อไม่มีสัญพจน์ใดสามารถเพิ่มเข้าไปในอนุประโยคได้ภายใต้ข้อจำกัดของวิทยาการศึกษาสำนึกที่ใช้ เมื่อเป็นไปตามเกณฑ์การหยุดที่2 อนุประโยคที่เกิดขึ้นในขั้นตอนวงวนในจะเป็นอนุประโยคผลลัพธ์เมื่ออนุประโยคนั้นมีความถูกต้องมากกว่าหรือเท่ากับ 85% ซึ่งจะเป็นอนุประโยคที่ไม่สมบูรณ์คือมีความถูกต้องน้อยกว่า 100 % นอกจากนี้ระบบ FOIL ยังมีความสะดวกสำรอง (backup facility) เพื่อใช้ในการกลับไปค้นหาสัญพจน์อื่นเมื่อการสร้างอนุประโยคล้มเหลว คือเมื่อมีสัญพจน์ถัดไปที่เป็นได้หลายสัญพจน์มีค่าเกนเท่ากัน ระบบ FOIL จะตั้งจุดตรวจสอบเมื่อการค้นหาอนุประโยคล้มเหลวและมีจุดตรวจสอบระบบจะกลับไปยังจุดตรวจสอบล่าสุด แล้วจะสร้างอนุประโยคด้วยสัญพจน์อื่นต่อไป ระบบมีการจำกัดจำนวนมากที่สุดของสัญพจน์ที่ให้เลือกได้กับจำนวนมากที่สุดของจุดตรวจสอบ ซึ่งในรุ่น 6.4 นี้มีค่าเท่ากับ 5 และ 20 ตามลำดับ

การเพิ่มสัญพจน์หนึ่งๆเข้าไปในอนุประโยคจะทำได้ถ้าจำนวนบิตที่เข้ารหัสอนุประโยคใหม่ไม่เกินจำนวนบิตที่แสดงตัวอย่างบวกที่ครอบคลุมอยู่ ถ้าเมื่อเป็นไปตามเกณฑ์การหยุดที่2 หรือเมื่อ

ไม่มีอนุประโยคถูกสร้างขึ้นในขั้นตอนวงวนในแล้วจะได้ว่าการทำงานในขั้นตอนวงวนนอกจะสิ้นสุดหรือเป็นไปตามเกณฑ์การหยุดที่1

ข้อดีของ FOIL

1. อนุประโยคผลลัพธ์ที่ได้จากการเรียนรู้สามารถเขียนให้อยู่ในรูปแบบ ถ้า...แล้ว (if_then) ได้ ซึ่งจะทำให้ง่ายต่อการเข้าใจ

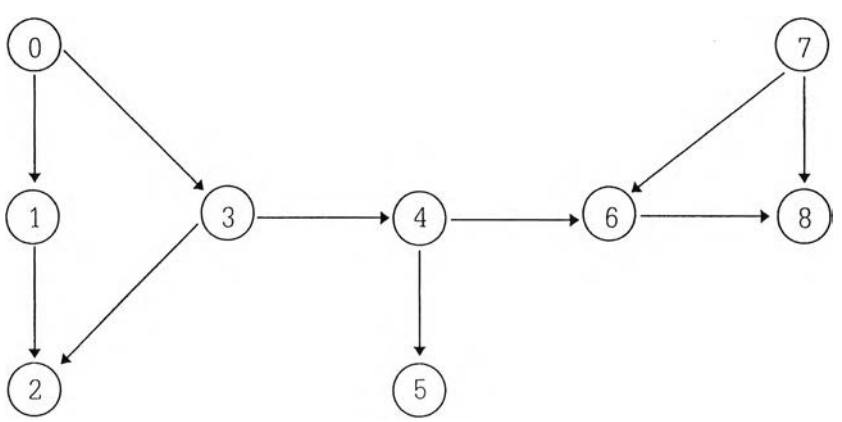
2. สามารถหานิยามของสิ่งที่ต้องการเรียนรู้ในรูปแบบเรียกซ้ำได้

3. สามารถใช้ความรู้ภูมิหลังในการสร้างอนุประโยคผลลัพธ์ ซึ่งทำให้สมมติฐานที่ได้จากการเรียนรู้สามารถนิยามได้กว้างขึ้น

ตัวอย่าง can-reach แสดงการเรียนรู้ทิศทางของรูปแบบข่ายงาน (directional networks) ดังรูปที่ 2.1 ของระบบ FOIL โดยมีความรู้ภูมิหลังคือ linked-into(X,Y) : (0,1), (0,3), (1,2), (3,2), (3,4), (4,5), (4,6), (6,8), (7,6), (7,8) และกลุ่มข้อมูลสอนของความสัมพันธ์ can-reach(X,Y) : (0,1), (0,2), (0,3), (0,4), (0,5), (0,6), (0,8), (1,2), (3,2), (3,4), (3,5), (3,6), (3,8), (4,5), (4,6), (4,8), (6,8), (7,6), (7,8) แล้วได้กลุ่มของกฎที่ได้จากการเรียนรู้ของ FOIL คือ

can-reach(X,Y) :- linked-into(X,Y).

can-reach(X,Y) :- linked-into(X,Z), can-reach(Z,Y).



รูปที่ 3.1 รูปแบบข่ายงานที่แสดงทิศทาง

FOIL เริ่มทำการหาอนุประโยคแรกจากกลุ่มสอนย่อย T_1 ที่มี 81 ตัวอย่าง โดยแบ่งเป็นตัวอย่างบวกจำนวน 19 ตัวอย่างและตัวอย่างลบจำนวน 62 ตัวอย่างดังนี้

T_1^+ : (0,1) (0,2) (0,3) (0,4) (0,5) (0,6) (0,8) (1,2) (3,2) (3,4)
(3,5) (3,6) (3,8) (4,5) (4,6) (4,8) (6,8) (7,6) (7,8)

T_1^- : (0,0) (0,7) (1,0) (1,1) (1,3) (1,4) (1,5) (1,6) (1,7) (1,8)

(2,0) (2,1) (2,2) (2,3) (2,4) (2,5) (2,6) (2,7) (2,8) (3,0)
(3,1) (3,3) (3,7) (4,0) (4,1) (4,2) (4,3) (4,4) (4,7) (5,0)
(5,1) (5,2) (5,3) (5,4) (5,5) (5,6) (5,7) (5,8) (6,0) (6,1)
(6,2) (6,3) (6,4) (6,5) (6,6) (6,7) (7,0) (7,1) (7,2) (7,3)
(7,4) (7,5) (7,7) (8,0) (8,1) (8,2) (8,3) (8,4) (8,5) (8,6)
(8,7) (8,8)

ถ้าสัญพจน์แรกที่ถูกเลือกเพิ่มเข้าไปทางขวามือของอนุประโยคแรกคือ linked-into(X,Y) ซึ่งจะได้เป็นอนุประโยคแรกคือ can-reach(X,Y) :- linked-into(X,Y). เนื่องจากสัญพจน์แรกนี้ครอบคลุมแต่ตัวอย่างบวกเท่านั้น ในรอบที่สองในการหาอนุประโยคต่อไปในขั้นตอนวงวนในด้วยกลุ่มสอนย่อย T_1 ที่ยังมีตัวอย่างบวกจำนวน 9 ตัวอย่างและตัวอย่างลบจำนวน 62 ตัวอย่าง ซึ่งตัวอย่างลบยังคงเหมือนที่กล่าวมาแล้วส่วนตัวอย่างบวกมีดังนี้ (0,2) (0,4) (0,5) (0,6) (0,8) (3,5) (3,6) (3,8) (4,8) ถ้า FOIL เลือกสัญพจน์ linked-into(X,Z) จะได้ว่าตัวอย่างลบที่ขึ้นต้นด้วย (2,...) (5,...) และ (8,...) จะถูกลบออก เนื่องจากสัญพจน์นี้ใช้ตัวแปรตัวใหม่คือ Z จึงเขียนตัวอย่างคู่ลำดับ (X,Y,Z) จากเดิม (X,Y) สำหรับความสัมพันธ์ linked-into ที่มีคู่ลำดับแบบ (X,Z) ซึ่งจะได้กลุ่มสอนย่อย T_2 ที่มีตัวอย่างบวกจำนวน 18 ตัวอย่างและตัวอย่างลบจำนวน 54 ตัวอย่างดังนี้

T_2^+ : (0,2,1) (0,2,3) (0,4,1) (0,4,3) (0,5,1) (0,5,3) (0,6,1)
(0,6,3) (0,8,1) (0,8,3) (3,5,2) (3,5,4) (3,6,2) (3,6,4)
(3,8,2) (3,8,4) (4,8,5) (4,8,6)

T_2^- : (0,0,1) (0,0,3) (0,7,1) (0,7,3) (1,0,2) (1,1,2) (1,3,2)
(1,4,2) (1,5,2) (1,6,2) (1,7,2) (1,8,2) (3,0,2) (3,0,4)
(3,1,2) (3,1,4) (3,3,2) (3,3,4) (3,7,2) (3,7,4) (4,0,5)
(4,0,6) (4,1,5) (4,1,6) (4,2,5) (4,2,6) (4,3,5) (4,3,6)
(4,4,5) (4,4,6) (4,7,5) (4,7,6) (6,0,8) (6,1,8) (6,2,8)
(6,3,8) (6,4,8) (6,5,8) (6,6,8) (6,7,8) (7,0,6) (7,0,8)
(7,1,6) (7,1,8) (7,2,6) (7,2,8) (7,3,6) (7,3,8) (7,4,6)
(7,4,8) (7,5,6) (7,5,8) (7,7,6) (7,7,8)

เมื่อเพิ่มสัญพจน์ linked-into(X,Z) เข้าไปทางขวามือของอนุประโยคที่สองแล้วพบว่ายังคงมีตัวอย่างลบอยู่จึงยังต้องทำการหาสัญพจน์ตัวต่อไป ถ้าเลือกสัญพจน์ตัวต่อไปคือ can-reach(Z,Y) ซึ่งสัญพจน์นี้จะครอบคลุมแต่ตัวอย่างบวกเท่านั้นคือ

T_3^+ : (0,2,1) (0,2,3) (0,4,3) (0,5,3) (0,6,3) (0,8,3) (3,5,4) (3,6,4) (3,8,4) (4,8,6)

ดังนั้นอนุประโยคที่สองคือ can-reach :- linked-into(X,Z), can-reach(Z,Y).
เนื่องจากจำนวนตัวอย่างบวกทั้งหมดในกลุ่มข้อมูลสอนถูกครอบคลุมด้วยอนุประโยคทั้งสองนี้ ได้ว่านิยามของ can-reach ที่ได้จึงสมบูรณ์ นอกจากนี้ถ้าเลือกสัญพจน์ linked-into(Z,Y) แทนสัญพจน์ can-reach(Z,Y) ซึ่งก็จะครอบคลุมแต่ตัวอย่างบวกเท่านั้นคือ T_3^+ : (0,2,1) (0,2,3) (0,4,3) (3,5,4) (3,6,4) (4,8,6) แต่ FOIL เลือกสัญพจน์ can-reach(Z,Y) แทนด้วยเหตุผลที่ว่า ในอนุประโยคที่สองมีตัวอย่างบวกจำนวน 18 ตัวอย่างและตัวอย่างลบจำนวน 54 ตัวอย่างได้ว่า
$I(T_2) = -\log_2(18/(18+54)) = 2.0$ สำหรับสัญพจน์ can-reach(Z,Y) มีแต่ตัวอย่างบวกจำนวน 10 ตัวอย่างได้ค่า $I(T_3) = -\log_2(10/(10+0)) = 0$ และค่าเกนเท่ากับ $10 * (I(T_2) - I(T_3)) = 20.0$ และอีกทางเลือกหนึ่งคือสัญพจน์ linked-into(Z,Y) ซึ่งก็มีแต่ตัวอย่างบวกเท่านั้นจำนวน 6 ตัวอย่างได้ค่า $I(T_3)=-\log_2(6/(6+0))=0$ และค่าเกนเท่ากับ $6*(I(T_2)-I(T_3))= 12.0$เนื่องจากสัญพจน์ can-reach(Z,Y) มีค่าเกนมากกว่าสัญพจน์ linked-into(Z,Y) FOIL จึงเลือกสัญพจน์ can-reach(Z,Y) ดังนั้นอนุประโยคผลลัพธ์ที่ได้จากการเรียนรู้คือ

can-reach(X,Y) :- linked-into(X,Y).
can-reach(X,Y) :- linked-into(X,Z), can-reach(Z,Y).

ระบบ RIPPER [12,13]

ระบบที่ใช้ในการเรียนรู้คำกำกวมอีกระบบหนึ่งคือ ระบบ RIPPER เพื่อนำมาเปรียบเทียบกับการเรียนรู้โดยใช้ระบบ FOIL เหตุผลที่เลือกใช้ระบบ RIPPER เนื่องจากเป็นระบบการเรียนรู้ที่มีประสิทธิภาพ ผลลัพธ์ที่ได้จากการเรียนรู้อยู่ในรูปแบบของกฎที่มนุษย์สามารถเข้าใจ และนำไปใช้ประโยชน์ได้ และสามารถใช้ในงานวิจัยทางด้านการศึกษาได้โดยไม่เสียค่าใช้จ่าย ในเนื้อหาต่อไปนี้จะกล่าวถึงความหมายของระบบ RIPPER การใช้งานระบบ RIPPER และขั้นตอนการทำงานของระบบ RIPPER

1. ความหมายของระบบ RIPPER

ระบบ RIPPER เป็นรูปแบบของการเรียนรู้แบบกฎประพจน์ (propositional rule learning) ที่ทำการสร้างกลุ่มของกฎ (ruleset) ของแต่ละประเภทที่กำหนดไว้อย่างถูกต้องในข้อมูลตัวอย่าง (training data) กฎที่สร้างขึ้นในกลุ่มของกฎอยู่ในรูปแบบเงื่อนไขที่เชื่อมด้วย และ(and) ดังนี้

if T_1 and T_2 and ... and T_n then class C_x.

โดยที่ T_1 and T_2 and ... and T_n คือตัวของกฎหรือเงื่อนไขต่างๆ ซึ่งมีรูปแบบที่เป็นไปได้ดังนี้

$A_n = v$ $A_c \geq \theta$ $A_c \leq \theta$ $v \in A_s$ หรือรูปนิเสธของรูปแบบที่กล่าวมา โดยที่ A_n เป็นนามของลักษณะประจำ (nominal attribute) v เป็นค่าที่มีในข้อมูลสอนสำหรับ A_n A_cเป็นตัวแปรแบบต่อเนื่อง θ เป็นค่าหนึ่งของ A_c ในข้อมูลสอน ส่วน A_s เป็นกลุ่มของค่าลักษณะประจำ (set-value attribute) และ v เป็นสมาชิกของ A_s

C_x เป็นประเภทของสิ่งที่ต้องการเรียนรู้ ในงานวิจัยนี้คือคำกำกวมแต่ละคำ

2. การใช้งานของระบบ RIPPER

คำสั่งเริ่มทำงานคือ

$ripper ชื่อแฟ้ม 1>ชื่อแฟ้มข้อมูล.out 2>&1

RIPPER ใช้ข้อมูล 4 แฟ้ม มีดังนี้

2.1 แฟ้มข้อมูล (data file) แฟ้มข้อมูลจะลงท้ายชื่อแฟ้มด้วย .data ประกอบด้วยกลุ่มของข้อมูลตัวอย่างที่ถูกจัดประเภทแล้ว แต่ละตัวอย่างประกอบด้วยรายการของค่าของคุณลักษณะ ที่คั่นด้วยเครื่องหมาย , ตามด้วยประเภทของตัวอย่างและเครื่องหมาย . โดยทั่วไปหนึ่งตัวอย่างอยู่บนบรรทัดเดียวกันแต่อาจแยกอยู่คนละบรรทัดก็ได้ ค่าของคุณลักษณะจะต้องเรียงตามลำดับที่กำหนดไว้ในแฟ้มชื่อ

2.2 แฟ้มทดสอบ (test file) แฟ้มทดสอบจะลงท้ายชื่อแฟ้มด้วย .test ประกอบด้วยข้อมูลที่มีรูปแบบเหมือนกับแฟ้มข้อมูล แต่ข้อมูลในแฟ้มทดสอบนี้จะถูกใช้สำหรับทดสอบกลุ่มของกฎที่เกิดจากการเรียนรู้

2.3 แฟ้มชื่อ (name file) แฟ้มชื่อจะลงท้ายชื่อแฟ้มด้วย .names ประกอบด้วยชื่อของประเภทที่ถูกจัดไว้ตามด้วยเครื่องหมาย . กับกลุ่มของการนิยามคุณลักษณะที่ใช้ในแฟ้มข้อมูล แต่ละคุณลักษณะที่นิยามประกอบด้วยชื่อของคุณลักษณะตามด้วยเครื่องหมาย : และตามด้วย atom ถ้าค่าของคุณลักษณะเป็นอะตอม (อะตอมประกอบด้วยตัวอักษร ตัวเลขและเครื่องหมาย _ ที่ต้องขึ้นต้นด้วยตัวอักษร) หรือ continuous ถ้าค่าเป็นแบบต่อเนื่อง หรือ set ถ้าค่าเป็นแบบกลุ่มของค่าคงที่ หรือ bag ถ้าค่าเป็นแบบถุง (bag-valued) หรือ symbolic ถ้าค่าเป็นสัญลักษณ์ หรือ nominal ถ้าค่าเป็นกลุ่มของของสัญลักษณ์ที่เป็นไปได้ โดยที่ทุกการกำหนดนิยามคุณลักษณะลงท้ายด้วยเครื่องหมาย . ดังตัวอย่าง

class_name.

attribute_name1 : atom.

attribute_name2 : set.

2.4 แฟ้มไวยากรณ์ (grammar file) แฟ้มไวยากรณ์จะลงท้ายชื่อแฟ้มด้วย .gram เป็นการกำหนดเงื่อนไขที่เป็นรูปแบบของกฎที่จะได้ออกมา

แฟ้มทดสอบ แฟ้มชื่อและแฟ้มไวยากรณ์ เป็นทางเลือกซึ่งจะไม่มีก็ได้โดยถ้าไม่มีแฟ้มทดสอบ ระบบ RIPPER จะไม่ทำการทดสอบกลุ่มของกฎที่เกิดจากการเรียนรู้ ถ้าไม่มีแฟ้มชื่อ RIPPER จะทำการกำหนดชื่อของคุณลักษณะและประเภทของตัวอย่างเอง และถ้าไม่มีแฟ้มไวยากรณ์ RIPPER จะทำการกำหนดเงื่อนไขรูปแบบของกฎที่จะได้จากการเรียนรู้ขึ้นมาเอง

3. ขั้นตอนการทำงานของระบบ RIPPER

ระบบ RIPPER ได้ปรับปรุงมาจากการเรียนรู้แบบไอเร็ป (IREP : Incremental Reduced Error Pruning) ที่พัฒนาโดย Furnkranz and Widmer คือ 1. ในการตัดกฎ (Pruning rules) โดย RIPPER จะยอมให้มีการตัดกฎที่มีเงื่อนไขมากกว่า 1 เงื่อนไขได้ ในขณะที่ไอเร็ปจะตัดกฎที่มีเงื่อนไข 1 เงื่อนไขเท่านั้น 2. ในการหยุดเพิ่มกฎ RIPPER จะหยุดเพิ่มกฎเมื่อกฎที่ได้นั้นมีความยาวของคุณสมบัติ (description length) ของกลุ่มของกฎและตัวอย่าง มีค่ามากกว่าความยาวของคุณสมบัติที่น้อยที่สุด โดย Cohen [12] ได้ทำการกำหนดให้ความยาวของคุณสมบัติที่น้อยที่สุดมีค่าเท่ากับ 64 ส่วนไอเร็ปจะหยุดเพิ่มกฎเมื่อกฎที่ได้มีข้อผิดพลาดมากกว่า 50 % และ 3. RIPPER มีคุณสมบัติเพิ่มเติมที่ไอเร็ปไม่มีคือ อนุญาตให้มีตัวแปรแบบตัวเลข และสามารถหาคำตอบจากตัวอย่างหลายคลาส (Multiple class) ได้ ในขณะที่ไอเร็ปยอมให้ตัวอย่างมีได้แค่ 2 คลาสเท่านั้น ในงานวิจัยนี้ตัวอย่างที่ใช้ในการเรียนรู้ใน 1 แฟ้มข้อมูลจะมีแค่ 2 คลาสเท่านั้น

การทำงานของ RIPPER คือจะทำการสร้างกลุ่มของกฎของแต่ละประเภทที่กำหนดไว้อย่างถูกต้องในข้อมูลตัวอย่าง โดยจะทำการแบ่งข้อมูลตัวอย่างออกเป็นกลุ่มการเติบโต (growing set) กับกลุ่มการตัด (pruning set) แล้วทำการสร้างกฎจากกลุ่มเติบโต การสร้างกฎเริ่มต้นด้วยไม่มีตัวของกฎ แล้วทำการเพิ่มเงื่อนไขเข้าไปในตัวของกฎจากซ้ายไปขวาจนกระทั่งไม่ครอบคลุมตัวอย่างลบของกลุ่มการเติบโต หลังจากกฎหนึ่งถูกสร้างขึ้นกฎจะถูกทำการตัด (prune) โดย RIPPER จะตัดเงื่อนไขทีละเงื่อนไขในทิศทางตรงข้ามโดยจะเริ่มจากเงื่อนไขสุดท้ายของกฎ แล้วนำกฎที่ได้ไปหาค่าผิดพลาดในกลุ่มการตัด ซึ่งจะตัดเงื่อนไขที่ทำให้ค่าผิดพลาดในกลุ่มการตัดมีค่ามากที่สุด หลังจากนั้น RIPPER จะนำกฎที่ได้ไปใส่ไว้ในกลุ่มของกฎ (ruleset) หลังจากกฎหนึ่งๆเพิ่มเข้าไปในกลุ่มของกฎ ระบบจะทำการคำนวณหาค่าความยาวของคุณสมบัติ (description length) ของกลุ่มของกฎและตัวอย่าง ซึ่งจะทำการหยุดเพิ่มกฎเมื่อความยาวของคุณสมบัติของกลุ่มของกฎและตัวอย่างมีค่ามากกว่าค่าความยาวของคุณสมบัติที่น้อยที่สุดที่คำนวณได้ จาก [12] ได้ทำการกำหนดค่าความยาวของคุณสมบัติที่น้อยที่สุดมีค่าเท่ากับ 64 หรือจะหยุดเพิ่มกฎเมื่อกฎที่ได้จากการเรียนรู้ครอบคลุมตัวอย่างบวกทั้งหมด

ข้อดีของ RIPPER

1. กฎที่ได้จากการเรียนรู้จะอยู่ในรูปแบบเงื่อนไข ถ้า...แล้ว (if_then) ทำให้ง่ายต่อการทำความเข้าใจ

2. การทำงานของ RIPPER จะทำงานได้รวดเร็วกว่าวิธีการเรียนรู้แบบกฎแบบอื่นๆ (Rule learning algorithm) ในกรณีที่ตัวอย่างที่นำมาใช้ในการเรียนรู้มีจำนวนมาก

3. ผู้ใช้สามารถกำหนดเงื่อนไขที่เป็นรูปแบบของกฎที่จะได้ออกมา

4. RIPPER มีคุณสมบัติการใช้กลุ่มของค่าลักษณะประจำ (set-value attribute) ซึ่งในกรณีนี้จะช่วยให้กฎที่ได้ออกมามีรูปแบบกะทัดรัด และสะดวกในการใส่ตัวอย่างในการเรียนรู้