บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การเรียนรู้ของเครื่อง

การศึกษาด้านการเรียนรู้ของเครื่อง (Mitchell, 1997) มีจุดมุ่งหมายเพื่อทำให้โปรแกรมคอมพิวเตอร์ได้พัฒนาความสามารถของตัวเองโดยอัตโนมัติจากตัวอย่างที่ผู้สอนกำหนดให้ ตัวอย่างเช่น การสร้างโปรแกรมที่รู้จำภาพตัวอักษรซึ่งวัดความสามารถในการรู้จำจากเปอร์เซ็นต์ของภาพตัวอักษรที่รู้จำได้ถูกต้อง โดยใช้ตัวอย่างเป็นภาพตัวอักษรซึ่งผู้สอนป้อนให้ เป็นต้น

โดยในการเรียนรู้โดยอาศัยตัวอย่างข้อมูลนั้น หากใช้จำนวนข้อมูลตัวอย่างเพียงส่วนหนึ่งมาใช้ในการเรียนรู้ เราจะเรียนกวิธีการเรียนรู้นี้ว่า การเรียนรู้เชิงอุปนัย (Inductive learning) ซึ่งอยู่บนสมมติฐานที่ว่า ถ้าเราสามารถหาสมมติฐานที่สามารถอธิบายเซตข้อมูลหนึ่งได้จากตัวอย่างข้อมูลซึ่งมากเพียงพอแล้ว สมมติฐานที่ได้นั้นสามารถที่จะใช้ในการอธิบายตัวอย่างข้อมูลที่ไม่เคยเห็นหรือเรียนรู้ตัวอย่างข้อมูลนั้นมาก่อนได้ ด้วยสมมติฐานนี้ทำให้การเรียนรู้เชิงอุปนัยจำเป็นต้องทดสอบกับตัวอย่างข้อมูลทดสอบซึ่ง ไม่อยู่ในเซตข้อมูลที่ใช้ในการเรียนรู้จึงจะสามารถตรวจสอบได้ว่าสมมติฐานที่ได้นั้นดีพอหรือไม่

วิธีการเรียนรู้ของเครื่องสามารถแบ่งออกได้เป็น 2 ประเภท คือ วิธีการที่เรียกว่า วิธีการแบบ black-box และ วิธีการแบบ white-box ซึ่งในวิธีการแบบ black-box นั้น การเรียนรู้ของเครื่องจะสร้างกฎหรือหลักการที่ทำให้เครื่องสามารถเรียนรู้และจำแนกตัวอย่างได้อย่างถูกต้อง แต่ไม่สามารถเข้าใจได้โดยมนุษย์อันเนื่องมาจากกฎหรือหลักการที่ได้จะใช้สำหรับเครื่องเท่านั้น หรืออาจจะมีปริมาณมากจนยากแก่การทำความเข้าใจหรือนำมาใช้ ซึ่งวิธีการนี้แม้ว่าเครื่องสามารถจะเรียนรู้และสามารถอธิบายข้อมูลได้ถูกต้องแต่ก็ไม่สามารถเข้าใจถึงพฤติกรรมการทำงานของ

วิธีการเรียนรู้ของเครื่องได้ ส่วนวิธีการแบบ white-box นั้น กฎหรือหลักการจะแสดงให้อยู่ในรูปของสัญลักษณ์หรือภาษาที่มนุษย์สามารถเข้าใจได้

นอกจากนี้ในการเรียนรู้ของเครื่องจากข้อมูลตัวอย่าง เรายังสามารถแบ่งการเรียนรู้กับการใช้ตัวอย่างข้อมูลที่ต่างชนิดกันได้ออกเป็น 3 แบบคือ

1.การเรียนรู้กับตัวอย่างข้อมูลที่เป็นทั้งตัวอย่างบวกและตัวอย่างลบในข้อมูลชนิดเดียวกัน

เช่น ตัวอย่างข้อมูลที่เป็นนกและตัวอย่างข้อมูลที่ไม่ใช่นก ซึ่งเราจะได้ความรู้ที่สามารถจำแนกตัวอย่างที่สามารถอธิบายตัวอย่างที่เป็นนกและตัวอย่างที่ไม่ใช่นกได้ เป็นต้น

2.การเรียนรู้กับตัวอย่างข้อมูลที่ต่างกันชนิดกันนำมาเรียนรู้ด้วยกัน เช่น ข้อมูลตัวอย่างของนก ดอกไม้ และ ผึ้ง ซึ่งเราจะได้ความรู้ที่สามารถจำแนกข้อมูลและสามารถอธิบายตัวอย่างที่เป็นนก ดอกไม้ และผึ้งได้ เป็นต้น

3.การเรียนรู้กับข้อมูลตัวอย่างบวกเพียงอย่างเดียว

จากการวิธีการเรียนรู้กับข้อมูลใน 2 แบบแรกนั้น สมมติฐานที่ได้จะเป็นการพยายามจำแนกข้อมูลที่ต่างชนิดกันหรือไม่เข้าพวกออกจากกัน ส่วนในการเรียนรู้กับตัวอย่างข้อมูลแบบสุดท้าย สมมติฐานที่ได้จะเป็นการเรียนรู้เพื่ออธิบายลักษณะของตัวอย่างแต่ละตัวในข้อมูลตัวอย่างเท่านั้น ซึ่งสมมติฐานที่ได้จากวิธีการเรียนรู้แบบที่ 3 นี้อาจจะไม่สามารถนำไปใช้ในการอธิบายข้อมูลตัวอย่างอื่นๆ ที่ไม่เคยเห็นได้

2.1.2 **วิธีการเรียนรู้โดยวิธีการโปรแกรมตรรกะเชิงอุปนัย**

วิธีการโปรแกรมตรรกะเชิงอุปนัย (Inductive Logic Programming) เป็นวิธีการเรียนรู้ของเครื่องแบบหนึ่งโดยใช้กฎลำดับที่หนึ่ง (first-order rules) ในการอธิบายตัวอย่างและกฎที่ได้จากการเรียนรู้ดังในตารางที่ 2.1 และใช้ความรู้ภูมิหลัง (B), เซตตัวอย่างบวก (E) ในการสร้างกฎที่ครอบคลุมตัวอย่างบวก (E^+) และไม่ครอบคลุมตัวอย่างลบ (E^-)

ตารางที่ 2.1

แสดงรูปแบบของการแปลงข้อมูลตัวอย่างเป็นรูปแบบกฎลำดับที่ 1

รุ่น	ระยะทาง	ราคา	y/n	รูปแบบกฎลำดับที่ 1
BMW Z3	50,000	5000	+	car(bmw_z3,50000,5000).
Audi V8	30,000	4000	+	car(audi_v8,30000,4000).
Fiat Uno	90,000	3000	-	:- car(fiat_uno,90000,3000).

วิธีการในการสร้างสมมติฐานโดยใช้วิธีการตรรกะเชิงอุปนัยจะเริ่มต้นด้วยการมีเซตตัวอย่างบวกและเซตตัวอย่างลบ E^+, E^- และความรู้ภูมิหลังที่อยู่ในรูปแบบของกฎลำดับที่ 1 การที่จะพยายามหาสมมติฐาน H ที่สามารถเรียนรู้ตัวอย่าง E^+, E^- ได้ โดยเมื่อใดก็ตามที่สามารถพิสูจน์ได้ว่าตัวอย่างบวก E^+ และตัวอย่างลบ E^- สามารถเป็นจริงได้ด้วยสมมติฐาน H ก็จะแสดงว่าสมมติฐาน H ที่ได้สามารถอธิบายตัวอย่างบวก E^+ และตัวอย่างลบ E^- ได้

เพื่อให้แน่ใจว่าเราสามารถหาคำตอบให้กับปัญหาที่จะทำการเรียนรู้ได้เราจำเป็นต้องตรวจสอบเงื่อนไขที่เรียกว่า Prior conditions โดยที่ผู้ใช้ต้องตรวสอบก่อนว่าถ้าด้วยความรู้ภูมิหลัง B สามารถทำให้ตัวอย่างลบ E^- ตัวใดตัวหนึ่งเป็นจริงได้แสดงว่าปัญหาที่เรียนรู้นี้อาจจะมีข้อมูลที่ไม่มีความแน่นอน ซึ่งทำให้สมมติฐานที่สร้างจากความรู้ภูมิหลังนี้ก็จะทำให้ตัวอย่างลบเป็นจริงอยู่เสมอหากใช้ความรู้ภูมิหลังนี้จำเป็นต้องทำการตรวจสอบ prior satisfiability

$$\forall e \text{ in } E^-(B \not\models e)$$

นอกจากนี้ต้องตรวจสอบว่าหากความรู้ภูมิสามารถทำให้ตัวอย่างบวก E^+ ที่มีทั้งหมดถูกต้องอยู่แล้วก็ไม่จำเป็นต้องทำการเรียนรู้ต่อไป ซึ่งจะขาดในเงื่อนไขของ prior necessity

$$\exists e \text{ in } E^+(B \not\models e)$$

เมื่อผ่านเงื่อนไข Prior conditions แล้วก็มาทำการทดสอบในเงื่อนไขที่เรียกว่า Posterior conditions ซึ่งในเงื่อนไขนี้จะเป็นการทดสอบสมมติฐานกับความรู้ภูมิหลังกับตัวอย่างทั้งบวกและ

ลบ ถ้าสมมติฐานที่ได้กับความรู้ภูมิหลังนำมาใช้ทดสอบกับตัวอย่างลบแล้วไม่รวมตัวอย่างลบใดๆ เข้ามาเลยจะถือว่าเป็นสมมติฐานที่ได้ผ่าน posterior satisfiability

$$\forall e \text{ in } E^{-}((B \wedge H) \not\models e)$$

และถ้าสมมติฐานที่ได้กับความรู้ภูมิหลังนำมาใช้ทดสอบกับตัวอย่างบวกแล้วสามารถรวมตัวอย่างบวกทั้งหมดเข้ามาได้จะถือว่าเป็นสมมติฐานที่ได้ผ่าน Posterior sufficiency

$$\forall e \text{ in } E^{+}((B \wedge H) \models e)$$

ซึ่งถ้าปัญหาที่จะทำการเรียนรู้ใดสามารถผ่านเงื่อนไข Posterior conditions ได้แล้วจะถูกว่าสมมติฐานที่ได้มีความสมบูรณ์และน่าเชื่อถือสูง

อย่างไรก็ดีวิธีการโปรแกรมตรรกะเชิงอุปนัยจำเป็นต้องสร้างและทดสอบสมมติฐานกับตัวอย่างมากมายถ้าจะทำการทดสอบทุกทางเลือกที่เป็นไปได้ก็อาจจะไม่เหมาะสมในการนำไปปฏิบัติ วิธีการที่ใช้ในการค้นหาในวิธีการโปรแกรมตรรกะเชิงอุปนัยจึงมีส่วนสำคัญในการเลือกและค้นหาสมติฐานที่เหมาะสม โดย (S. Muggleton, L. De Raedt, 1994) กำหนดนิยามของการทำการอุปนัย (induction) และการนิรนัย (deduction) ไว้คือ

- สมมติฐาน G จะมีการวางนัยโดยทั่วไป (general) มากกว่าสมมติฐาน S เมื่อ G $\models$ S และสามารถกล่าวได้ว่าสมมติฐาน S มีความเฉพาะเจาะจง (specific) มากกว่าสมมติฐาน G

- การนิรนัยของการอนุมานกฎ(deductive inference rule) r ซึ่งทาบกลุ่มรวมของอนุประโยคในสมมติฐาน G ลงบนกลุ่มรวมของอนุประโยคในสมมติฐาน S เพื่อให้ได้ G $\models$ S จะเรียก r ว่าเป็นกฎที่ทำให้เฉพาะเจาะจงมากขึ้น

- การอุปนัยของการอนุมานกฎ(inductive inference rule) r ซึ่งทาบกลุ่มรวมของอนุประโยคในสมมติฐาน S ลงบนกลุ่มรวมของอนุประโยคในสมมติฐาน G เพื่อให้ได้ G $\models$ S จะเรียก r ว่าเป็นกฎที่ทำให้มีความเป็นกลางมากขึ้น

ภายใต้การนิยามที่กำหนดจะเห็นได้ว่าในทุกๆ ระบบที่ใช้วิธีการโปรแกรมตรรกะเชิงอุปนัยจะใช้หลักการนี้ในการค้นหาสมมติฐาน เพียงแต่จะแตกต่างกันตรงที่การทำงานจะเป็นแบบการพยายามทำให้สมมติฐานมีความเฉพาะเจาะจงมากขึ้นหรือมีความเป็นกลางมากขึ้น

ในระบบที่พยายามทำให้สมมติฐานมีความเป็นกลางมากขึ้นนั้น ระบบจะสร้างสมมติฐานที่มีความเฉพาะเจาะจงที่สุดแล้วค่อยพยายามเพิ่มกฎที่ทำให้มีความเป็นกลางมากขึ้น โดยที่ยังคงรักษาสมมติฐานไม่ให้ครอบคลุมตัวอย่างลบเข้ามา ในทางกลับกันบางระบบที่ใช้การทำสมมติฐานให้มีความเฉพาะเจาะจงมากขึ้นนั้นจะทำการสร้างสมมติฐานที่ไม่มีอนุประโยคใดๆ อยู่เลยจากนั้นจึงพยายามเพิ่มกฎที่ทำให้สมมติฐานมีความเฉพาะเจาะจงมากขึ้นเข้าไป

ในการปรับแต่ง (Pruning) และจัดเรียงสมมติฐานที่ได้ (Sorting) โดยปกติสมมติฐานสร้างขึ้นเพื่อใช้อธิบายข้อมูลตัวอย่างบวกอาจะสามารถธิบายตัวอย่างลบบางตัวได้ด้วยซึ่ง โดยที่สมมติฐานมีความเป็นกลางมากกว่าก็จะสามารถอธิบายตัวอย่างลบได้มากกว่าสมมติฐานที่มีความเฉพาะเจาะจง เพื่อให้สามารถแก้ไขตัวอย่างลบในสมมติฐาน วิธีการโปรแกรมตรรกะเชิงอุปนัยจะมีวิธีการในการทำงานคือ

1.กำหนดเซตของสมมติฐานที่มีให้เป็น QH

2.ในขณะที่กำลังทำการค้นหา สมมติฐาน Q จะถูกเลือกออกมาจากเซตของสมมติฐาน QH มาทำการปรับแต่งโดยเพิ่มการอนุมานกฎ (inference rule) เพื่อสร้างสมมติฐานใหม่ จากนั้นทำการเพิ่มสมมติฐานใหม่ที่ได้ลงในเซตของสมมติฐาน

3.ทำซ้ำในข้อ 2 จนครบตามเงื่อนไขจึงหยุดการทำงาน

โดยการเลือกสมมติฐานใดๆ ในเซตของสมมติฐานมาทำการปรับแต่งนั้นจะดูจากค่าความน่าจะเป็นที่สมมติฐานจะสามารถครอบคลุมความรู้ภูมิหลังและตัวอย่างที่ถูกต้องซึ่งจะถูกคำนวณขึ้นสำหรับแต่ละสมมติฐานในเซตโดยสมการที่ได้จากของเบย์ (Bayesian) จากค่าความน่าจะเป็นที่ได้สมมติฐานที่มีความน่าจะเป็นสูงจะถูกเลือกนำมาปรับปรุงก่อนแล้วค่อยทำกับสมมติฐานอื่นที่มีค่าความน่าจะเป็นสูงรองลงมาเรื่อยๆ เป็นลำดับถัดไป

ในระบบที่ทำให้สมมติฐานมีความเป็นกลางมากขึ้นนั้นเป็นการยากที่จะพยายามแก้ไขไม่ให้สมมติฐานใหม่ที่สร้างไม่อธิบายตัวอย่างลบ เนื่องจากระบบทำการเลือกสมมติฐานเดิมมีที่ความเฉพาะเจาะจงอยู่แล้วมาทำการปรับแต่งให้มีความเป็นกลางมากขึ้น สมมติฐานที่ได้ก็จะสามารถอธิบายตัวอย่างบวกได้มากขึ้น ไม่มีผลกับการแก้ไขหรือลดตัวอย่างลบที่ถูกอธิบายได้ในสมมติฐานเดิมได้ ส่วนระบบที่ทำให้สมมติฐานมีความเฉพาะเจาะจงมากขึ้นนั้น ระบบมีความสามารถที่จะแก้ไขตัวอย่างลบที่ถูกอธิบายในสมมติฐานเดิมได้ อันเนื่องมาจากการทำให้

เฉพาะเจาะจงมากขึ้นนั้น สมมติฐานที่ได้จะสามารถอธิบายตัวอย่างบวกได้น้อยลงซึ่งรวมทั้งตัวอย่างลบที่สมมติฐานเดิมออาจจะอธิบายไว้ก็จะถูกตัดทอนออกไปได้ด้วย ทำให้ระบบนี้สามารถทำงานกับเซตข้อมูลที่มีข้อมูลสัญญาณรบกวน (Noisy data) ได้ดี นอกจากนี้ระบบยังมีความยืดหยุ่นในเรื่องของการปรับปรุงสมมติให้สามารถสอดคล้องกับเงื่อนไขหลัง (Posterior conditions) ที่ผู้ใช้อาจจะสามารถกำหนดให้สมมติฐานสามารถอธิบายตัวอย่างลบได้บ้าง แทนที่จะเลือกตัดสมมติฐานข้อนั้นทิ้งไป

ในส่วนของการจัดเรียงสมมติฐานในเซตของสมมติฐาน เราสามารถจะใช้ข้อมูลตัวอย่างเป็นตัวช่วยในการตัดสินใจ โดยในระบบที่ทำให้สมมติฐานมีความเป็นกลางมากขึ้น เราจะจัดเรียงสมมติฐานตามจำนวนตัวอย่างบวกที่สมมติฐานแต่ละตัวนั้นสามารถอธิบายได้ นั่นคือสมมติฐานที่อธิบายตัวอย่างบวกได้มากที่สุดจะเป็นสมมติฐานที่ดีที่สุด ส่วนในระบบที่ทำให้สมมติฐานมีความเฉพาะเจาะจงมากขึ้นนั้น เราจะสนใจในตัวอย่างลบที่แต่ละสมมติฐานสามารถอธิบายได้ โดยที่สมมติฐานที่สามารถอธิบายตัวอย่างลบได้น้อยที่สุดจะเป็นสมมติดีที่สุดสำหรับระบบนี้

นอกเหนือจากลดเวลาการทำงานของวิธีการโปรแกรมเชิงตรรกะ ด้วยวิธีการลดขนาดปริภูมิข้อมูลที่วิธีการโปรแกรมเชิงตรรกะจะต้องค้นหาแล้ว เรามีวิธีการที่เรียกว่า ข้อจำกัดของภาษา (Language restrictions) ซึ่งก็คือ การกำหนดรูปแบบหรือจำนวนของอนุประโยคในสมมติฐานที่ทำการเรียนรู้ สามารถทำได้หลายๆ วิธี เช่น กำหนดจำนวนตัวแปรที่ใช้ในการสร้างอนุประโยค, กำหนดลักษณะของส่วนหัว (head) และส่วนของตัวกฎ (body) โดยระบุค่าคงที่, กำหนดจำนวนอนุประโยคในสมมติฐาน เป็นต้น ซึ่งวิธีเหล่านี้ช่วยเพิ่มความเร็วให้กับการทำงานมากขึ้น แต่ผู้ใช้จำเป็นต้องมีความรู้และปรับเปลี่ยนให้เหมาะสมกับข้อมูลตัวอย่างที่นำมาเรียนรู้ด้วย

2.1.3 **วิธีการทดสอบความถูกต้องแบบไขว้ข้าม**

วิธีการในการทดสอบความถูกต้องแบบไขว้ข้าม (Cross-Validation) เป็นวิธีการหนึ่งที่ใช้ในการประมาณค่าความผิดพลาดในการกฎหรือวิธีการในการเรียนรู้ของเครื่อง หลักการของวิธีการนี้คือการทดสอบความถูกต้องแบบไขว้ข้ามโดยการใช้กลุ่มข้อมูลสำหรับการทดสอบซึ่งได้มาจากกลุ่มข้อมูลที่ใช้ในการทดลอง โดยแยกกลุ่มข้อมูลสำหรับการทดสอบ ออกจากกลุ่มข้อมูลที่ใช้สำหรับการเรียนรู้ และจะไม่นำกลุ่มข้อมูลสำหรับทดสอบมาใช้จนกว่าการเรียนรู้ทั้งหมดจะเสร็จสิ้น ซึ่งประสิทธิภาพที่ได้จากการทดสอบเป็นการประมาณค่าผิดพลาดที่ได้ของกฎหรือวิธีการนั้นๆ โดย

วิธีการทดสอบความถูกต้องแบบไขว้ข้ามที่เป็นที่นิยมวิธีการหนึ่งคือ วิธีทดสอบความถูกต้องแบบไขว้ข้าม K ชุดข้อมูลย่อย (K-fold Cross-Validation)

แนวคิดในการทำงานของการทดสอบความถูกต้องแบบไขว้ข้าม K ชุดข้อมูลย่อย โดยการทำซ้ำการทดสอบความถูกต้องโดยใช้เวลาเท่ากับ K ซึ่งทำการแบ่งข้อมูลในการทดลองออกเป็น K ชุดข้อมูลย่อย (จะใช้แทนด้วย D) และในแต่ละชุดข้อมูลย่อยจะประกอบด้วยข้อมูลตัวอย่างที่ได้จากข้อมูลที่กระจายกัน

ขั้นตอนของการทำงาน ดังนี้

1. ทำซ้ำจนถึงเวลา K (จนกว่าข้อมูลในชุดข้อมูลย่อยทุกตัวจะถูกใช้สำหรับการทดสอบ)

1.1) กำหนดชุดข้อมูลย่อยที่ใช้ในการทดสอบ 1 ชุดข้อมูลย่อย D_k จากชุดข้อมูลย่อยที่ใช้ในการทดลอง และใช้ชุดข้อมูลย่อยที่เหลือ $D_{i\neq K}$ (โดยไม่รวมชุดข้อมูลย่อยที่ใช้ทดสอบ) เป็นชุดข้อมูลสำหรับสอน

1.2) สอนเน็ตเวิร์กด้วยชุดข้อมูลเรียนรู้

1.3) ทดสอบเน็ตเวิร์กด้วยชุดข้อมูลย่อยที่ใช้ทดสอบ D_k ซึ่งได้ผลการทดสอบเป็นค่าผิดพลาด Etest, K (ค่าผิดพลาดจากการทดสอบ test ด้วยชุดข้อมูลย่อยตัวที่ k)

2. หาค่าผิดพลาดเฉลี่ยจากการประมาณค่าผิดพลาดที่เน็ตเวิร์กได้ให้คำตอบของการทดสอบ K ชุดข้อมูลย่อยสมการหาค่าผิดพลาดเฉลี่ย

$$\text{Err} = \frac{1}{K}\sum_{i=1}^{K} E_{test,K}$$

ในการใช้งานทั่วไปนักวิจัยได้ใช้การทดสอบความถูกต้องแบบไขว้ข้าม โดยใช้จำนวน K ชุดข้อมูลย่อยเท่ากับ 5 หรือ 10 ชุดข้อมูลย่อย ซึ่งจากการค้นคว้าเกี่ยวกับงานวิจัยที่ได้นำวิธีการทดสอบความถูกต้องแบบไขว้ข้ามมาใช้ พบว่า ในการใช้จำนวน K ชุดข้อมูลย่อยเท่ากับ 10 หรือเรียกว่า การทดสอบความถูกต้องแบบไขว้ข้าม 10 ชุดข้อมูลย่อย ซึ่งเป็นเทคนิคที่มีประสิทธิภาพที่ใช้กับการทดสอบความถูกต้อง จากการนำเสนอโดย Weiss และ Kulikowski (1991) และได้มีงานวิจัยหลายงานวิจัยที่ได้ให้การยอมรับ และใช้วิธีการทดสอบความถูกต้องแบบไขว้ข้ามจำนวน 10 ชุดข้อมูลย่อย เป็นวิธีการทดสอบมาตรฐานในการทดสอบความถูกต้องทางด้านการเรียนรู้ของ

เครื่อง (Smith, Nugent and McClean, 2003; Claudia, Dorffner and Lee, 1998; George, 1994; Belacel and Boulassel, 2001; Cunningham, Carney and Jacob, 2000; Sinthupinyo, Pipanmaekaporn and Kijsirikul, 2003; Anderson and Martinez, 1999; Wlodzislaw and Krzysztof, 1999)

2.2 งานวิจัยที่เกี่ยวข้อง

ในส่วนของผลงานวิจัยที่เกี่ยวข้องจะประกอบด้วยการศึกษางานวิจัยที่ใช้ในการเพิ่มความเร็วในการเรียนรู้กับข้อมูล 2 งานวิจัยและวิธีการเพิ่มความถูกต้องแม่นให้กับข้อมูลอีก 4 งานวิจัยคือ

Fonseca, Silva และ Camacho (2005) ได้ศึกษาและเปรียบเทียบงานวิจัยที่ทำโปรแกรมตรรกะแบบอุปนัยในรูปแบบของการประมวลผลแบบขนาน เพื่อหาวิธีการที่ดีที่สุด จากวิธีการใช้การประมวลผลแบบขนานกับโปรแกรมตรรกะเชิงอุปนัย 3 วิธี คือ พีซีที (parallel coverage tests: PCT), ดีพีแอลอาร์ (data parallel learn rule: DPLR) และ ดีพีไอแอลพี (data parallel ILP: DPILP) โดยทำการเปรียบเทียบและวัดผลในเรื่องของการใช้จำนวนหน่วยประมวลผลและประเภทของหน่วยความจำ (ใช้หน่วยความจำร่วมกันหรือแยกหน่วยความจำ) เพื่อให้ได้ผลลัพธ์ที่ดีและใช้เวลาน้อยที่สุด

ผลที่ได้คือในการหาจำนวนหน่วยประมวลผลที่เหมาะสม พบว่ากับวิธีการที่ใช้และเซตข้อมูลที่นำมาทดสอบนั้นยิ่งใช้จำนวนหน่วยประมวลผลจำนวนมาก (ทดสอบที่ 1, 2, 4 และ 8) ก็ยิ่งสามารถทำให้การเรียนรู้เสร็จได้รวดเร็วมากขึ้น ส่วนในเรื่องของการใช้หน่วยความจำที่เหมาะสมนั้นพบว่าการให้แต่ละหน่วยประมวลผลใช้หน่วยความจำร่วมกันสามารถทำงานได้รวดเร็วมากกว่าการใช้หน่วยความจำแยกอันเนื่องมาจากไม่เสียเวลาในการรับและส่งข้อมูลระหว่างการทำงานและวิธีการที่เหมาะสมในการทำการประมวลผลแบบขนาน กับโปรแกรมตรรกะเชิงอุปนัยที่ดีที่สุดคือ ดีพีไอแอลพี (DPILP) ซึ่งทำการแบ่งเซตข้อมูลตัวอย่างออกเป็นส่วนย่อย จากนั้นกันนำไปเรียนรู้แล้วนำกฎที่เรียนได้มารวมกัน จะเป็นวิธีการที่ให้ผลลัพธ์ได้รวดเร็วและได้ความถูกต้องมากที่สุด ซึ่งวิธีในการรวมกฎที่ได้จากหน่วยประมวลผลแต่ละตัวนั้น ดีพีไอแอลพี (DPILP) จะใช้วิธีการเลือกกฎโดยเรียงลำดับตามความถูกต้องและครอบคลุมของกฎ จากนั้นจึงเลือกกฎที่ดีที่สุดในขณะนั้นเข้ามารวม แล้วทำการประเมินผลกฎที่เหลือเพื่อทำการเลือกต่อไป อย่างไรก็ดีผู้เขียนไม่ได้ให้รายละเอียดในการทำงานขั้นตอนนี้

Dutra, Page, Costa และ Shavlik (2000) เสนอวิธีการทำในการทำเอนเซมเบิลกับโปรแกรมตรรกะแบบอุปนัยโดยใช้วิธีการที่เรียกว่าแบ็กกิง (Bagging) ซึ่งวิธีการนี้จะทำการเลือกข้อมูลโดยใช้การสุ่มเลือกข้อมูลจากตัวอย่างทั้งหมดมาส่วนหนึ่งจากนั้นก็ทำการสร้างตัวอย่างที่ซ้ำกับตัวอย่างที่มีในกลุ่มนั้นๆ มาเปรียบเทียบกับวิธีการสร้างกลุ่มตัวอย่างด้วยวิธีการสุ่มเลือกตัวอย่างตามปกติ จากนั้นจึงนำมาเรียนรู้เพื่อให้ได้กฎโดยใช้ aleph (A. Srinivasan, 2001)เมื่อได้กฎมาแล้วจะนำมารวมกันด้วยใช้ค่าสมมติเป็นตัวกำหนดว่ากฎไหนสามารถนำมารวมได้ ซึ่งผลจากการวิจัยพบว่าความถูกต้องของการสุ่มเลือกตัวอย่างตามปกติมีความถูกต้องมากกว่าอันเนื่องมาจากกฎที่เรียนรู้มาจากตัวอย่างทั้งหมดขณะที่วิธีการแบ็กกิง นั้นไม่ได้รวมตัวอย่างทั้งหมดมาทำให้โอกาสในการได้กฎที่ครบถ้วนมีน้อยกว่า

Prati และ Flach (2005) เสนอวิธีการในการเลือกและลดจำนวนกฎที่ได้จากกลุ่มของกฎจำนวนมากด้วยที่ผ่านกระบวนการ Appriori (F. Bodon, 2003) วิธีการที่เรียกว่า รอคเคอร์ (ROCCER) ซึ่งจะใช้จำนวนตัวอย่างบวกที่สามารถจำแนกได้ (tpr) และ จำนวนตัวอย่างบวกที่ไม่สามารถจำแนกได้ (fpr) มาวาดกราฟที่เรียกว่า ROC Curve ซึ่งจากงานวิจัยของ Furnkranz และ Flach (2001) ได้แสดงการดูความครอบคลุมกฎสามารถดูได้จากเส้นโค้ง (ROC Curve) ของกราฟ ROC โดยกฎๆ หนึ่งจะถูกแทนให้อยู่ในรูปแบบของคู่อันดับ (fpr,tpr) ใน Roc curve โดยวิธีการจะทำการกำหนดกฎตั้งต้นที่เอาไว้แบ่งแยกข้อมูลตัวอย่างบวกและลบเป็นค่า (0,0) และ (1,1) จากนั้นจะทำการเพิ่มคู่อันดับของกฎตัวที่ต้องการจะเลือกเข้าไปในกราฟ ซึ่งจะทำให้เกิดพื้นที่ในระบบแกน xy ซึ่งเรียกว่า Roc convex hull ซึ่งกฎต่อๆ ไปที่จะนำมาเพิ่มจะนำมาเทียบกับพื้นที่เดิมที่มีอยู่ ถ้ากฎใหม่ที่นำมาเพิ่ม มีคู่อันดับ (fpr,tpr) ที่อยู่นอกหรือเหนือกว่าพื้นที่ Roc convex hull กฎนั้นก็จะถูกเลือกเข้ามาด้วยและจะทำกับปรับเปรียบพื้นที่ Roc convex hull ใหม่ แต่ถ้ากฎใหม่ที่นำมาเพิ่มมีคู่อันดับที่นำมาจุดลงบนพื้นที่ Roc convex hull อยู่แล้วจะไม่นำกฎนั้นมาใช้ ซึ่งผู้วิจัยทำการเปรียบเทียบกับวิธีการในการสร้างกฎอีกหลายตัวเช่น slipper, ripper,C4.5 เป็นต้น พบว่าวิธีการนี้สามารถลดจำนวนกฎลงได้ดีกว่าทุกวิธียกเว้น ripper ในข้อมูลบางประเภท ส่วนในแง่ของความถูกต้องนั้นวิธีการนี้สามารถให้ค่าความถูกต้องที่ดียกเว้นข้อมูลที่นำมาเรียนมีความผิดเพี้ยนไม่สมบูรณ์ แต่วิธีการนี้ก็ใช้เวลาในการทำงานมากกว่าวิธีการอื่นๆ ที่ $O(n^2)$ เมื่อ n คือจำนวนของกฎ

Cohen และ Singer (1999) ได้เสนอวิธีในการสร้างกฎที่สามารถเข้าใจได้ง่ายและใช้เวลาในการทำงานเพื่อสร้างกฎไม่มาก วิธีนี้ชื่อว่า สลิปเปอร์ (SLIPPER) ซึ่งมีวิธีในการทำงานคือ เริ่มต้นแบ่งข้อมูลตัวอย่างออกเป็น 2 กลุ่ม คือ กลุ่มที่ใช้ในการสร้างกฎ และ กลุ่มที่ใช้ในการตัดแต่งกฎ

โดยที่กลุ่มแรกจะมีจำนวนเป็น 2/3 ของทั้งหมด จากนั้นจะเริ่มต้นสร้างกฎขึ้นโดยเลือกเงื่อนไขทีละเงื่อนไขแล้วมาทดสอบกับกลุ่มข้อมูลที่ใช้สร้างกฎและเก็บคะแนนของกฎซึ่งคำนวณมาจากตัวอย่างที่ครอบคลุมทั้งตัวอย่างบวกและลบสำหรับได้ไว้ จากนั้นทำการเพิ่มเงื่อนไขให้กฎไปเรื่อยๆ จนได้กฎที่ไม่ครอบคลุมตัวอย่างลบเลย หรือ ไม่สามารถเพิ่มค่าคะแนนของกฎได้ ซึ่งกฎที่ได้อาจจะมีความจะถูกนำมาทดสอบกับกลุ่มข้อมูลที่ใช้ในการตัดแต่งกฎเพื่อให้ได้กฎที่ไม่เฉพาะเจาะจงกับข้อมูลกลุ่มใดกลุ่มหนึ่งมากเกินไป โดยทำการตัดเงื่อนไขบางอันออกจากกฎและคำนวณหากฎที่ครอบคลุมตัวอย่างได้ดีที่สุด เมื่อได้กฎทั้งหมดมาแล้วจะทำขั้นตอนในการลดกฎที่เหมือนกันหรือครอบคลุมตัวอย่างคล้ายกันออกเพื่อให้จำนวนกฎที่น้อยลง ซึ่งในการวิจัยนี้ทดสอบกับข้อมูลตัวอย่างที่เป็นแบบปัญหาสองกลุ่ม (two-type classification)ทั้งหมด ซึ่งผลที่ได้เมื่อเทียบกับวิธีการสร้างกฎอื่นๆ คือ RIPPER, C4.5 และ C5 แล้วพบว่า วิธีการนี้สามารถสร้างกฎที่ให้ความถูกต้องได้มากกว่าโดยใช้เวลาในการทำงานมากที่สุดไม่เกิน $O(n\log n)$ เมื่อ n คือจำนวนตัวอย่างที่ใช้ทั้งหมด

Verbaeten และ Assche (2003) เสนอวิธีในการลดข้อมูลที่มีข้อมูลผิดพลาดอยู่ในตัวอย่าง (noise) ก่อนที่จะนำไปทำไปใช้ในการเรียนรู้ โดยเสนอวิธีในการกรองข้อมูล 2 แบบคือ การกรองข้อมูลด้วยวิธีการโหวตและการใช้ค่าน้ำหนักจากวิธีการบูสต์ (Boosting) ซึ่งที่วิธีการกรองโดยการโหวตจะใช้วิธีการ 2 วิธีการคือการทำ cross-validated committee's ซึ่งมีหลักการคือ จะแบ่งข้อมูลตัวอย่างจำนวนเท่าๆ กันออกเป็นทั้งหมด n กลุ่มจากนั้นจะทำการเรียนทั้งหมด n ครั้ง โดยแต่ละครั้งจะเว้นไม่นำตัวอย่างกลุ่มหนึ่งมาเรียนรู้ด้วย วิธีที่ 2 คือการใช้วิธีการแบ็กกิง ให้กับตัวอย่าง โดยการสุ่มเลือกตัวอย่างออกมาเป็นกลุ่มจากนั้นจะเพิ่มจำนวนตัวอย่างให้ซ้ำกับข้อมูลตัวอย่างเดิมที่มีในกลุ่ม ส่วนวิธีการกรองข้อมูลโดยใช้วิธีการบูสต์นั้น เนื่องจากวิธีการบูสต์ หากนำตัวอย่างข้อมูลที่มีข้อมูลผิดพลาดอยู่ในตัวอย่างมาทำการเรียนรู้จะทำให้ค่าน้ำหนักที่สูง ซึ่งทำให้เราสามารถนำมาใช้ในการหาข้อมูลตัวอย่างที่มีข้อมูลผิดพลาดได้ โดยการทดลองนี้จะใช้ข้อมูลตัวอย่างที่ไม่มีข้อมูลผิดพลาดในตัวอย่างเลย และทำการเพิ่มข้อมูลที่ผิดพลาดลงในตัวอย่างโดยการสร้างกลับตัวอย่างบวกให้เป็นตัวอย่างลบและทำกลับกัน ซึ่งผลลัพธ์ที่ได้จะพบว่าวิธีการกรองข้อมูลโดยการโหวตด้วยการทำ cross-validated committee's และ วิธีการแบ็กกิง สามารถกรองข้อมูลที่ผิดพลาดออกจากตัวอย่างได้ดีขณะที่วิธีการกรองโดยดูจากค่าน้ำหนักของวิธีการบูสต์กลับไม่ได้ผลดีนักเนื่องจากค่าน้ำหนักที่คำนวณได้มีค่าใกล้เคียงกันทำให้วิธีการนี้เลือกกรองข้อมูลผิดพลาด

Sebag และ Rouveirol (1997) เสนอวิธีในการลดเวลาในการประมวลของวิธีการโปรแกรมตรรกะเชิงอุปนัย โดยใช้การเลือกเฉพาะตัวอย่างที่เหมาะสมกับสมมติฐานเพื่อนำมาใช้ในการเรียนรู้เท่านั้นแทนที่จะต้องทำการเปรียบเทียบตัวอย่างทั้งหมดกับสมมติฐานที่มีทั้งหมด ซึ่งด้วยวิธีการนี้ จะทำให้เวลาที่ใช้ประมวลผลจากเดิมที่เป็น $O(e^n)$ ลดเหลือประมาณ $O(n^2)$

วิธีการที่ใช้นี้เรียกว่า STILL (stochastic inductive learning) โดย STILL เกิดจากการใช้ DiVS (Disjunctive Version Space framework) ซึ่งจะทำการสร้างกลุ่มของสมมติฐานที่ครอบคลุมตัวอย่างอย่างน้อย 1 ตัวและทำการสร้างตัวแทนของตัวอย่างสำหรับสมมติฐานแต่ละตัว จากนั้นใช้วิธีการที่เรียกว่า Stochastic ซึ่งจะเลือกตัวแทนของตัวอย่างที่ได้จาก DiVS โดยการเลือกตัวแทนที่มีความสอดคล้องกับตัวแปรที่อยู่ในอนุประโยคที่ได้ในแต่ละสมมติฐานตามจำนวนที่ผู้ใช้สามารถกำหนดได้ ซึ่งทำให้จำนวนตัวอย่างที่จะนำไปทำการเรียนรู้ลดลง นอกจากนี้ผู้วิจัยยังเพิ่มวิธีการเลือกตัวอย่างโดยอาศัยความรู้ของผู้วิจัยเองในการเลือกตัวอย่างมาทำการเรียนรู้ด้วย ผลการทดลองพบว่าการวิธีการนี้สามารถลดเวลาในการเรียนรู้และมีความแม่นยำใกล้เคียงกับวิธีการเรียนรู้แบบอื่นๆ แต่วิธีการนี้จะได้ผลลัพธ์เป็นสมมติฐานจำนวนมากอาจจะยากต่อการทำความเข้าใจและผู้ใช้จำเป็นต้องรู้ค่าที่เหมาะสมในการทำ Stochastic เพื่อให้ได้ผลลัพธ์ที่ดี

จากการศึกษางานวิจัยทั้ง 6 พบว่าจากงานวิจัย ของ Fonseca, Silva และ Camacho (2005) การแบ่งตัวอย่างข้อมูลเพื่อนำมาเรียนรู้ในการโปรแกรมตรรกะเชิงอุปนัยจะสามารถทำให้การเรียนรู้สามารถทำได้เสร็จได้รวดเร็วขึ้น ซึ่งวิธีการที่เหมาะสมที่จะใช้ในการทำก็คือแบ่งข้อมูลตัวอย่างออกเป็นกลุ่มๆ จากนั้นแยกกันนำไปเรียนรู้ เมื่อได้กฎจากการเรียนรู้ก็นำมารวมกันโดยในดูค่าความครอบคลุมตัวอย่างของกฎเป็นหลักซึ่งเราสามารถหาวิธีเพื่อเพิ่มความถูกต้องของกฎให้มากขึ้นได้ เช่นนำวิธีในงานวิจัย ของ Prati, และ Flach (2005) มาใช้ร่วมกัน อาจทำให้กฎมีความถูกต้องมากขึ้นได้

ส่วนในงานวิจัยของ Dutra, Page, Costa และ Shavlik (2000) และงานวิจัยของ Verbaeten และ Assche (2003) จะเน้นในเรื่องของการเพิ่มความถูกต้องแม่นยำให้กับกฎโดยการเลือกข้อมูลก่อนที่จะนำไปเรียนและสร้างกฎจากการเรียนรู้ด้วยวิธีการโปรแกรมตรรกะเชิงอุปนัย โดยงานวิจัยของ Dutra, Page, Costa และ Shavlik (2000) นั้นเปรียบเทียบกับการสุ่มเปลี่ยนลำดับข้อมูลกับการทำแบ็กกิง ก่อนนำมาเรียนรู้ ซึ่งเห็นได้ว่าวิธีการทำแบ็กกิง อาจทำให้ข้อมูลตัวอย่างที่นำมาเรียนขาดหายไปทำให้ค่าความถูกต้องลดลงแต่หากข้อมูลตัวอย่างที่นำมาทดสอบมีข้อมูลที่มีข้อมูลผิดพลาดอยู่ในตัวอย่างมากๆ วิธีการแบ็กกิง อาจจะให้ผลลัพธ์ที่ดีกว่าแบบสุ่ม

เปลี่ยนลำดับได้ เนื่องมาจากงานวิจัยของ Verbaeten และ Assche (2003) แสดงให้เห็นว่าการใช้แบ็กกิงในการกรองข้อมูลก่อนจะสามารถลดข้อมูลผิดพลาดในตัวอย่างได้

ส่วนในงานวิจัยของ Cohen และ Singer (1999) แสดงให้เห็นถึงวิธีการทำโปรแกรมตรรกะเชิงอุปนัยแบบเอนเซมเบิลที่มีการแบ่งข้อมูลเป็นส่วนๆ สำหรับการสร้างกฎซึ่งสามารถสร้างกฎที่มีความถูกต้องได้ดีและวิธีการทำงานไม่ซับซ้อนแต่ถ้าหากขนาดปริภูมิของข้อมูที่ทำการเรียนรู้มีขนาดใหญ่ก็น่าจะใช้เวลาในการประมวลผลที่นานด้วย

สุดท้ายงานวิจัยของ Sebag และ Rouveirol (1997) สามารถลดเวลาในการประมวลผลโดยการเรียนรู้ด้วยวิธีการโปรแกรมตรรกะเชิงอุปนัยลงได้มากโดยที่ยังคงความถูกต้องแม่นยำได้อีกด้วย แต่ผลลัพธ์หรือสมมติฐานที่ได้จากการประมวลผลจะมีจำนวนมากทำให้ยากต่อการทำความเข้าใจหรือนำไปใช้และยากในการทำงานเนื่องจากต้องกำหนดค่าให้เหมาะสมกับการทำงานในการเลือกสุ่มตัวอย่างเพื่อนำมาใช้ในการเรียนรู้