# บทที่ 2

# ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

## 2.1 ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีที่เกี่ยวข้องในวิทยานิพนธ์ฉบับนี้ครอบคลุมเนื้อหาเกี่ยวกับการเรียนรู้ของเครื่อง (Machine Learning) ได้แก่

1. ประเภทของการเรียนรู้ ประกอบไปด้วย การเรียนรู้แบบเกียจคร้าน และ การเรียนรู้แบบขยัน
2. โครงสร้างการนิยามและวิธีการทำงานของโปรแกรมตรรกะเชิงอุปนัย ตลอดจนผลลัพธ์ที่ได้จากการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัย
3. ลักษณะและวิธีการทำงานของการรวมเชื่อมโยงกฎ และผลลัพธ์ที่ได้จากการใช้การรวมเชื่อมโยงกฎ
4. นิวรอลเน็ตเวิร์ก ประกอบด้วย โครงสร้างของนิวรอลเน็ตเวิร์ก, อิทธิพลของค่าตัวแปรที่ส่งผลกระทบต่อการทำงานของนิวรอลเน็ตเวิร์ก

### 2.1.1 ประเภทของการเรียนรู้

การเรียนรู้ของเครื่องโดยทั่วไปแบ่งออกได้เป็นสองลักษณะคือ การเรียนรู้แบบขยัน *(Eager Learning)* และ การเรียนรู้แบบเกียจคร้าน *(Lazy Learning)*

<u>การเรียนรู้แบบขยัน</u> คือลักษณะของการเรียนรู้ที่มีการจัดเตรียมหรือทำการเรียนรู้เสร็จเรียบร้อยรู้ไว้ก่อนล่วงหน้า ตัวอย่างของวิธีการเรียนรู้แบบขยัน เช่น นิวรอลเน็ตเวิร์ก เป็นต้น

<u>การเรียนรู้แบบเกียจคร้าน</u> คือลักษณะของการเรียนรู้ที่ไม่มีการประมวลผล การเรียนรู้หรือจัดเตรียมข้อมูลบางอย่างไว้ก่อน จะเริ่มเรียนรู้หรือทำงานเมื่อมีความต้องการผลลัพธ์เท่านั้น เพื่อหลีกเลี่ยงการประมวลผลส่วนเกินที่จะทำให้เกิดการเสียเวลาในการเรียนรู้อย่างยาวนาน ตัวอย่างการเรียนรู้แบบเกียจคร้าน

งานวิจัย Lazy Learning of Bayesian Rules ของ (Zijian และ Geoffery, 2000) เสนอวิธีการทำการเรียนรู้แบบเกียจคร้านของการเรียนรู้แบบเบย์ ด้วยวิธีการลดจำนวนข้อมูลที่จำเป็นจะต้องสร้างขึ้นในระหว่างขั้นตอนการเรียนรู้ โดยการแบ่งชุดข้อมูลที่ใช้ในการเรียนรู้ออกเป็น

ส่วนๆ แล้วเรียนรู้แบบเบย์ทีละส่วน จากนั้นจึงค่อยนำผลลัพธ์ที่เป็นกฎมาเชื่อมต่อกัน ผลที่ได้จากงานวิจัยนี้คือสามารถลดเวลาการเรียนรู้แบบเบย์ได้

งานวิจัย Lazy Decision Tree (Jerome และ Ron, 1996) เสนอวิธีการสร้างต้นไม้ตัดสินใจแบบเกียจคร้าน โดยแนวความคิดที่จะสร้างต้นไม้ตัดสินใจเท่าที่จำเป็นในเวลาที่มีความต้องการที่จำแนกตัวอย่างข้อมูล โดยการสร้างต้นไม้ตัดสินใจ จากชุดข้อมูลเซตย่อย (Sub set) มาจากข้อมูลที่ต้องเรียนรู้ทั้งหมด เมื่อจะจำแนกข้อมูลจะใช้การลงคะแนนเสียงของแต่ละต้นไม้ย่อยเป็นตัวตัดสินใจจำแนกข้อมูล ผลที่ได้สามารถลดเวลาการจำแนกข้อมูลที่ต้องเสียเวลาในการสร้างต้นไม้ตัดสินใจจากข้อมูลทั้งหมดลงได้

### 2.1.2 **โปรแกรมตรรกะเชิงอุปนัย** (Inductive Logic Programming)

วิธีการโปรแกรมตรรกะเชิงอุปนัยเป็นวิธีการเรียนรู้ของเครื่องแบบหนึ่งโดยใช้กฎลำดับที่หนึ่ง (first-order rules) ในการอธิบายตัวอย่าง โดยอาศัยการค้นหากฎที่ดีที่สุดจากกฎทั้งหมดที่สามารถอธิบายตัวอย่างที่ต้องการได้อย่างเหมาะสม คือจะต้องอธิบายตัวอย่างที่ต้องการและ กฎเหล่านั้นจะไม่ไปอธิบายหรือครอบคลุมตัวอย่างที่ไม่ต้องการหรือตัวอย่างที่มีลักษณะที่ผิด ข้อมูลที่ใช้ในการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยประกอบด้วย

- กลุ่มของข้อมูลที่ทำการทดสอบแล้วโดยต้องประกอบด้วย ตัวอย่างที่ทำการทดสอบที่ได้ผลเป็นบวก ($E^+$) และตัวอย่างทดสอบที่ได้ผลเป็นลบ($E^-$)
- ความรู้ภูมิหลังของตัวอย่าง(Background Knowledge) (B)
- สมมติฐานซึ่งอธิบายด้วยภาษาของโปรแกรมตรรกะเชิงอุปนัย

ตัวอย่างของการนิยามข้อมูลและความสัมพันธ์

daughter(mary,ann). มีความหมายว่า mary เป็นลูกสาวของ ann

mother(ann,mary). มีความหมายว่า ann เป็นแม่ของ mary

สมการของการเรียนรู้โปรแกรมตรรกะเชิงอุปนัย

$\forall$ e in E+ ((B $\wedge$ H) $\models$ e). สมการที่(1)

$\forall$ e in E- ((B $\wedge$ H) $\not\models$ e). สมการที่ (2)

จากสมการการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัย จะอธิบายได้ว่าการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยจะใช้วิธีการสร้างสมมติฐานโดยจะเริ่มต้นด้วยการมีชุดข้มูลตัวอย่างบวกและชุดข้อมูลตัวอย่างลบ $E^+$, $E^-$ และความรู้ภูมิหลังที่อยู่ในรูปแบบของกฎลำดับที่หนึ่ง และพยายามค้นหาสมมติฐาน H ที่สามารถจำแนกตัวอย่าง $E^+$, $E^-$ ได้ โดยการค้นหานั้นทำโดยนำความสัมพันธ์มาต่อเพิ่มความยาวให้กับสมมติฐานนั้น เมื่อใดก็ตามที่สามารถค้นหาสมมติฐานที่จำแนกตัวอย่างบวก $E^+$และตัวอย่างลบ $E^-$ ได้ก็จะได้กฎที่จำแนกตัวอย่างบวกและลบออกจากกัน โดยกฎที่ทำการค้นหาจะต้องสอดคล้องกับสมการ กล่าวคือ สมมติฐานที่จะมาเป็นกฎได้นั้นจะต้องครอบคลุมตัวอย่างบวกทุกตัวอย่าง ตามเงื่อนไขจากสมการที่ 1 และไม่ครอบคลุมตัวอย่างลบเลยตามเงื่อนไขสมการที่ 2

อย่างไรก็ดีวิธีการโปรแกรมตรรกะเชิงอุปนัยจำเป็นต้องสร้างสมมติฐานขึ้นมาเพื่อทดสอบกับตัวอย่างมากมาย ดังนั้นถ้าจะทำการทดสอบทุกทางเลือกที่เป็นไปได้ก็อาจจะไม่เหมาะสมในการนำไปปฏิบัติ วิธีการที่ใช้ในการค้นหาของวิธีการเรียนรู้โปรแกรมตรรกะเชิงอุปนัยจึงมีส่วนสำคัญในการเลือกและค้นหาสมติฐานที่เหมาะสม โดย (S. Muggleton, L. De Raedt, 1994) กำหนดนิยามของการทำอุปนัย (induction) เป็นส่วนกลับของการทำการนิรนัย (deduction) ไว้คือ

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

การหักลบของการอนุมานกฎ(Deductive inference rule) r ซึ่งทาบกลุ่มรวมของอนุประโยคในสมมติฐาน G ลงบนกลุ่มรวมของอนุประโยคในสมมติฐาน S เพื่อให้ได้ $G \models S$ จะเรียก r ว่าเป็นกฎที่ทำให้เฉพาะเจาะจงมากขึ้น

การนำเข้าของการอนุมานกฎ(Inductive inference rule) r ซึ่งทาบกลุ่มรวมของอนุประโยคในสมมติฐาน S ลงบนกลุ่มรวมของอนุประโยคในสมมติฐาน G เพื่อให้ได้ $G \models S$ จะเรียก r ว่าเป็นกฎที่ทำให้มีความเป็นกลางมากขึ้น

ภายใต้การนิยามที่กำหนดจะเห็นได้ว่าในทุกๆ ระบบที่ใช้วิธีการโปรแกรมตรรกะเชิงอุปนัยจะใช้หลักการนี้ในการค้นหาสมมติฐาน เพียงแต่จะแตกต่างกันตรงที่การทำงานจะเป็นแบบการพยายามทำให้สมมติฐานมีความเฉพาะเจาะจงมากขึ้นหรือลดความเฉพาะเจาะจงลงโดยทำให้สมมติฐานครอบคลุมตัวอย่างมากขึ้น

การใช้งานโปรแกรมตรรกะเชิงอุปนัยนั้นมีทั้งแบบที่สนใจการสร้างสมมติฐานที่ครอบคลุมตัวอย่าง แบบไม่เฉพาะเจาะจงโดยมีวิธีการค้นหาสมมติฐานนั้นโดยเริ่มจากสมมติฐานที่

มีความเฉพาะเจาะจงมากที่สุดก่อน แล้วค่อยขยายสมมติฐานนั้นด้วยการลดทอนเงื่อนไขความสัมพันธ์หรือเพิ่มสมมติฐานใหม่เข้าไป แต่อย่างไรก็ดีการลดเงื่อนไขในแต่ละครั้งนั้นจะต้องตรวจสอบดูด้วยว่าสมมติฐานใหม่ไม่ได้ครอบคลุมตัวอย่างลบที่ไม่ต้องการเข้ามาด้วย

ส่วนงานอีกด้านหนึ่งที่มุ่งเน้นในการค้นหาสมมติฐานที่มีความฉพาะเจาะจงสูงเพื่อมาอธิบายตัวอย่างที่ต้องการนั้นจะเริ่มจากสมมติฐานที่ไม่เฉพาะเจาะจงเลย แล้วค่อยๆ เพิ่มเงื่อนไขหรือเพิ่มกฎที่เฉพาะเจาะจงมากขึ้นๆ

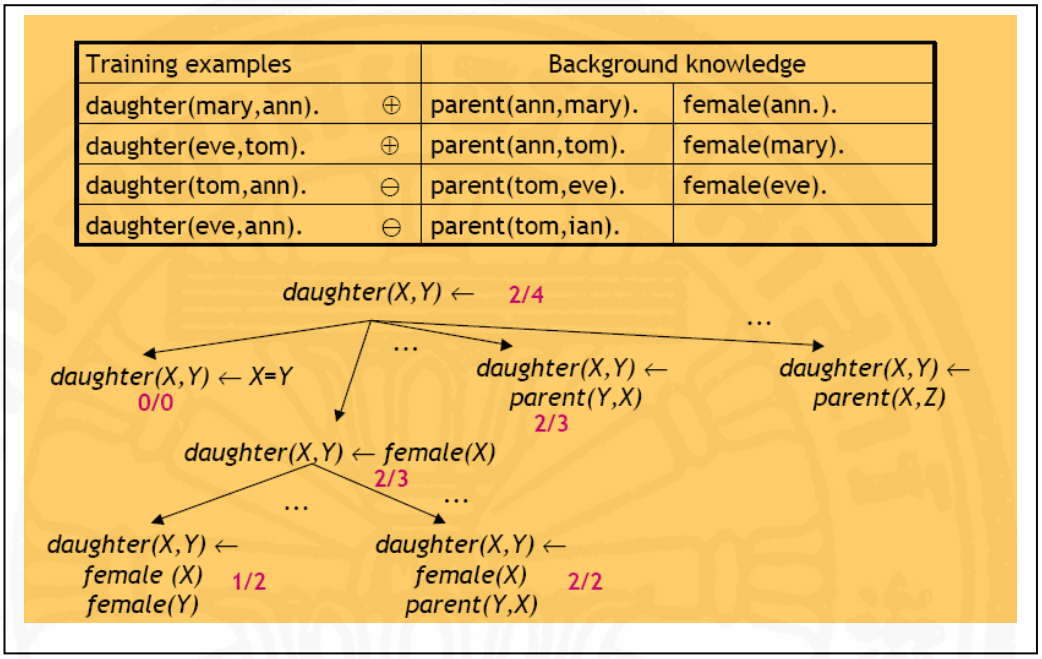
ในระบบที่ทำให้สมมติฐานมีความเป็นกลางมากขึ้นนั้นเป็นการยากที่จะพยายามแก้ไขไม่ให้สมมติฐานใหม่ที่สร้างไม่อธิบายตัวอย่างลบ เนื่องจากระบบทำการเลือกสมมติฐานเดิมมีที่ความเฉพาะเจาะจงอยู่แล้วมาทำการปรับแต่งให้มีความเป็นกลางมากขึ้น สมมติฐานที่ได้ก็จะสามารถอธิบายตัวอย่างบวกได้มากขึ้น ไม่มีผลกับการแก้ไขหรือลดตัวอย่างลบที่ถูกอธิบายได้ในสมมติฐานเดิมได้ ส่วนระบบที่ทำให้สมมติฐานมีความเฉพาะเจาะจงมากขึ้นนั้น ระบบมีความสามารถที่จะแก้ไขตัวอย่างลบที่ถูกอธิบายในสมมติฐานเดิมได้ อันเนื่องมาจากการทำให้เฉพาะเจาะจงมากขึ้นนั้น สมมติฐานที่ได้จะสามารถอธิบายตัวอย่างบวกได้น้อยลงซึ่งรวมทั้งตัวอย่างลบที่สมมติฐานเดิมอาจจะอธิบายไว้ก็จะถูกตัดทอนออกไปได้ด้วย ทำให้ระบบสามารถทำงานกับชุดข้อมูลที่มีข้อมูลผิดพลาด (Noisy data) ได้ดี นอกจากนี้ระบบยังมีความยืดหยุ่นในเรื่องของการปรับปรุงสมมติให้สามารถสอดคล้องกับเงื่อนไขภายหลัง (Posterior conditions) ที่ผู้ใช้อาจจะสามารถกำหนดให้สมมติฐานสามารถอธิบายตัวอย่างลบได้บ้าง แทนที่จะเลือกตัดสมมติฐานข้อนั้นทิ้งไป

จากภาพที่ 2.1 แสดงการค้นหากฎที่ดีที่สุด โดยแสดงความสัมพันธ์ของกฎเป็นลักษณะแผนภูมิต้นไม้ โดยตัวเลขที่กำกับกฎแต่ละข้ออยู่คือ ค่าความครอบคลุมตัวอย่างบวก/ค่าความครอบคลุมตัวอย่างทั้งหมด ซึ่งการทำงานของการค้นหากฎของโปรแกรมตรรกะเชิงอุปนัยนั้น เมื่อสร้างกฎออกมาได้จึงนำกฎนั้นมาเปรียบเทียบกับตัวอย่างบวกและตัวอย่างลบทั้งหมด ซึ่งจะทำจนกระทั่งได้กฎที่ดีที่สุดที่สามารถอธิบายได้แต่ตัวอย่างบวก

จากภาพที่2.1 กฎที่ดีที่สุดที่ได้จากการค้นหาคือ Daughter(X,Y):- female(X), parent(Y,X). เนื่องจากครอบคลุมตัวอย่างบวกครบหมดทั้งสองตัวอย่างและไม่ครอบคลุมตัวอย่างลบเลย

ภาพที่ 2.1

ภาพแสดงตัวอย่างการค้นหากฎของโปรแกรมตรรกะเชิงอุปนัย



ตารางที่2.1

ตารางแสดงค่าความครอบคลุมของกฎที่มีต่อตัวอย่าง (ตัวอย่างจากชุดข้อมูล eastbound)

| กฎที่ | รายละเอียดของกฎ | ค่าความครอบคลุมตัวอย่าง (ครอบคลุม=1,ไม่ครอบคลุม=-1) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | East 1 | East 2 | East 3 | East 4 | East 5 | West 1 | West 2 | West 3 | West 4 | West 5 |
| 1 | eastbound(A) :- has_car(A, B), short(B), open_car(B). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | eastbound(A) :- has_car(A, B). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | eastbound(A) :- has_car(A, B), wheels(B, 3). | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | -1 |
| 4 | eastbound(A) :- has_car(A, B), closed(B). | 1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 | -1 | -1 |
| 5 | eastbound(A) :- has_car(A, B), closed(B), shape(B, rectangle). | 1 | 1 | 1 | -1 | 1 | 1 | -1 | 1 | -1 | -1 |
| 6 | eastbound(A) :- has_car(A, B), wheels(B, 3), has_car(A, C). | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | -1 |
| 7 | eastbound(A) :- has_car(A, B), short(B), wheels(B, 2). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 8 | eastbound(A) :- has_car(A, B), load(B, triangle, 1). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 |
| 9 | eastbound(A) :- has_car(A, B), closed(B), has_car(A, C). | 1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 | -1 | -1 |
| 10 | eastbound(A) :- has_car(A, B), shape(B, rectangle), load(B, triangle, 1). | 1 | -1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 |
| 11 | eastbound(A) :- has_car(A, B), short(B), load(B, triangle, 1). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 |
| 12 | eastbound(A) :- has_car(A, B), open_car(B), has_car(A, C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 13 | eastbound(A) :- has_car(A, B), has_car(A, C), wheels(C, 3). | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | -1 |
| 14 | eastbound(A) :- has_car(A, B), load(B, circle, 1). | 1 | -1 | 1 | -1 | 1 | -1 | 1 | 1 | 1 | -1 |
| 15 | eastbound(A) :- has_car(A, B), wheels(B, 2), load(B, circle, 1). | 1 | -1 | 1 | -1 | 1 | -1 | 1 | 1 | 1 | -1 |
| 16 | eastbound(A) :- has_car(A, B), has_car(A, C), load(C, circle, 1). | 1 | -1 | 1 | -1 | 1 | -1 | 1 | 1 | 1 | 1 |
| 17 | eastbound(A) :- has_car(A, B), has_car(A, C), shape(C, rectangle). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 18 | eastbound(A) :- has_car(A, B), wheels(B, 2), load(B, triangle, 1). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 |
| 19 | eastbound(A) :- has_car(A, B), closed(B), load(B, triangle, 1). | 1 | -1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 20 | eastbound(A) :- has_car(A, B), has_car(A, C), open_car(C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 21 | eastbound(A) :- has_car(A, B), shape(B, rectangle), wheels(B, 3). | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 1 | -1 | -1 |
| 22 | eastbound(A) :- has_car(A, B), shape(B, rectangle), has_car(A, C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 23 | eastbound(A) :- has_car(A, B), has_car(A, C), load(C, triangle, 1). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 24 | eastbound(A) :- has_car(A, B), open_car(B), shape(B, rectangle). | 1 | -1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 | 1 |
| 25 | eastbound(A) :- has_car(A, B), open_car(B), wheels(B, 2). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 26 | eastbound(A) :- has_car(A, B), shape(B, rectangle), wheels(B, 2). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 | 1 |
| 27 | eastbound(A) :- has_car(A, B), has_car(A, C), short(C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 28 | eastbound(A) :- has_car(A, B), has_car(A, C), has_car(A, D). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 29 | eastbound(A) :- has_car(A, B), shape(B, rectangle), load(B, circle, 1). | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | -1 |
| 30 | eastbound(A) :- has_car(A, B), closed(B), wheels(B, 2). | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 |
| 31 | eastbound(A) :- has_car(A, B), long(B), wheels(B, 2). | 1 | -1 | -1 | -1 | -1 | 1 | 1 | -1 | 1 | 1 |
| 32 | eastbound(A) :- has_car(A, B), load(B, triangle, 1), has_car(A, C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 |
| 33 | eastbound(A) :- has_car(A, B), short(B). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 34 | eastbound(A) :- has_car(A, B), has_car(A, C), closed(C). | 1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 | -1 | -1 |
| 35 | eastbound(A) :- has_car(A, B), has_car(A, C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 36 | eastbound(A) :- has_car(A, B), wheels(B, 2). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 37 | eastbound(A) :- has_car(A, B), short(B), has_car(A, C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 38 | eastbound(A) :- has_car(A, B), long(B), has_car(A, C). | 1 | -1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 |
| 39 | eastbound(A) :- has_car(A, B), long(B), shape(B, rectangle). | 1 | -1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 |
| 40 | eastbound(A) :- has_car(A, B), wheels(B, 2), has_car(A, C). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 41 | eastbound(A) :- has_car(A, B), long(B). | 1 | -1 | 1 | -1 | -1 | 1 | 1 | 1 | 1 | 1 |
| 42 | eastbound(A) :- has_car(A, B), shape(B, rectangle). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 43 | eastbound(A) :- has_car(A, B), open_car(B). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 44 | eastbound(A) :- has_car(A, B), short(B), closed(B). | 1 | 1 | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 |
| 45 | eastbound(A) :- has_car(A, B), has_car(A, C), wheels(C, 2). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 46 | eastbound(A) :- has_car(A, B), short(B), shape(B, rectangle). | 1 | 1 | 1 | 1 | 1 | 1 | 1 | -1 | 1 | -1 |
| 47 | eastbound(A) :- has_car(A, B), short(B), load(B, circle, 1). | 1 | -1 | 1 | -1 | 1 | -1 | 1 | 1 | 1 | -1 |

จากตารางที่ 2.1 แสดงตัวอย่างกฎที่ได้จากการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัย ตัวอย่างจากชุดตัวอย่างสำหรับการศึกษาวิธีการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัย โดยมี 10 ตัวอย่าง และ ตัวอย่างEast 1-5 เป็นตัวอย่างบวก, ตัวอย่าง West1-5 เป็นตัวอย่างลบ ซึ่งจะพบว่ากฎที่ดีที่สุดสำหรับข้อมูลชุดนี้คือกฎที่ 44 เนื่องจากครอบคลุมตัวอย่างบวกทั้ง 5 ตัวได้ครบและไม่ครอบคลุมตัวอย่างลบเลย

### 2.1.3 **การจำแนกด้วยกฎจากการรวมเชื่อมโยงกฎ** (Association Rules)

ในปัจจุบันการจำแนกข้อมูลด้วยการเรียนรู้ด้วยเครื่องนั้นทำได้หลายวิธี อีกวิธีหนึ่งที่มีการศึกษาและนำไปใช้งานอย่างแพร่หลายคือการจำแนกข้อมูลจากกฎที่ได้จากการรวมเชื่อมโยงกฎ ( Association Rules) การจำแนกข้อมูลด้วยกฎจากการรวมเชื่อมโยงกฎนั้นจาก งานวิจัยของ (Stefan Mutter) ได้แบ่งขั้นตอนออกเป็น 3 ขั้นตอน ได้แก่ 1) การค้นหากฎโดยใช้การรวมเชื่อมโยงกฎ 2) การคัดเลือกกฎ 3) การถ่วงน้ำหนักให้กับกฎ

#### 2.1.3.1 **การค้นหากฎโดยวิธีการรวมเชื่อมโยงกฎ**

ลักษณะของการเรียนรู้โดยอาศัยการเชื่อมโยงกฎนั้นคือ เป็นการหาความสัมพันธ์ของข้อมูลโดยดูจากรูปแบบของข้อมูลที่เกิดขึ้นกันอย่างซ้ำๆ โดยปกติการค้นหากฎหรือรูปแบบข้อมูลที่เกิดขึ้นซ้ำๆ นั้น ข้อมูลเริ่มต้นจะมาจากฐานข้อมูล (Database) และวิธีการที่จะได้ข้อมูลออกมาจากฐานข้อมูลนั้นจะใช้วิธีการทำเหมือนแร่ข้อมูล (Data mining) รวมกับการใช้เทคนิคการค้นหานับจำนวนรูปแบบซ้ำๆ ที่เกิดขึ้นในข้อมูลตัวอย่างเช่น Apriori รูปแบบของกฎที่ได้จากการเชื่อมโยงข้อมูลโดยค้นหารูปแบบซ้ำๆ ที่เกิดขึ้นกับชุดข้อมูลนั้นจะได้ออกมาอยู่ในรูปแบบของตรรกศาสตร์ประพจน์ (Propositional logic) ซึ่งสามารถใช้อธิบายข้อมูลได้เช่นกัน

รูปแบบที่เกิดขึ้นซ้ำๆ กันของข้อมูลในฐานข้อมูลใดๆ นั้นจะมีเป็นจำนวนมาก ดังนั้นจึงต้องมีขั้นตอนตรวจสอบและตัดสินใจ ที่จะเลือกรูปแบบซ้ำๆ นั้นมาเป็นกฎเพื่อใช้งานได้ โดยวิธีการของการเชื่อมโยงกฎนั้นจะใช้ความน่าจะเป็นที่เกิดขึ้นกับรูปแบบข้อมูลนั้นๆ มาเป็นค่าที่ใช้ตัดสินใจ โดยค่าที่ใช้ในการคำนวณตัดสินใจนั้นประกอบด้วย

<u>ค่าสนับสนุน</u> (Support) คือ ความน่าจะเป็นของการเกิดรายการของสิ่งที่เราสนใจ โดยค่าสนับสนุนนั้นหาได้จาก เป็นเศษส่วนระหว่างจำนวนรายการที่เกิดขึ้นของสิ่งที่เราสนใจ กับจำนวนรายการที่เกิดขึ้นได้ทั้งหมด

ค่าสนับสนุน(A) = จำนวนรายการ A ที่เกิดขึ้น / จำนวนรายการที่เกิดขึ้นได้ทั้งหมด

<u>ความความเชื่อถือ</u> (Confidence) คือ ค่าที่จะใช้ตัดสินใจว่ากฎที่ค้นหามาได้มีโอกาสเกิดขึ้นได้มากน้อยเพียงไร โดยค่าความเชื่อถือจะต้องมีค่ามากกว่าค่า (Threshold) ค่าหนึ่งจึงจะยอมรับได้ว่ากฎนั้นเชื่อถือได้ โดยทั่วไปค่า (Threshold) จะตั้งค่าไว้มากกว่า 50%

ค่าความเชื่อถือ(A->C) = ค่าสนับสนุน (A->C) / ค่าสนับสนุน (A)

ภาพที่ 2.2

แสดงตัวอย่างการค้นหากฎด้วยการรวมเชื่อมโยงกฎ

| TID | Products |
|---|---|
| 1 | A, B, E |
| 2 | B, D |
| 3 | B, C |
| 4 | A, B, D |
| 5 | A, C |
| 6 | B, C |
| 7 | A, C |
| 8 | A, B, C, E |
| 9 | A, B, C |

| TID | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 0 |
| 4 | 1 | 1 | 0 | 1 | 0 |
| 5 | 1 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 1 | 0 | 1 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 | 1 |
| 9 | 1 | 1 | 1 | 0 | 0 |

กฎที่ 1 A, B => E: ค่าความเชื่อถือ =2/4 = 50%

กฎที่ 2 A, E => B: ค่าความเชื่อถือ =2/2 = 100%

กฎที่ 3 B, E => A: ค่าความเชื่อถือ =2/2 = 100%

กฎที่ 4 E => A, B: ค่าความเชื่อถือ =2/2 = 100%

กฎที่ 5 A =>B, E: ค่าความเชื่อถือ =2/6 = 33 %< 50%

กฎที่ 6 B => A, E: ค่าความเชื่อถือ =2/7 = 28% < 50%

จากภาพที่ 2.2 Tid เป็นตารางแสดงรายการที่เกิดขึ้น เมื่อนำมาคำนวณค่าความเชื่อถือปรากฏว่ามีกฎข้อ 1- 4 ที่มีค่าความเชื่อถือมากกว่าหรือเท่ากับ 50% จึงถือว่าเป็นกฎที่ผ่าน ส่วนกฎที่ 5, 6 ไม่ถือว่าเป็นกฎเนื่องจากค่าความเชื่อถือน้อยกว่า 50%

### 2.1.3.2 **การเลือกกฎของการจำแนกด้วยกฎจากการรวมเชื่อมโยงกฎ**

ขั้นตอนนี้เป็นการลดทอนกฎจำนวนมากที่จะเกิดขึ้นจากการค้นหากฎด้วยวิธีการรวมเชื่อมโยงกฎ วิธีการเลือกกฎมีหลายวิธีเช่น

2.1.3.2.1 **การตัดเล็มโดยใช้การจัดลำดับตามการวางนัยโดยทั่วไปจนถึงความเฉพาะเจาะจง** (Pruning using a General to specific ordering) การเลือกกฎด้วยวิธีแบ่งออกได้เป็นสองวิธีย่อยๆ 1) การใช้ค่าความเฉพาะเจาะจงปกติ วิธีนี้จะต้องมีการนิยามคำว่าเฉพาะเจาะจงก่อน เช่น ความเฉพาะเจาะจงอาจจะนิยามจากจำนวนตัวอย่างที่ครอบคลุมได้ หรืออาจจะนิยามความเฉพาะเจาะจงจากจำนวนตัวเชื่อมกฎ หรือความยาวของกฎ ว่ากฎนั้นประกอบด้วยเงื่อนไขกี่เงื่อนไขเป็นต้น 2) อีกวิธีหนึ่งคือใช้ฟังก์ชั่นหรือรูปแบบการคำนวณมาอธิบายค่าความเฉพาะเจาะจง ตัวอย่างเช่นการเลือกกฎจากวิธี Apriori

2.1.3.2.2 **การตัดเล็มจากค่าอัตราการเติบโต** (Pruning using growth rate) การเลือกกฎด้วยอัตราการขยายตัวนี้เป็นการเลือกกฎโดยเทียบจากค่าสนับสนุนที่เปลี่ยนแปลงไปอย่างมีนัยสำคัญของกฎ

2.1.3.2.3 **การตัดเล็มจากค่าทางสถิติ** (Pruning based on statistics) การเลือกกฎด้วยค่าทางสถิติ เช่นค่าความครอบคลุมของกฎที่มีต่อตัวอย่างทั้งหมดในฐานข้อมูล เป็นต้น

### 2.1.3.3 **การถ่วงน้ำหนักให้กับกฎ**

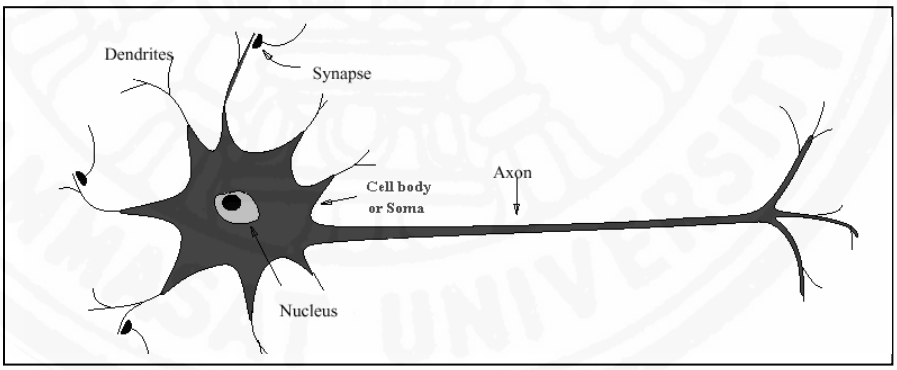
หลังจากการเลือกกฎในข้อ 2 กฎที่ได้ก็ยังไม่สามารถจะนำมาใช้จำแนกข้อมูลได้ทันที เนื่องจากในขณะนี้เรามีกฎที่จะใช้ทดสอบกับตัวอย่าง แต่กฎเหล่านี้ยังไม่สามารถระบุได้ว่าตัวอย่างที่กฎนั้นสามารถอธิบายได้ถูกจำแนกเป็นกลุ่มใด ดังนั้นจึงต้องมีขั้นตอนการทำงานที่จะระบุกลุ่มให้กับกฎเมื่อนำไปทดสอบกับตัวอย่าง งานวิจัย (Stefan Mutter) ได้ใช้วิธีการลงคะแนนเสียงเพื่อสร้างข้อมูลของการระบุกลุ่มให้กับกฎ

### 2.1.4 นิวรอลเน็ตเวิร์ก

นิวรอลเน็ตเวิร์ก เป็นสาขาหนึ่งในด้านการเรียนรู้ของเครื่องที่มีความน่าสนใจ และถูกนำไปประยุกต์ใช้งานในด้านต่างๆ กันอย่างแพร่หลาย โดยหลักการในการทำงานของนิวรอลเน็ตเวิร์กจะอาศัยการเลียนแบบโครงข่ายสมองของมนุษย์ซึ่งมีส่วนประกอบของ 1) ปลายประสาทรับรู้สัญญาณไฟฟ้า (Dendrite) 2) ส่วนรวบรวมข้อมูลที่ได้จากปลายประสาทรับรู้สัญญาณไฟฟ้า (Soma หรือ Cell body) 3) ท่อส่งข้อมูลเพื่อไปยังปลายประสาทรับรู้สัญญาณไฟฟ้าตัวอื่นๆ (Axon) กล่าวคือ ส่วนของปลายประสาทรับรู้สัญญาณไฟฟ้า เปรียบเสมือนอินพุต (Input) ที่เป็นตัวรับข้อมูลเข้ามา ส่วนรวบรวมข้อมูลที่ได้จากปลายประสาทรับรู้สัญญาณไฟฟ้า เปรียบเสมือนส่วนที่รวบรวมข้อมูลต่างๆ ที่ได้รับอินพุตเข้ามาจากหลายๆ แหล่ง และส่วนของท่อส่งข้อมูลเพื่อไปยังปลายประสาทรับรู้สัญญาณไฟฟ้าตัวอื่นๆ จะเปรียบเสมือนเอาต์พุต (Output) ที่จะส่งผลลัพธ์หรือคำตอบที่ได้ออกไปยังนิวตรอนตัวอื่นๆ ในโครงข่ายต่อไป ลักษณะของโครงข่ายสมองมนุษย์ดังภาพที่ 2.3

ภาพที่ 2.3

ลักษณะโครงข่ายสมองมนุษย์




ที่มา: Jain, Mao and Mohiuddin (1996)

ในด้านของคอมพิวเตอร์ได้นำความสามารถของนิวรอลเน็ตเวิร์กมาใช้เรียนรู้ ในปัญหาหลายแบบทั้งปัญหาที่เป็นฟังก์ชันจำนวนจริง (real-valued) ฟังก์ชันไม่ต่อเนื่อง (discrete-valued) รวมทั้งฟังก์ชันที่เป็นค่าเวกเตอร์ (vector-valued) จากตัวอย่างที่ผู้สอนป้อนให้ โดยแนวความคิดในการสร้างนิวรอลเน็ตเวิร์กใช้การจำลองนิวรอน (neuron) และเส้นเชื่อมต่อของนิวรอน เป็นองค์ประกอบพื้นฐาน และนำมาประกอบร่วมกันเป็นเครือข่าย (networks)

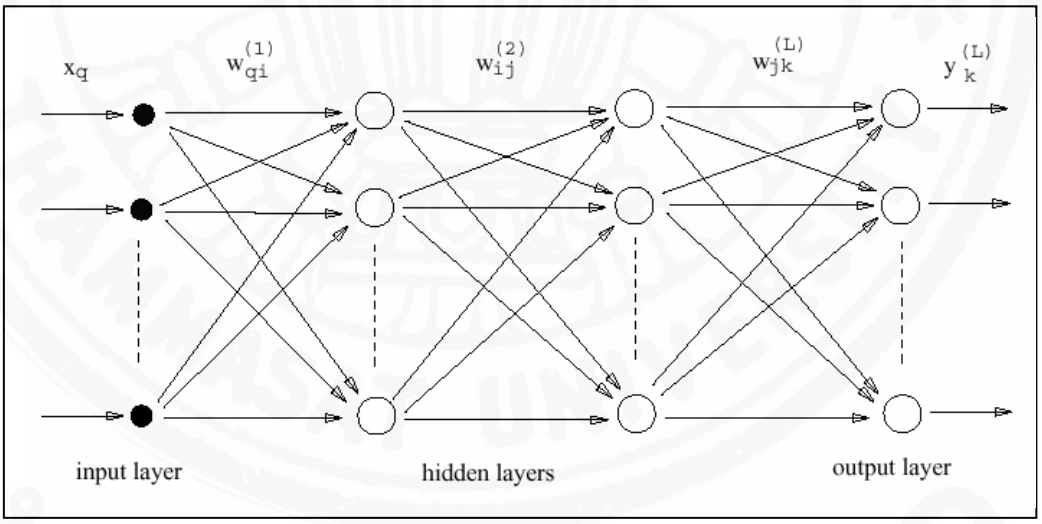
### 2.1.4.1 โครงสร้างของนิวรอลเน็ตเวิร์ก

การใช้งานนิวรอลเน็ตเวิร์กจำเป็นที่จะต้องศึกษาถึงโครงสร้าง และองค์ประกอบของนิวรอลเน็ตเวิร์กแบบที่สนใจนำมาประยุกต์ใช้ โดยนิวรอลเน็ตเวิร์กมีหลายแบบ ซึ่งแต่ละแบบก็จะมีความแตกต่างกันทั้งโครงสร้างและการทำงาน ในงานวิจัยที่นำเสนอนี้จะได้แสดงโครงสร้างและการทำงานของนิวรอลเน็ตเวิร์กแบบมัลติเลเยอร์เพอเซปตรอน มีรายละเอียดดังนี้

นิวรอลเน็ตเวิร์กแบบมัลติเลเยอร์เพอเซปตรอนและอัลกอริทึมแบ็กพรอพาเกชัน มัลติเลเยอร์เพอเซปตรอนนิวรอลเน็ตเวิร์ก (Multilayer Perceptron Neural Network: MLP) หรือ เอ็มแอลพี ดังภาพที่ 4 เป็นแบบหนึ่งของนิวรอลเน็ตเวิร์กมีความสามารถในการสร้างดิชิชันเซอร์เฟสแบบไม่เป็นเชิงเส้น (nonlinear decision surface)

ภาพที่ 2.4

ภาพแสดงลักษณะของมัลติเลเยอร์เพอเซปตรอนนิวรอลเน็ตเวิร์ก




องค์ประกอบย่อยของมัลติเลเยอร์เพอเซปตรอนนิวรอลเน็ตเวิร์ก เป็นองค์ประกอบที่สามารถทำงานกับฟังก์ชันแบบไม่เป็นเชิงเส้น (Nonlinear function) โดยการใช้องค์ประกอบซิกมอยด์ในการทำงาน ดังภาพที่ 2.5

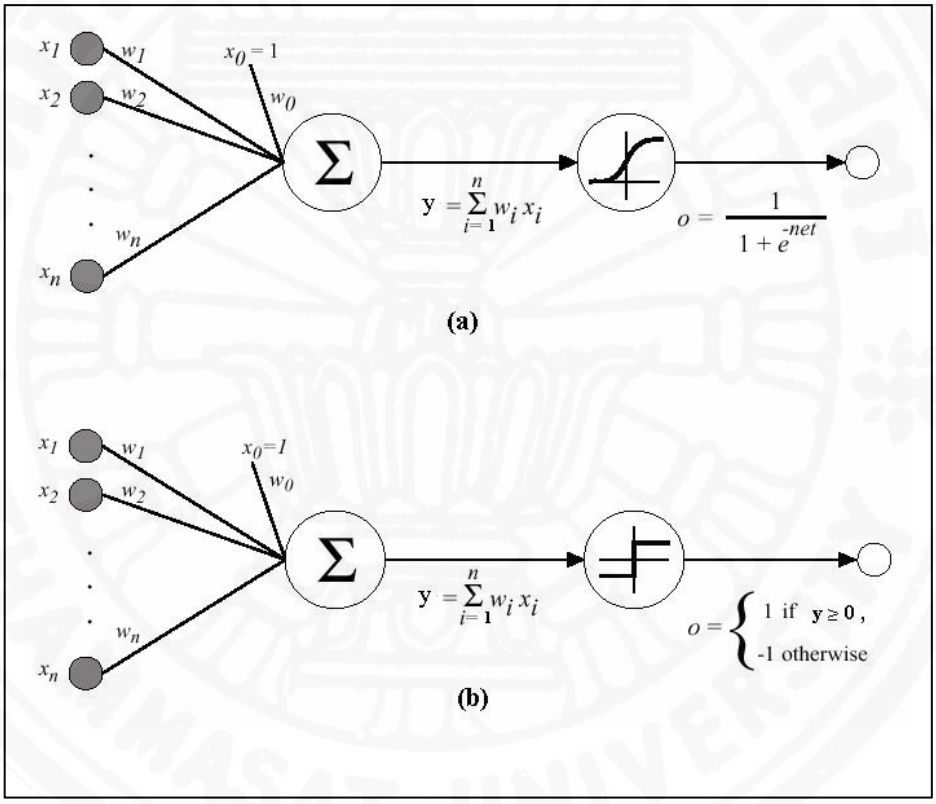
องค์ประกอบซิกมอยด์จะทำงานโดยหาผลคูณเวกเตอร์ของเวกเตอร์อินพุตกับค่าน้ำหนักของเส้นเชื่อมระหว่างอินพุตนิวรอนกับนิวรอนในชั้นถัดไป จากนั้นนำผลที่ได้ผ่านตัวกรอง

(Threshold) เอาต์พุตขององค์ประกอบซิกมอยด์เป็นฟังก์ชันต่อเนื่อง (continuous function) ที่มีค่าเข้าใกล้ 0 หรือ 1 ในขณะที่เอาต์พุตของเพอเซปตรอนธรรมดาเป็นค่าที่ไม่ต่อเนื่อง

ภาพที่ 2.5

ภาพแสดงองค์ประกอบซิกมอยด์และไบโพล่าร์

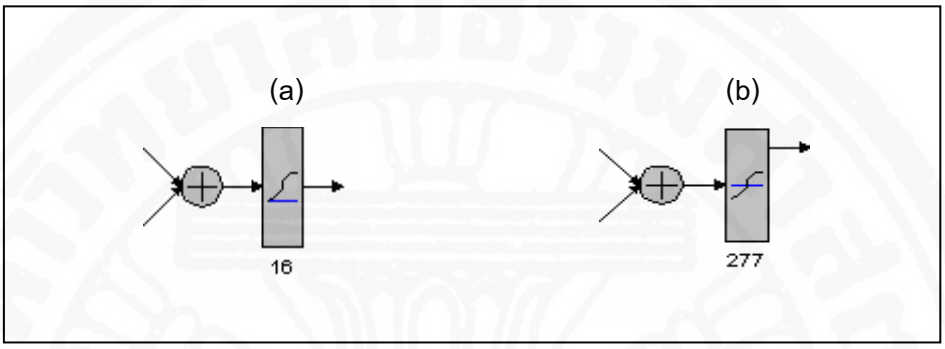
(a) องค์ประกอบซิกมอยด์ และ (b) องค์ประกอบไบโพลาร์



ภาพที่ 2.6

ภาพแสดงองค์ประกอบซิกมอยด์และไบโพล่าร์ในโปรแกรมแมทแล็บ (MATLAB)

(a) องค์ประกอบซิกมอยด์ และ (b) องค์ประกอบไบโพลาร์




โดยฟังก์ชันซิกมอยด์ มีคำนิยามเป็นดังนี้

จาก $O = \sigma (w^u \cdot x^u)$ (2.1)

โดยที่ $\sigma(y) = \frac{1}{1+e^{-y}}$ (2.2)

เมื่อ O เป็น เอาต์พุต

$x^u$ เป็นอินพุต

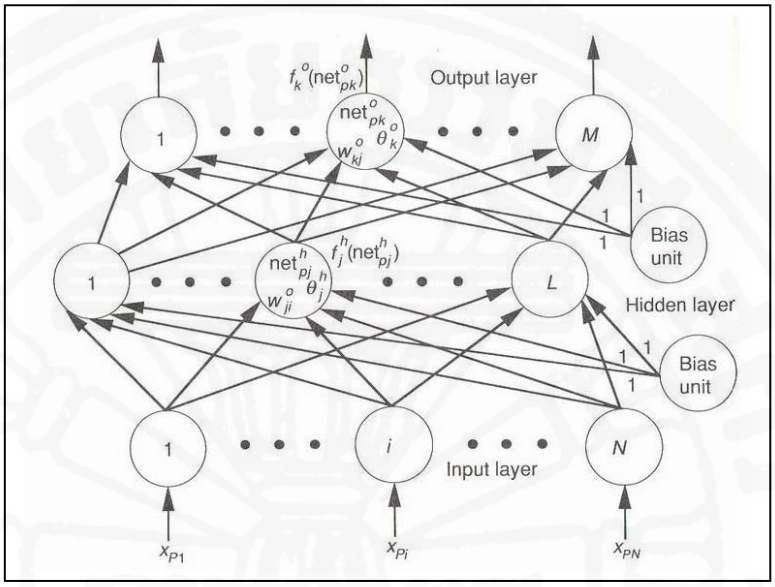
$w^u$ เป็นค่าน้ำหนักของอินพุตนั้นๆ

$\sigma$ เป็นฟังก์ชันซิกมอยด์ (sigmoid function) ซึ่งให้ค่าเอาต์พุตระหว่าง 0 ถึง 1

ขั้นตอนวิธีการเรียนรู้ของมัลติเลเยอร์เพอเซปตรอน ใช้อัลกอริทึมแบ็กพรอพาเกชันสำหรับการปรับค่าน้ำหนักให้กับนิวรอลเน็ตเวิร์กแบบหลายชั้น จะใช้วิธีการ เกรเดียนต์เดสเซนต์ (gradient descent) เพื่อลดกำลังสองของค่าผิดพลาด (squared error) ระหว่างเอาต์พุตที่ได้จากเน็ตเวิร์ก (actual output) และค่าเป้าหมาย (target output) ของอินพุตเหล่านี้ให้น้อยที่สุด โดยขั้นตอนวิธีสำหรับการปรับค่าน้ำหนักของอัลกอริทึมแบ็กพรอพาเกชัน ดังนี้

ภาพที่ 2.7

โครงสร้างนิวรอลเน็ตเวิร์กประกอบการอธิบายอัลกอริทึมแบ็กพรอพาเกชัน




ที่มา: Freeman and Skapura (1991)

กำหนดให้ (ใช้โครงสร้างนิวรอลเน็ตเวิร์ก ตามภาพที่ 2.12) ตัวอย่างที่ใช้ในการเรียนรู้อยู่ในรูปแบบคู่ลำดับ $(\vec{x}, \vec{y})$ เมื่อ $\vec{x}$ เป็นอินพุตเวกเตอร์ของนิวรอลเน็ตเวิร์ก และ $\vec{y}$ เป็นเอาต์พุตเป้าหมายของนิวรอลเน็ตเวิร์ก ฟังก์ชันตัวกระตุ้น (activation function) ใช้ฟังก์ชันซิกมอยด์ $f(y) = \frac{1}{1+e^{-y}}$ $\eta$ เป็นค่าอัตราการเรียนรู้ (Learning rate) อินพุตขององค์ประกอบ i ไปยังองค์ประกอบ j แทนด้วย $x_{ji}$ และค่าน้ำหนักขององค์ประกอบ i ไปยังองค์ประกอบ j แทนด้วย $w_{ji}$

1. สร้างนิวรอลเน็ตเวิร์กตามโครงสร้างที่ออกแบบและกำหนดจำนวนนิวรอนของแต่ละชั้น

2. กำหนดค่าน้ำหนักเริ่มต้นแบบสุ่มให้มีค่าน้อยๆ (เช่น ระหว่าง -0.05 ถึง 0.05) ให้แก่เส้นเชื่อมทุกเส้นในโครงสร้างเน็ตเวิร์ก

3. ปรับค่าน้ำหนักด้วยขั้นตอนดังนี้

   (ก.) นำตัวอย่าง $(\vec{x}, \vec{y})$ มา 1 ตัวจากเซตของตัวอย่างที่ใช้สอนทั้งหมด

   (ข.) นำตัวอย่าง $\vec{x}$ เข้าสู่ส่วนอินพุตของเน็ตเวิร์ก

(ค.) คำนวณผลรวมขององค์ประกอบชั้นซ่อน

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{ji} \quad (2.3)$$

(ง.) คำนวณเอาต์พุตของผลรวมองค์ประกอบที่ได้ของชั้นซ่อน

$$o_{pj}^{h} = f_{j}^{h}(net_{pj}^{h}) = \frac{1}{1+e^{-net_{pj}^{h}}} \quad (2.4)$$

(จ.) คำนวณผลรวมขององค์ประกอบชั้นเอาต์พุต

$$net_{pk}^{o} = \sum_{i=1}^{L} w_{kj}^{o} o_{pj}^{h} \quad (2.5)$$

(ฉ.) คำนวณเอาต์พุตของผลรวมองค์ประกอบชั้นเอาต์พุต

$$o_{pk}^{k} = f_{k}^{o}(net_{pk}^{o}) = \frac{1}{1+e^{-net_{pk}^{o}}} \quad (2.6)$$

(ช.) ทำขั้นตอนตั้งแต่ (ข.) ถึง (จ.) สำหรับทุกๆ ชั้นของเน็ตเวิร์ก

(ซ.) คำนวณค่าความคลาดเคลื่อน $\delta_{pk}^{o}$ สำหรับรูปแบบที่ใช้สอน p กับทุกองค์ประกอบ k ในชั้นเอาต์พุต

$$\delta_{pk}^{o} = o_{k}(1-o_{k})(y_{k}-o_{k}) \quad (2.7)$$

(ฌ.) คำนวณค่าความคลาดเคลื่อน $\delta_{pj}^{h}$ สำหรับรูปแบบที่ใช้สอน p กับทุกองค์ประกอบ j ในชั้นซ่อน

$$\delta_{pj}^{h} = o_{j}(1-o_{j})\sum_{k=1}^{L}\delta_{pk}^{o} w_{kj} \quad (2.8)$$

(ญ.) ปรับค่าน้ำหนักของเส้นเชื่อม ให้กับชั้นเอาต์พุต

$$w_{kj}(t+1) = w_{kj}(t) + \Delta w_{kj} \quad (2.9)$$

$$\text{โดยที่ } \Delta w_{kj} = \eta \delta_{pk}^{o} o_{pj}^{h} \quad (2.10)$$

(ฎ.) ปรับค่าน้ำหนักของเส้นเชื่อม ให้กับชั้นซ่อน

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji} \quad (2.11)$$

$$\text{โดยที่ } \Delta w_{ji} = \eta \delta_{pj}^{h} x_{i} \quad (2.12)$$

(ฏ.) ทำซ้ำขั้นตอน (ข.) ถึงขั้นตอน (ฎ.) สำหรับทุกคู่ลำดับเวกเตอร์อินพุตในตัวอย่างที่ใช้สอนทั้งเซต โดยจะเรียกว่าเป็นการสอนจำนวน 1 รอบ (epoch)

(ท.) ทำซ้ำขั้นตอน (ก.) ถึงขั้นตอน (ฏ.) สำหรับการสอนหลายรอบ เพื่อที่จะทำให้ลดค่าผิดพลาดกำลังสอง ซึ่งการคำนวณหาค่าผิดพลาดกำลังสองใช้สมการดังนี้

$$E(\vec{w}) = \frac{1}{2}\sum_{p\in D}\sum_{k\in output}(y_{kd} - o_{kd})^2 \qquad (2.13)$$

โดยที่ Outputs คือ เซตขององค์ประกอบเอาต์พุตในเน็ตเวิร์ก

D คือ เซตของตัวอย่างที่ใช้สอน

$y_{kd}$ คือ ค่าเอาต์พุตเป้าหมายขององค์ประกอบเอาต์พุตที่ k และตัวอย่างที่ p

$o_{kd}$ คือ ค่าเอาต์พุตขององค์ประกอบเอาต์พุตที่ k และตัวอย่างที่ p

## 2.2 งานวิจัยที่เกี่ยวข้อง

จากปัญหาของการใช้งานกฎจากโปรแกรมตรรกะเชิงอุปนัยทั้ง 2 ข้อคือ 1) ปัญหาเวลาการเรียนรู้ที่ใช้เวลานาน และ 2) กฎที่ได้จากการเรียนรู้โปรแกรมตรรกะเชิงอุปนัยนั้นมีความเฉพาะเจาะจงมากเกินไป จนไม่สามารถใช้อธิบายตัวอย่างใหม่ๆ ที่ไม่ตรงกับตัวอย่างที่ใช้เรียนรู้หรือตัวอย่างที่มีสัญญาณรบกวนได้ งานวิจัยอื่นที่ได้พัฒนารูปแบบการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยที่เกี่ยวข้องกับงานวิจัยนี้ได้แก

งานวิจัยเรื่อง Approximate ILP Rule by Backpropagation Neural Network: A Result on Thai Character Recognition ของ บุญเสริม กิจศิริกุล และ สุกรี สินธุภิญโญ (2542) ได้นำเสนอการประมาณกฎของโปรแกรมตรรกะเชิงอุปนัยด้วยวิธีแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก โดยแก้ปัญหาของการเรียนรู้แบบโปรแกรมตรรกะเชิงอุปนัยที่ไม่สามารถจำแนกตัวอย่างใหม่หรือตัวอย่างที่มีสัญญาณรบกวนได้อย่างถูกต้อง งานวิจัยนี้ได้ใช้ตัวอย่างของการจำแนกตัวอักษรในภาษาไทยที่มีโอกาสมีรูปแบบตัวอักษรไม่ตรงกับรูปแบบที่ใช้ในการเรียนรู้ โดยในขั้นตอนแรกจะทำการแบ่งกฎที่ได้จากการเรียนรู้ ออกเป็นกฎย่อยๆ ก่อนที่จะนำไปถ่วงน้ำหนักด้วยแบ็กพรอพาเกชันนิวรอลเน็ตเวิร์ก ผลที่ได้จากงานวิจัยนี้พบว่าเมื่อกฎถูกแบ่งออกเป็นกฎย่อยๆ และถ่วงน้ำหนักของแต่ละกฎสามารถจะจำแนกตัวอย่างที่เป็นตัวอย่างใหม่ได้อย่างถูกต้อง

งานวิจัยเรื่อง Improving Multiclass ILP by Combining Partial Rules with Winnow Algorithm: Results on Classification of Dopamine Antagonist Molecules ของ สุกรี สินธุภิญโญและคณะ (2547) งานวิจัยนี้มุ่งเน้นที่การจำแนกข้อมูลแบบหลายกลุ่ม โดยการที่จะจำแนกข้อมูลแบบหลายกลุ่มด้วยกฎลำดับที่หนึ่งที่ได้จากโปรแกรมตรรกะเชิงอุปนัยนั้น จะต้องแก้ปัญหาที่

กฎมีความเฉพาะเจาะจงมากเกินไป แนวคิดของงานวิจัยนี้จะเหมือนกับงานวิจัยแรก คือกฎทั้งกฎที่มีความเฉพาะเจาะจงสูงจึงสามารถครอบคลุมตัวอย่างได้น้อย แต่ถ้าใช้แต่เพียงบางส่วนของกฎจะทำให้ความเฉพาะเจาะจงลดลงเป็นผลให้ครอบคลุมตัวอย่างได้มากขึ้น การแบ่งข้อมูลออกเป็นกฎย่อยนี้จะช่วยให้มีข้อมูลที่จะตัดสินใจจำแนกได้มากขึ้น วิธีการจำแนกข้อมูลนั้นจะทำโดยนำข้อมูลค่าความครอบคลุมของตัวอย่างที่มีต่อกฎย่อย มาเป็นข้อมูลตั้งต้นเพื่อถ่วงน้ำหนักให้กับกฎย่อย โดยอาศัยวิธีการถ่วงน้ำหนักแบบวินโน่ (Winow)

เพื่อแก้ปัญหาข้อ2 ที่เกี่ยวกับความเฉพาะเจาะจงของกฎจากโปรแกรมตรรกะเชิงอุปนัย งานวิจัยทั้งสองงานนี้มีแนวคิดที่ว่าการใช้บางส่วนของกฎมาช่วยลดความเฉพาะเจาะจง และเพิ่มข้อมูลที่จะใช้จำแนกตัวอย่าง ซึ่งทำให้การจำแนกมีประสิทธิภาพมากขึ้นโดยสามารถนำกฎของโปรแกรมตรรกะเชิงอุปนัยไปจำแนกข้อมูลใหม่ๆ ที่ไม่อยู่ในชุดข้อมูลเรียนรู้ได้ และสามารถจำแนกตัวอย่างในชุดข้อมูลแบบหลายกลุ่มได้

แต่กฎย่อยที่ใช้ในงานวิจัยทั้งสองงานนี้ต่างก็ได้มาจากกฎที่ได้เรียนรู้โดยสมบูรณ์แล้ว คือจำเป็นจะต้องมีการเรียนรู้ที่สมบูรณ์ก่อนในรอบแรกเพื่อนำผลลัพธ์มาวิเคราะห์แบ่งแยกออกเป็นกฎย่อย และนำไปต่อด้วยการถ่วงน้ำหนักด้วยวิธีต่างๆ ดังนั้นปัญหาเรื่องเวลาที่ใช้ในการเรียนรู้ยังมีอยู่เมื่อใช้วิธีการแบ่งกฎย่อยและถ่วงน้ำหนัก

งานวิจัยเรื่อง Using Bayesian Classifiers to Combine Rules ของ Jesse, Vitor, Irene, David และ Ines (2004) ศึกษาการรวมกฎที่ได้จากการเรียนรู้โปรแกรมตรรกะเชิงอุปนัยจากฐานข้อมูลที่มีความสัมพันธ์หลากหลายของเหมืองข้อมูล (multi-relational data mining) โดยมุ่งประเด็นปัญหาไปที่เมื่อกฎที่ได้มาจากการเรียนรู้ไม่สามารถนำมาจำแนกตัวอย่างได้พอดี โปรแกรมเชิงตรรกะจะให้ผลลัพธ์ออกมาเป็นกฎหลายๆ กฎซึ่งแต่ละกฎจะแสดงความสัมพันธ์หลักๆ ในแต่ละส่วนไว้ งานวิจัยนี้เสนอการรวมและการเลือกกฎหลังจากการเรียนรู้จากโปรแกรมตรรกะเชิงอุปนัย โดยใช้การจำแนกและเลือกแบบโหวต และแบบ เบเซียนเพื่อให้ได้กฎที่สามารถนำมาจำแนกข้อมูลได้

งานวิจัยเรื่อง Frequent Pattern Discovery in First-Order Logic ของ Luc Dehaspe (1998) ศึกษาการค้นหารูปแบบซ้ำๆ ที่เกิดขึ้นในกลุ่มข้อมูลของกฎลำดับที่หนึ่ง งานวิจัยนี้ให้แนวทางการสร้างกฎอีกแบบหนึ่งซึ่งแตกต่างกับการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัย กฎที่ได้มาจากงานวิจัยนี้จะยังไม่สามารถนำไปจำแนกตัวอย่างได้ทันทีเนื่องจากเป็นแค่รูปแบบกฎที่เกิดซ้ำๆ กันในกลุ่มข้อมูล จึงจำเป็นที่จะต้องนำไปประยุกต์เพื่อใช้งานต่อ แต่ข้อดีของกฎเหล่านี้คือมี

ความเฉพาะเจาะจงน้อย และเวลาที่ใช้ในการค้นหากฎเหล่านี้ใช้เวลาน้อยกว่าการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยมาก

งานวิจัยเรื่อง Classification using Association Rules ของ Stefan Mutter (2004) งานวิจัยนี้แสดงวิธีจำแนกข้อมูลด้วยกฎจากการเรียนรู้แบบรวมเชื่อมโยงกฎ (Classification by association rules) ที่ให้ผลลัพธ์กฎอยู่ในรูปแบบของตรรกะศาสตร์ประพจน์ (Propositional Logic) โดยนำเสนอขั้นตอนในการนำกฎที่ได้จากการทำเหมืองข้อมูล (data mining) ที่ผ่านขั้นตอนหากฎเชื่อมโยง (Association rule) นั้นมาทำการจำแนกข้อมูล หลังจากได้กฎจากการรวมกฎแล้วจะต้องมีการคัดเลือกกฎ และการถ่วงน้ำหนักให้กับกฎ จึงจะสามารถนำกฎนั้นมาใช้จำแนกข้อมูลได้

งานวิจัยเรื่อง Partial classification using association Rules ของ Kamal และ Stefanos (1997) และ งานวิจัย Approximate Association Rule ของ Ming, Jyothsna และ Diane (2001) ศึกษาการจำแนกข้อมูลด้วยกฎจากการเรียนรวมเชื่อมโยงกฎ โดยปัญหาของการค้นหากฎด้วยวิธีการรวมเชื่อมโยงกฎนั้นคือ 1) เมื่อมีข้อมูลบางส่วนในฐานข้อมูลสูญหายหรือมีข้อมูลที่ผิดพลาดปะปนอยู่ด้วย ซึ่งจะทำให้เกิดการคำนวณค่าที่ผิดพลาด ส่งผลให้ผลลัพธ์ของการค้นหากฎนั้นออกมาไม่สมบูรณ์ 2) การค้นหากฎที่อธิบายตัวอย่างที่มีความถี่น้อย ที่รวมอยู่ในฐานข้อมูลขนาดใหญ่ ซึ่งความถี่ของการเกิดรูปแบบข้อมูลนั้นไม่สามารถจะถูกตัดสินให้เป็นกฎได้ โดยการพัฒนารูปแบบการค้นหาให้กฎออกโดยแบ่งการค้นหารูปแบบนั้นเป็นส่วนย่อยๆ ผลที่ได้รับคือสามารถค้นหากฎที่มีความถี่ต่ำจากฐานข้อมูลขนาดใหญ่ได้ และสามารถให้ผลลัพธ์ได้ดีขึ้นเมื่อมีข้อมูลผิดพลาดปะปนอยู่ในฐานข้อมูล

งานวิจัยเรื่อง Learning Lazy Rules to Improve the Performance of Classifiers ของ Kai, Zijian และ Geoffery (1999) ได้ศึกษาแนวคิดของการเรียนรู้แบบเกียจคร้านเพื่อใช้ในการเรียนรู้ของเครื่องในแบบต่างๆ โดยระบุลักษณะของการเรียนรู้แบบเกียจคร้านและวิธีการใช้งานที่เกิดขึ้นในปัจจุบัน

งานวิจัย Lazy Decision Tree ของ (Jerome และ Ron และ Yeogirl , 1996) ได้ศึกษาการจำแนกข้อมูลด้วยต้นไม้ตัดสินใจแบบเกียจคร้านที่จะทำการสร้างต้นไม้ตัดสินใจเมื่อเวลาที่ต้องจำแนกข้อมูลเท่านั้น โดยการสร้างต้นไม้ตัดสินใจจะไม่ได้สร้างต้นไม้เพียงต้นเดียวจากข้อมูลทั้งหมด แต่จะสร้างต้นไม้หลายๆ ต้นที่เกี่ยวข้องกับข้อมูลที่จะจำแนกแล้วนำต้นไม้ทั้งหมดมารวมกันภายหลัง ด้วยวิธีการลงคะแนนเสียงจากต้นไม้ตัดสินใจหลายๆ ต้นที่สร้างขึ้น ผลที่ได้คือ

สามารถลดเวลาการสร้างต้นไม้ตัดสินใจลงได้อย่างมาก โดยที่ผลการจำแนกข้อมูลไม่เปลี่ยนแปลงไปจากการสร้างต้นไม้ตัดสินใจจากข้อมูลทั้งหมด

งานวิจัย Lazy Learning of Bayesian Rules ของ Zijian และ Geoffrey (2000) ได้ศึกษาวิธีการเรียนรู้แบบเกียจคร้านบนพื้นฐานการเรียนรู้ด้วยกฎแบบเบเซียน ด้วยวิธีการลดจำนวนข้อมูลที่จำเป็นจะต้องสร้างขึ้นในระหว่างขั้นตอนการเรียนรู้ โดยการแบ่งชุดข้อมูลที่ใช้ในการเรียนรู้ออกเป็นส่วนๆ แล้วเรียนรู้ด้วยเบเซี่ยนทีละส่วน แล้วค่อยนำผลลัพธ์ที่เป็นกฎมาเชื่อมต่อกัน ผลที่ได้จากงานวิจัยนี้คือสามารถลดเวลาการเรียนรู้แบบเบเซี่ยนได้

วิธีการจำแนกข้อมูลในวิทยานิพนธ์นี้ จะปรับปรุงวิธีการเรียนรู้แบบกฎย่อยถ่วงน้ำหนักของงานวิจัย (บุญเสริม กิจศิริกุล และ สุกรี สินธุภิญโญ, 2542) และ (สุกรี สินธุภิญโญและคณะ, 2547) โดยจะใช้การเรียนรู้แบบไม่สมบูรณ์สร้างกฎแทนการเรียนรู้โปรแกรมตรรกะเชิงอุปนัยเดิม การเรียนรู้ที่ไม่สมบูรณ์ที่จะนำมาสร้างกฎนั้นจะใช้แนวคิดจากงานวิจัยของ (Luc Dehaspe, 1998) โดยใช้วิธีการค้นหารูปแบบซ้ำๆ ของข้อมูลที่เกิดขึ้นในกลุ่มของข้อมูลลำดับที่หนึ่ง ซึ่งผลลัพธ์ที่นั้นจะได้เป็นกฎจำนวนมากที่ไม่เฉพาะเจาะจง ซึ่งการปรับปรุงในลักษณะเช่นนี้จะใกล้เคียงกับลักษณะการเรียนรู้แบบเกียจคร้าน ดังนั้นวิธีการเรียนรู้ในงานวิจัยนี้จึงเรียนกว่าการจำแนกข้อมูลจากกฎกึ่งเกียจคร้าน จากนั้นจึงใช้กฎกึ่งเกียจคร้านแทนกฎย่อยโดยนำไปถ่วงน้ำหนักกฎเพื่อใช้จำแนกตัวอย่างต่อไป ซึ่งจากความสามารถของนิวรอลเน็ตเวิร์กที่แสดงให้เห็นประสิทธิภาพในการจำแนกข้อมูลที่มีความซับซ้อนได้อย่างดีจากงานวิจัย (วรวุฒิ ศรีสุขขำ,2547) โดยสามารถจำแนกคลื่นไฟฟ้าเพื่อทำนายลักษณะการเคลื่อนไหวของข้อต่อขามนุษย์ได้ ซึ่งลักษณะของข้อมูลในงานวิจัยนี้มีจำนวนข้อมูลมาก การใช้นิวรอลเน็ตเวิร์กเข้าช่วยประมาณค่าจากกฎที่มีความซับซ้อนนั้น นิวรอลเน็ตเวิร์กน่าจะมีความสามารถทำได้อย่างมีประสิทธิภาพ

## 2.3 วิธีการจำแนกข้อมูลที่เกี่ยวข้อง

### 2.3.1 วิธีการจำแนกข้อมูลด้วยกฎย่อยแบบถ่วงน้ำหนัก

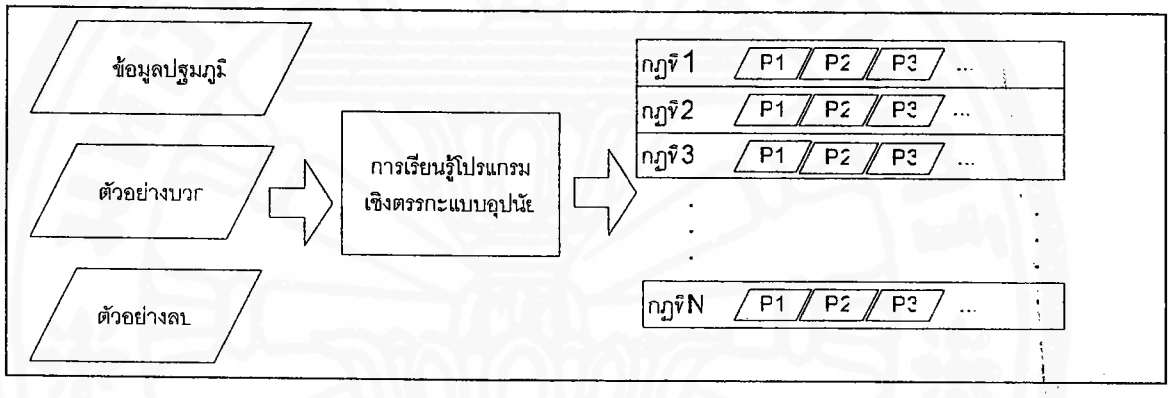
การจำแนกข้อมูลด้วยกฎย่อยแบบถ่วงน้ำหนัก เป็นแนวคิดของงานวิจัย (บุญเสริม และ สุกรี, 2542, หน้า 162-173) และ (สุกรี สินธุภิญโญ และ คณะ, 2004, หน้า 183-195) ที่ใช้บางส่วนของกฎที่ได้จากการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยมาสร้างกฎย่อยๆ เพื่อลดความเฉพาะเจาะจงที่เป็นปัญหาของกฎต้นฉบับที่เป็นผลลัพธ์จากการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยปกติ โดยมีขั้นตอนการเรียนรู้เพื่อจำแนกข้อมูลดังนี้

### 2.3.1.1 การสร้างกฎจากการเรียนรู้โปรแกรมตรรกะเชิงอุปนัย

ขั้นตอนนี้ประกอบด้วย การเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยอย่างสมบูรณ์ เพื่อให้ได้ผลลัพธ์ที่เป็นกฎลำดับที่หนึ่ง ที่สามารถอธิบายได้แต่เฉพาะตัวอย่างบวก ตามรูปแบบการเรียนรู้ปกติของโปรแกรมตรรกะเชิงอุปนัย

ภาพที่ 2.8

ภาพแสดงการเรียนรู้แบบสมบูรณ์ของโปรแกรมตรรกะเชิงอุปนัย




### 2.3.1.2 การสร้างกฎย่อยจากผลลัพธ์ของการเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัย

เมื่อได้ผลลัพธ์จากขั้นตอนที่ 1 นำผลลัพธ์นั้นมาเข้าสู่กระบวนการแยกสร้างกฎย่อย การสร้างกฎย่อยนั้นคล้ายกับการทำ Power set ในวิชาคณิตศาสตร์ เพียงแต่ว่าไม่สามารถจะเลือกกฎที่ไม่มีความสัมพันธ์กันมารวมกันได้ การสร้างกฎย่อยจึงจำเป็นต้องอาศัยความรู้ภูมิหลังในส่วนที่จะอธิบายว่ากฎแต่ละข้อนั้น มีส่วนใดเป็นข้อมูลเข้า และมีส่วนใดเป็นข้อมูลผลลัพธ์

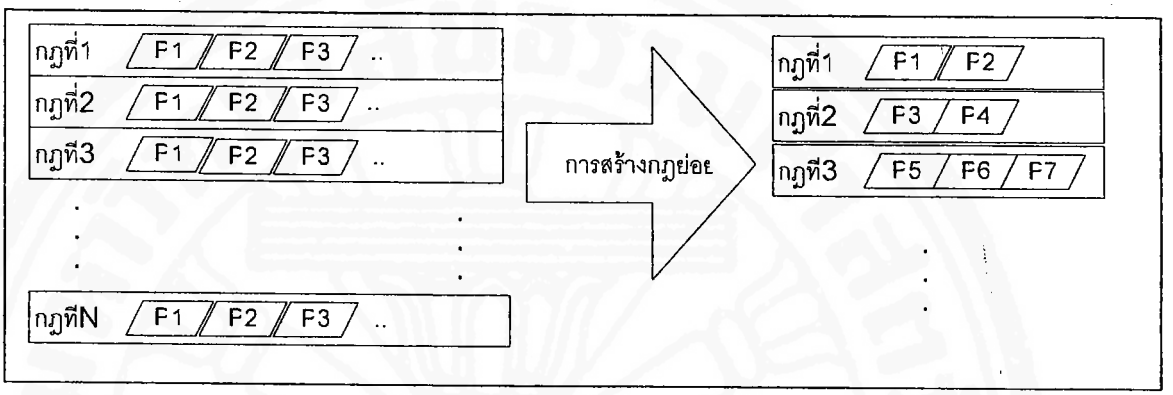
### 2.3.1.3 การถ่วงน้ำหนักให้กับกฎย่อย

เมื่อได้กฎย่อยมาจากขั้นตอนที่ 2 นำตัวอย่างทั้งหมดเข้าเปรียบเทียบกับกฎย่อยทั้งหมดโดยบันทึกค่าความครอบคลุมของกฎกับตัวอย่างนั้นไว้ โดยจัดกลุ่มตามกฎ และนำมาสร้างข้อมูลเพื่อเตรียมให้กับนิวรอลเน็ตเวิร์ก โดยเงื่อนไขการให้ค่าใช้วิธีเดียวกับ การเรียนรู้แบบกึ่งเกียจคร้านคือ ถ้าตัวอย่างนั้นถูกครอบคลุมด้วยกฎจะให้ค่าเป็น 1 และถ้าไม่ถูกครอบคลุมจะให้ค่า -1

จากนั้นสร้างนิวรอลเน็ตเวิร์กที่มีโครงสร้างแบบเดียวกับการเรียนรู้แบบกึ่งเกียจคร้านเพื่อทำการถ่วงน้ำหนักให้กับกฎ

ภาพที่ 2.9

ภาพแสดงขั้นตอนการสร้างกฎย่อย




### 2.3.2 การจำแนกข้อมูลด้วยกฎจากโปรแกรมตรรกะเชิงอุปนัยแบบลงคะแนน

การเรียนรู้ด้วยกฎจากโปรแกรมตรรกะเชิงอุปนัยแบบลงคะแนน เกิดขึ้นเมื่อต้องจำแนกข้อมูลแบบหลายกลุ่ม การใช้กฎจากโปรแกรมตรรกะเชิงอุปนัยอย่างเดียวจะทำให้ได้ผลลัพธ์ที่มีความถูกต้องต่ำ เมื่อนำข้อมูลตัวอย่างมาทดสอบกับกฎของโปรแกรมตรรกะเชิงอุปนัย จะเกิดผลลัพธ์คือ

<u>แบบที่ 1</u> ไม่พบกฎใดๆ เลยที่สามารถอธิบายตัวอย่างนั้นได้ เนื่องจากกฎที่เรียนรู้มามีความเฉพาะเจาะจงมากเกินไป ซึ่งผลที่ตามมาคือ ตัวอย่างนั้นๆ จะไม่สามารถจำแนกได้ว่าอยู่ในกลุ่มข้อมูลชุดใด

<u>แบบที่ 2</u> มีกฎมากกว่า 1 กฎที่สามารถอธิบายตัวอย่างนั้นได้ ถ้าเกิดขึ้นในกรณีนี้ ตามวิธีปกติของโปรแกรมตรรกะเชิงอุปนัยก็จะไม่สามารถสรุผลได้ว่าตัวอย่างนี้จะไปอยู่ในกลุ่มใด

จากปัญหาทั้งสองข้อการจำแนกข้อมูลแบบหลายกลุ่ม โดยใช้แต่เพียงกฎที่ได้จากการเรียนรู้โปรแกรมตรรกะเชิงอุปนัยไม่เพียงพอ เนื่องจากผลลัพธ์ของการจำแนกจะมีความถูกต้องที่ต่ำ ดังนั้นจำเป็นจะต้องเพิ่มการทำงานบางอย่างเข้าไปเพื่อรองรับเหตุการณ์ทั้งสองกรณีที่จะเกิดขึ้น ซึ่งจะในวิทยานิพนธ์นี้ได้เลือกวิธีการลงคะแนนเพิ่มขึ้นเพื่อรองรับปัญหาที่จะเกิดขึ้น

ขั้นตอนการจำแนกของโปรแกรมตรรกะเชิงอุปนัยแบบลงคะแนนประกอบด้วย 3 ขั้นตอนคือ

**2.3.2.1 การเรียนรู้โดยโปรแกรมตรรกะเชิงอุปนัย**

ขั้นตอนนี้ประกอบด้วย การเรียนรู้ของโปรแกรมตรรกะเชิงอุปนัยอย่างสมบูรณ์ เพื่อให้ได้ผลลัพธ์ที่เป็นกฎลำดับที่หนึ่ง ที่สามารถอธิบายได้แต่เฉพาะตัวอย่างบวก ตามรูปแบบการเรียนรู้ปกติของโปรแกรมตรรกะเชิงอุปนัย โดยใช้ข้อมูลภูมิหลัง ประกอบกับตัวอย่างบวก และตัวอย่างลบทั้งหมด

**2.3.2.2 การสร้างข้อมูลฐานคะแนนเสียง**

1. เมื่อได้ผลลัพธ์จากขั้นตอนที่ 1 แล้วนำกฎเหล่านั้นมาทดสอบกับตัวอย่างทั้งหมด เพื่อบันทึกค่าความครอบคลุมของกฎทุกกฎกับตัวอย่างทุกตัวอย่างไว้
2. นับจำนวนของตัวอย่างที่กฎนั้นครอบคลุม โดยแบ่งตามกลุ่มของตัวอย่างนั้น
3. การกำหนดกลุ่มคำตอบให้กับกฎ โดยกำหนดกลุ่มที่กฎนั้นจะตอบจากกลุ่มคำตอบที่มีจำนวนสูงสุดที่กฎนั้นครอบคลุม ตัวอย่างเช่น กฎข้อที่ 1 ครอบคลุม ตัวอย่างของกลุ่มที่ 1 20 ตัวอย่าง, กลุ่มที่ 2 10 ตัวอย่าง, กลุ่มที่ 3 5 ตัวอย่าง ดังนั้นกฎที่ 1 จะมีกลุ่มประจำกฎเป็น 1

**2.3.2.3 การใช้ข้อมูลฐานคะแนนเสียงในการจำแนกข้อมูล**

เมื่อจะทำการจำแนกข้อมูลเริ่มต้นที่นำตัวอย่างที่ต้องการจำแนกมาทดสอบกับกฎทุกกฎเพื่อหาว่ามีกฎใดที่ครอบคลุมตัวอย่างที่กำลังจำแนกได้บ้าง ซึ่งจะเกิดกรณีได้ดังนี้

1. เมื่อมีกฎเพียงกฎเดียวที่ครอบคลุมตัวอย่างที่ทำการจำแนก กลุ่มที่จะเป็นคำตอบคือกลุ่มประจำกฎของกฎนั้น ที่ครอบคลุมตัวอย่างที่จำแนก
2. เมื่อไม่มีกฎใดเลยที่สามารถครอบคลุมตัวอย่างที่ทำการจำแนก กลุ่มที่จะเป็นคำตอบคือกลุ่มที่มีจำนวนตัวอย่างในขั้นตอนการเรียนรู้ที่มากที่สุด
3. เมื่อมีกฎมากกว่า 1 กฎครอบคลุมตัวอย่างที่ที่ทำการจำแนก คำตอบคือกลุ่มประจำกฎของกฎที่ครอบคลุมตัวอย่างที่มากที่สุด เช่น ตัวอย่างที่ทำการจำแนก ถูกครอบคลุมด้วยกฎที่ 1

และ 2 โดยในขั้นตอนการเรียนรู้กฎที่ 1 มีครอบคลุมตัวอย่างทั้งหมด 10 ตัวอย่าง กฎที่ 2 ครอบคลุมตัวอย่าง 15 ตัวอย่าง ดังนั้นกลุ่มที่เป็นคำตอบคือกลุ่มประจำกฎของกฎที่ 2