บทที่ 3
อัลกอริทึมสำหรับการเพิ่มขยายการค้นหากฎความสัมพันธ์โดยอาศัยหลักของความน่าจะเป็น

ปัจจุบันพบว่าข้อมูลต่างๆ มีการปรับเปลี่ยนตลอดเวลาทั้งในด้านการเพิ่ม, การลบหรือแก้ไขข้อมูล การเปลี่ยนแปลงข้อมูลให้มีความทันสมัยเป็นปัจจุบันจะมีผลต่อการเปลี่ยนแปลงกฎความสัมพันธ์ที่ได้ค้นหาไว้แล้ว ดังนั้นจึงต้องทำการค้นหากฎความสัมพันธ์ใหม่เมื่อข้อมูลเกิดการเปลี่ยนแปลงกับฐานข้อมูล

การเพิ่มขยายการค้นหากฎความสัมพันธ์เป็นอัลกอริทึมที่ใช้ในปรับปรุงกฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ เมื่อมีการเพิ่ม transaction ใหม่เข้าไปใน original databaseในที่นี้จะเรียกส่วนของรายการข้อมูลใหม่ที่เพิ่มเข้าไปนี้ว่า incremental database (db) ซึ่ง transaction ที่เพิ่มเข้ามานี้จะมีผลต่อกฎความสัมพันธ์ที่ได้จากการทำคาต้าไมน์นิ่งใน original database (DB) โดยอาจทำให้กฎที่มีอยู่ไม่มีความถูกต้องเมื่อทำการค้นหากฎความสัมพันธ์ที่ปรากฏในฐานข้อมูลที่ได้รับการปรับปรุงใหม่ทั้งหมด (updated database : UP)

การปรับปรุงฐานข้อมูลใหม่ อาจหมายถึง การเพิ่มฐานข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม, การลดรายการในฐานข้อมูลเดิม และกรณีที่มีทั้งการลดและเพิ่มรายการในฐานข้อมูลเดิม สำหรับในงานวิจัยนี้จะเป็นการค้นหากฎความสัมพันธ์ในกรณีของเพิ่มรายการใหม่เข้าไปในฐานข้อมูลเดิมเท่านั้น

เมื่อฐานข้อมูลที่ได้จากการปรับปรุงได้ผ่านกระบวนการค้นหากฎความสัมพันธ์แล้ว อาจเกิดการเปลี่ยนแปลง frequent itemset เดิมทำให้เกิดการสแกนฐานข้อมูลเดิมใหม่ โดยทั่วไปขนาดของฐานข้อมูลเดิมมักจะมีขนาดใหญ่กว่าฐานข้อมูลใหม่ที่เพิ่มเข้ามา ดังนั้นการสแกนฐานข้อมูลเดิมจะมีผลต่อความถูกต้องของกฎความสัมพันธ์, เวลาที่ใช้ในการประมวลผล รวมถึงทำให้เกิดการสิ้นเปลืองทรัพยากรในการค้นหา frequent itemset ที่พบในฐานข้อมูลใหม่แต่เป็น small itemset ใน original database

ซึ่งกรณีที่ต้องทำการสแกน original database สำหรับค้นหา frequent itemset ที่พบจาก incremental database นี้ จะพบว่าสมาชิกของ frequent itemset ใหม่ไม่ทุกตัวที่จะกลายมาเป็น frequent itemset ใน updated database ดังนั้นการที่ต้องทำการสแกน original databaseใหม่นั้นจะเป็นการเสียเวลาที่ใช้ในการค้นหาเนื่องจาก itemset ที่เกิดใน incremental database จะมีจำนวนมากกว่าที่พบใน original database ด้วยการพิจารณาค่าของ frequent itemset จาก $min\_sup_{db} * |db|$

อัลกอริทึม Apriori เป็นอัลกอริทึมที่ได้รับความนิยมในการค้นหากฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ การไมน์นิ่งเป็นในลักษณะที่เรียกว่า level-wise search ดังนั้นในแต่ละ k-itemset จะเก็บ frequent itemset ที่เกิดจากการไมน์นิ่ง เพื่อใช้ในการสร้างกฎความสัมพันธ์ที่ได้จากการดึงความรู้ที่ซ่อนอยู่ในฐานข้อมูลไว้

การที่ข้อมูลต่างๆ ในฐานข้อมูล อาจเกิดการเปลี่ยนแปลงขึ้น ในลักษณะของฐานข้อมูลที่เรียกว่า dynamic database การไมน์นิ่ง dynamic database จากหลักการทำงานของ Apriori ถึงแม้ผลการค้นหากฎความสัมพันธ์ที่ได้จะมีความถูกต้องด้วยกระบวนการไมน์นิ่งฐานข้อมูลที่ปรับปรุงใหม่ทั้งหมด โดยไม่ได้นำความรู้เดิมที่ได้จากการไมน์นิ่ง original databaseมาใช้ ทำให้เสียเวลาในการค้นหา frequent k-itemset ใหม่ทั้งหมด

เพื่อลดจำนวนการค้นหา frequent k-itemset ใหม่ทั้งหมด ด้วยหลักการของ Apriori อัลกอริทึม FUP ได้นำเสนอวิธีการแก้ปัญหาด้วยการนำความรู้ที่ได้จากการไมน์นิ่ง original database มาใช้ให้เกิดประโยชน์ คือนำ frequent k-itemset เดิมที่ได้จากการไมน์นิ่งมาทำการปรับปรุงค่า support count ที่ปรากฏในฐานข้อมูลใหม่ ซึ่งเป็นการลดการค้นหา frequent itemset ใน original database ซ้ำ และเมื่อเกิด frequent k-itemset ใน incremental database จะมีการนำ frequent k-itemset ใหม่นี้มาทำการสแกนเพื่อหาค่า support count ใน original database เพื่อให้ได้frequent k-itemset ที่ถูกต้องสำหรับสร้างกฎความสัมพันธ์

ความรู้ที่ได้จากการไมน์นิ่งในฐานข้อมูลทั้งหมดจะเป็น frequent itemset เท่านั้น ดังนั้น Thomas et al [6] และFeldman et al [12] ได้นำแนวคิดของ negative border มาพัฒนาอัลกอริทึม โดยจะทำการเก็บข้อมูลของ frequent k-itemset และส่วนที่เรียกว่า negative border k-itemset ไว้ทั้งหมดเพื่อลดการสแกน original database อัลกอริทึมนี้จะทำงานได้ดีในกรณีที่ไม่พบว่ามี frequent k-itemset ใหม่เกิดขึ้น แต่ในทางกลับกัน ถ้าพบว่ามี frequent k-itemset ใหม่เกิดขึ้นจะต้องมีการสแกนฐานข้อมูลปรับปรุงทั้งหมด ดังที่งานวิจัย [13] ได้ทำการเปรียบเทียบประสิทธิภาพการทำงานระหว่างอัลกอริทึม negative border [6] เทียบกับอัลกอริทึมApriori และพบว่าการทำงานของอัลกอริทึม negative border เหมาะกับ item ที่ไม่มากและ execution time ของ negative border ในกรณีที่พบ frequent k-itemset ใหม่และต้องสแกนฐานข้อมูลทั้งหมดนั้นใช้เวลาไม่แตกต่างหรืออาจจะนานกว่าการค้นหากฎความสัมพันธ์ด้วย Apriori นอกจากนี้ทั้งงานวิจัยของ Thomas et al และFeldman et al ไม่มีการนำเสนอในส่วนของการ recover frequent k-itemset ใหม่ซึ่งอาจเกิดขึ้นได้ใน incremental database เมื่อค่า minimum support threshold มีค่าน้อย

เพื่อลดปัญหาการจัดเก็บข้อมูลและการสแกนฐานข้อมูลใหม่ทั้งหมดในกรณีที่พบ frequent itemset ใหม่ Hong et al [14] และ Amornchewin และ Kreesuradej [15] ได้นำเสนอแนวคิด expected frequent itemset เพื่อลดปัญหาการจัดเก็บข้อมูลทุกตัวที่เป็นสมาชิกของ candidate k-itemset ที่ไม่เป็น frequent k-itemset แนวคิดนี้จะมีการนำเสนอการหา expected frequent itemset ที่

มีโอกาสจะกลายมาเป็น frequent itemset เมื่อมี incremental database เพิ่มเข้ามา ทั้งนี้เพื่อลดหรือหลีกเลี่ยงการสแกน original database โดยจะมีการคำนวณหาขนาดของ incremental database ที่ใช้ในการประมาณค่าของ expected frequent itemset เพื่อให้สามารถทำงานได้อย่างมีประสิทธิภาพ แนวคิดนี้จะให้ความสำคัญกับ expected frequent k-itemset เช่นเดียวกับ frequent k-itemset ยกเว้นในส่วนของการสร้างกฎความสัมพันธ์ที่จะใช้ frecuent k-itemset ($k \geq 2$) มาสร้างกฎความสัมพันธ์ ดังนั้น expected frequent k-itemsetจะถูกนำมาใช้ในการสร้าง candidate k-itemset ในขั้นตอนของการ join ด้วยหลักการเดียวกับ apriori ซึ่งทำให้จำนวน itemset ที่ต้องจัดเก็บมีจำนวนมาก

สำหรับงานวิจัยนี้จะเป็นการนำเสนอวิธีการแก้ปัญหาการทำดาต้าไมน์นิ่งในการหากฎความสัมพันธ์ที่ได้จากฐานข้อมูลที่ปรับปรุงใหม่จากการเพิ่มรายการข้อมูลเข้าไปในฐานข้อมูลเดิม เพื่อลดจำนวนครั้งและจำนวน itemset ที่จะต้องสแกนในฐานข้อมูลเดิม ด้วยแนวคิดดังนี้

1. การใช้ค่าความน่าจะเป็นที่ได้จากการคำนวณ โดยอาศัยหลักการของ Bernoulli trial มาช่วยในการค้นหา itemset ที่มีโอกาสจะกลายเป็น frequent itemset ได้เมื่อมีรายการจากฐานข้อมูลใหม่เพิ่มเข้ามา เรียก itemset นี้ว่า expected frequent itemset โดยค่าของ expected frequent itemset ได้จะต้องผ่านค่ากำหนดของค่าความน่าจะเป็นอย่างน้อยที่ผู้ใช้เป็นผู้กำหนด (user-defined) คือ $prob_{pl}$ ด้วยค่าความน่าจะเป็นของ itemset ที่มีโอกาสเป็น frequent itemset แน่นอนจะมีค่าค่อนข้างสูงหรือใกล้เคียง 1 ในขณะที่บาง itemset ที่มีค่าความน่าจะเป็นที่ผ่าน $prob_{pl}$ อาจจะกลายเป็น frequent itemset ได้เมื่อมีค่า support count จำนวนหนึ่งปรากฏใน incremental database ทำให้สามารถลดจำนวน expected frequent itemset ที่จะเก็บไว้สำหรับไมน์นิ่งในครั้งต่อไป

2. ในกรณีที่การไมน์นิ่งค้นพบ frequent itemset ใหม่ ซึ่งในกรณีนี้จะหมายถึง itemset ที่เป็น small itemset ใน original database ถ้าค่า support count ของ frequent itemset นี้มีค่าจำนวนหนึ่งที่มากกว่าค่า minimum support threshold ของ incremental database การจะทราบว่า frequent itemset ใหม่นี้จะเป็น frequent itemset ใหม่ใน updated database ได้จะสามารถทำได้ด้วยการสแกน original database ดังที่พบในงานวิจัยของ FUP

งานวิจัยนี้ได้นำเสนอแนวคิดที่จะลดจำนวน frequent itemset ใหม่ที่จะนำไปสแกนใน original database โดยนำค่า minimum expected frequent ที่ได้จากการไมน์นิ่ง original database มาใช้ในการประมาณค่า support count สูงสุดให้กับfrequent itemset ใหม่ หรือเป็น small itemset ใน original database

จากที่พบว่า frequent itemset ใหม่เป็น small itemset ใน original database ซึ่งค่า support count สูงสุดของ small itemset นี้ที่เป็นไปได้จะต้องน้อยกว่าค่า minimum expected frequent แน่นอน ดังนั้นค่าประมาณสูงสุดที่เป็นไปได้ของ small itemset นี้คือ minimum expected frequent ลบ 1 เมื่อทราบค่าประมาณสูงสุดแล้วเราจะนำมาพิจารณาความเป็นไปได้หรือโอกาสที่ frequent itemset ใหม่นี้จะกลายมาเป็น frequent itemset ใน updated database ได้โดยนำค่า คือ minimum expected

frequent -1 มาบวกกับค่า support count ของ frequent itemset ใหม่ ถ้าผลบวกที่ได้มากกว่าหรือเท่ากับ min_sup_{UP} ซึ่งจัดว่าเป็น itemset ที่มีโอกาสจะเป็น frequent itemset หรือ expected frequent itemset ใน updated database ดังนั้นจะเก็บไปสแกนใน original database ในขั้นตอนสุดท้ายสำหรับทุก k-itemset

3. การค้นหา frequent i temset สำหรับ itemset ใหม่ที่เกิดขึ้น ในที่นี้หมายถึง itemset นั้นๆ ปรากฏแต่เพียงใน incremental database เนื่องจากถ้าการไมน์นิ่งด้วยค่า minimum support threshold ที่น้อย อาจทำให้ itemset ใหม่ที่เพิ่งปรากฏใน incremental database กลายเป็น frequent itemset ได้ในที่นี้เปรียบเสมือนสินค้าใหม่ที่เพิ่งผลิตออกมาและได้รับความนิยมมาก ดังนั้นนอกจากการปรับปรุงค่า support count ของ frequent itemset และ expected frequent itemset แล้วงานวิจัยนี้จะครอบคลุมการค้นหา frequent itemset ใหม่ที่เกิดขึ้นจาก incremental database ที่เพิ่มเข้ามาได้อย่างถูกต้องและมีประสิทธิภาพ โดยอัลกอริทึมที่นำเสนอนี้จะสามารถตรวจสอบได้ว่า item ใดเป็น item ใหม่ที่เกิดใน incremental database จากการ update 1-itemset ทำให้ทราบว่าถ้าเป็น item ใหม่ที่เกิดใน incremental databaseและไม่มีความจำเป็นต้องไปสแกนใน original database จึงแตกต่างจากอัลกอริทึมของ FUP นอกจากนี้การค้นหา frequent itemset ใหม่นี้จะแตกต่างจาก negative border ที่ไม่มีในส่วนของการค้นหา frequen t itemset ใหม่ที่นอกเหนือจาก frequent k-itemset และ Boder k-itemset ที่มี

ดังจะมีการนำเสนอขั้นตอนการทำงานในส่วนต่อไปดังนี้

3.1 Probability-based Incremental Association Rule Discovery Algorithm

งานวิจัยนี้เป็นการนำเสนอแนวคิดของการใช้ค่าความน่าจะเป็นมาช่วยในการประมาณค่า itemset ที่คาดว่าจะกลายเป็น frequent itemset โดยนำทฤษฎีของเบอร์นูลลีมาประยุกต์ใช้ในการพิจารณาการเกิดของ frequent itemset ซึ่งประกอบด้วย 2 เหตุการณ์คือสำเร็จและไม่สำเร็จ ถ้าพบว่า itemset ใดที่เป็น frequent itemset และเหตุการณ์ที่ itemset ใดเป็น small itemset หรือไม่ผ่านค่า minimum support threshold เป็นความไม่สำเร็จ

ด้วยแนวคิดของทฤษฎีเบอร์นูลลีในงานวิจัยนี้ได้นำมาประยุกต์ใช้ในการประมาณค่าของ itemset ที่คาดว่าจะกลายเป็น frequent itemset โดยนำค่าสถิติการเกิดของ itemset ต่างๆ ที่ปรากฏใน original database ซึ่งในที่นี้ให้มีขนาดเท่ากับ m transactions ด้วยค่าความน่าจะเป็นที่ itemset นั้นๆ ปรากฏใน original database ซึ่งหมายถึงค่าความสำเร็จที่ itemset นี้ปรากฏใน original database แทนด้วยค่า p คำนวณได้ดังนี้

$$p = \frac{sup\ port\ count}{|DB|}$$

โดย p หมายถึง ค่าความน่าจะเป็นที่เกิดความสำเร็จจากการทดลอง

|DB| หมายถึงขนาด original database ในที่นี้คือ m transactions

จากทฤษฎีของเบอร์นูลลีนี้สามารถนำมาใช้ในการประมาณค่าของการเกิดของ itemset ต่างๆ ที่มีโอกาสจะกลายเป็น frequent itemet ใน updated database เมื่อมี incremental ขนาด n transations เพิ่มเข้าไปใน original database ขนาด m transations ด้วยจำนวน support count x จำนวนที่จะทำให้ item นี้เป็น frequent itemset ใน updated database ในที่นี้ ค่า suppor. count x นี้ คำนวณได้ดังนี้

x = (minimum support * (|DB|+|db|) - support count DB

จากค่าต่างๆ ข้างต้นสามารถนำมาคำนวณค่าความน่าจะเป็นที่ item A จะเป็น frequent itemset ใน updated database ขนาด n+m transaction ได้ดังสมการ(1)

$$P(A) = \binom{n+m}{x} \cdot p^x \cdot (1-p)^{n+m-x} \qquad (1)$$

โดยทั่วไป การพิจารณาว่า itemset ใดจะเป็น frequent itemset ได้นั้นจะดูได้จากค่า support count ของ itemset นั้นที่ปรากฏในฐานข้อมูลมีค่ามากกว่าหรือเท่ากับค่า minimum support threshold ที่ผู้ใช้กำหนด แต่ในบาง itemset ที่มีค่า support count น้อยกว่าค่า minimum support threshold นั้นอาจแต่มีค่าความน่าจะเป็นของการเกิดที่มากพอที่จะผ่านค่าความน่าจะเป็นที่ผู้ใช้กำหนดเมื่อมีการเพิ่ม incremental database ที่ขนาดหนึ่ง ในที่นี้ค่าความน่าจะเป็นที่ผู้ใช้กำหนดเรียกว่า $prob_{pl}$ ซึ่งเป็นค่าประมาณการที่ผู้ใช้ยอมรับได้ว่า itemset นี้มีโอกาสที่เกิดขึ้นอย่างน้อย k ค่า เมื่อมี incremental database ขนาด n transaction เพิ่มเข้ามาใน original database ขนาด m transaction ได้ด้วยค่าความน่าจะเป็นที่คำนวณได้ โดยอาศัยความน่าจะเป็น p ซึ่งเป็นค่าที่เกิดจริงใน original database โดยในแต่ละ itemset สามารถคำนวณหาค่าความน่าจะเป็นที่จะเกิดใน updated database อย่างน้อย k ค่าได้ดังสมการที่ (2)

$$P(x<k)_{item} = \sum_{x=0}^{k-1} \binom{n+m}{x} p^x (1-p)^{n+m-x} \qquad (2)$$

(n+m) หมายถึง ขนาดของข้อมูลของ updated database ด้วยขนาดของ original data (|DB|) ขนาด m transactions และ incremental database (|db|) ขนาด n transactions

p หมายถึง ค่าความน่าจะเป็นที่เกิดความสำเร็จจากการทดลอง ซึ่งคำนวณได้ดังนี้

$$p = \frac{sup\ port\ count}{|DB|}$$

(1-p) หมายถึง ค่าความน่าจะเป็นที่การทดลองไม่มีความสำเร็จ

k หมายถึง จำนวนครั้งที่การทดลองจะเกิดความสำเร็จ คำนวณได้ดังนี้

$$k = min\_sup_{UP} - sup\ port\ count_{DB}$$

สำหรับค่าความน่าจะเป็นที่คำนวณได้จากสมการ (2) จะเป็นค่าความน่าจะเป็นอย่างน้อยที่จะเกิดด้วยค่า support count อย่างน้อย k ค่า เมื่อมีการเพิ่ม incremental database ขนาด n transaction เข้าไป ดังนั้นค่าความน่าจะเป็นของ itemset นี้จะกลายเป็น frequent itemset ใน updated database ด้วยค่าความน่าจะเป็นเท่าใด สามารถคำนวณได้ดังสมการ (3)

$$P\left(x \geq k\right)_{item} = 1 - P\left(x < k\right)_{item} \qquad (3)$$

โดย $P\left(x \geq k\right)_{item}$ หมายถึง ความน่าจะเป็นที่ item จะมีโอกาสที่ค่า support count x มีค่ามากกว่า k

k หมายถึง ค่าความแตกต่างระหว่าง min_sup_{UP} และค่า support count ของ item (support $count_{item}$) ที่จะทำให้ item นั้นๆ กลายมาเป็น frequent itemset ใน updated database โดยค่า $k \geq 0$ แต่ถ้าค่าที่คำนวณได้มีค่าน้อยกว่า 0 จะถูกปรับให้เป็น 0

สำหรับงานวิจัยนี้จะเป็นการนำเสนออัลกอริทึมที่มีประสิทธิภาพในการไมน์นิ่งฐานข้อมูลโดยสามารถลดการสแกนฐานข้อมูลใหม่ทั้งหมด จากแนวคิดที่มีการนำความรู้จากการไมน์นิ่ง original database ด้วยการจัดเก็บทั้งส่วนที่เป็น frequent k-itemset และส่วนที่คาดว่าจะกลายเป็น frequent k-itemset โดยอาศัยหลักการของความน่าจะเป็นเข้ามาช่วยในการกรองข้อมูลที่คาดว่าจะกลายเป็น frequent k-itemset จริงเมื่อมีการเพิ่มฐานข้อมูลใหม่เข้ามา นอกจากนี้ยังสามารถค้นหากฎความสัมพันธ์ที่เกิดจาก items ใหม่ที่เกิดขึ้นในฐานข้อมูลใหม่ได้อย่างถูกต้องสมบูรณ์

โดยงานวิจัยนี้จะแบ่งการทำงานออกเป็น 2 ส่วนสำคัญดังนี้

1. การค้นหา frequent itemset และ expected frequent itemset ใน original database

ด้วยค่าทางสถิติของ items นั้นที่เกิดขึ้นจริงในฐานข้อมูลจะนำมาใช้ในการทำนายความน่าจะเป็นที่ items นั้นๆ จะเกิดขึ้น จากการคำนวณในสมการ (3) เมื่อมีการกำหนดขนาดของ incremental database ที่เพิ่มเข้ามาแล้วนั้น อัลกอริทึมนี้ได้มีการนำเสนอค่าที่กำหนดโดยผู้ใช้เพิ่มอีก 1 ค่าเพื่อใช้ในการพิจารณา itemset ที่คาดว่าจะเป็น frequent itemset ใน updated database เรียกค่านี้ว่า minimum expected frequent และเรียก itemset ที่มีค่าความน่าจะเป็นจากการคำนวณและมีค่าความน่าจะเป็นผ่าน minimum expected frequent itemsetนี้ว่า expected frequent itemset โดย expected frequent itemset นี้จะมีค่า support count น้อยกว่า minimum support threshold แต่มีค่า support count มากกว่าค่า minimum expected frequent itemset ในส่วนของ original database จะเก็บ itemset ที่เป็น frequent itemset และ Expected frequent itemset เพื่อลดการสแกน original database สำหรับ itemset ที่เป็น expected frequent itemset ทำให้การ update ข้อมูลต่างๆ ใช้เวลาน้อยลง

นอกจากนี้เมื่อเราทราบค่า incremental database ที่แน่นอน ทำให้เราสามารถคำนวณค่าความน่าจะเป็นที่ยอมรับได้ว่า itemset นี้มีโอกาสที่จะกลายมาเป็น frequent itemset ใน updated database นั้นควรจะต้องเกิดใน original database เท่าไร ซึ่งในที่นี้จะเรียก itemset ที่เรายอมรับนี้ว่า Promising Frequent Itemset ทั้งนี้เพื่อไม่ให้มีการเก็บจำนวน itemsetนี้มากเกินไปเราจะกำหนด user defined อีก 1 ตัวคือค่าความน่าจะเป็นที่ยอมรับได้ หรือ P เมื่อมีการเพิ่ม incremental database เข้าไปหรือใน updated database เช่น ถ้ากำหนดค่าความน่าจะเป็นที่ยอมรับให้เป็น Frequent itemset ได้ ที่ P = 0.06 นั่นหมายความว่าใน 100 ครั้งจะเกิด itemset นี้อย่างน้อย 6 ครั้ง ใน updated database จะเก็บ itemset ที่มีค่าความน่าจะเป็นมากกว่าหรือเท่ากับ P =0.06 เมื่อมีการเพิ่ม incremental database เข้าไป ซึ่งจะต้องเป็น itemset ที่ค่า support count น้อยกว่า min_sup เป็น Promising frequent itemset

ตัวอย่างการคิดค่าความน่าจะเป็น ด้วยทฤษฎีเบอร์นูลลี

การหาค่าความน่าจะเป็นของ item ใน original database โดยเมื่อทราบ candidate 1-itemset และค่า support count แล้วดังรูปที่ 3.1 จะนำค่า support ที่มีไปหา ค่าความน่าจะเป็นที่จะเกิดใน updated database โดยทำการเพิ่ม incremental database ด้วย size ที่เท่าๆ กัน เช่น ถ้า |DB| =10 ,|db| = 5 , % min_sup = 40% ดังนั้น min_sup^{DB} = 4 , min_sup^{db} =2 และ min_sup^{UP} = 6

TID	List of item
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, E
10	A,C

Itemset	support
A	7
B	7
C	6
D	2
E	3

รูปที่ 3.1 ตัวอย่างของ original database และcandidate 1-itemset ของ original database

การคำนวณจะนำค่า min_sup^{UP} ลบด้วยค่า support เพื่อหาว่าค่า k ที่ต้องการใน incremental database เป็นเท่าไร ถ้าผลลบที่ได้มีค่าน้อยกว่าหรือเท่ากับ 0 (กรณีเป็น Frequent itemset หรือค่า support ที่มี มีค่ามากกว่า min_sup^{UP}) จะคิดที่ k = 0 แต่ถ้าผลลบที่ได้ มากกว่า 0 จะนำมาเข้าสูตรการคำนวณหาค่าความน่าจะเป็นดังนี้

$$P(x \geq k) = 1 - P(x \leq k)$$

$$P(x < k) = \sum_{x=0}^{k-1} \binom{n}{x} p^x (1-p)^{n-x}$$

ตัวอย่างการคำนวณค่าความน่าจะเป็นของแต่ละ item แสดงในรูปที่ 3.2

$$P(x \geq 6)_A = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{7}{10}^x \frac{3}{10}^{15-x} = 1$$

$$P(x \geq 6)_B = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{7}{10}^x \frac{3}{10}^{15-x} = 1$$

$$P(x \geq 6)_C = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{6}{10}^x \frac{4}{10}^{15-x} = 1$$

$$P(x \geq 6)_D = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{2}{10}^x \frac{8}{10}^{15-x} = 0.06$$

$$P(x \geq 6)_E = 1 - \sum_{x=0}^{5} \binom{15}{x} \frac{3}{10}^x \frac{7}{10}^{15-x} = 0.28$$

รูปที่ 3.2 ตัวอย่างการคำนวณหาค่าความน่าจะเป็นของ candidate 1-itemset

สำหรับงานวิจัยนี้เราจะกำหนดค่า user-defined อีก 1 ตัวคือ ค่า prob ที่สามารถยอมรับได้ เช่นกำหนดให้ $prob_{pl}$=0.03 หมายความว่า เรายอมรับค่าความน่าจะเป็นที่จะเกิดขึ้น ว่าใน 100 ครั้ง จะเกิด itemset นี้ 3 ครั้ง เก็บไว้เป็น expected frequent itemset

จากตัวอย่างคือ L_1 = { A,B,C} และ PL_1 = {E} เมื่อได้ค่า L และ PL 1-itemset ใน original database มาแล้วจะนำ L และ PL 1-itemset มา join กันจะได้ candidate 2-itemset เพื่อนำมาหาค่า frequent 2-itemset (L_2)และ expected frequent itemset (PL_2) ดังรูปที่ 3.3 และทำการ join L และ PL 2-itemset สำหรับ candidate 3-itemset (C_3) เพื่อนำมาหาค่า frequent 3-itemset (L_3)และ expected frequent itemset (PL_3) ดังรูปที่ 3.4

ได้ผลลัพธ์ดังนี้

[illegible]	support
AB	4
AC	4
AE	3
BC	3
BE	3
CE	1

L2	support
AB	4
AC	4
[illegible]	support
AE	3
BC	3
BE	3

รูปที่ 3.3 แสดงcandidate 2-itemset, frequent 2-itemset และ expected frequent 2-itemset

C3(DB)	support
ABC	1
ABE	3

PL3	support
ABE	3

รูปที่ 3.4 แสดงcandidate 3-itemset, frequent 3-itemset และ expected frequent 3-itemset

2. การupdate frequent itemset และ expected frequent itemsct ใน Updated database

เมื่อมี incremental database เพิ่มเข้ามาใน original database ซึ่งมีผลต่อ frequent itemset และ expected frequent itemset และยังอาจปรากฏ itemset ใหม่ได้อีกด้วย ดังนั้นขั้นตอนการ update นี้จึงเป็นขั้นตอนที่สำคัญ

ในงานวิจัยส่วนใหญ่จะต้องมีการสแกนทั้ง original database และ incremental database ดังเช่น อัลกอริทึม FUP ที่จะมีการ update ค่า support count ใหม่ที่ปรากฏใน incremental database ให้กับ frequent itemset ในขณะเดียวกันเมื่อพบว่ามี frequent itemset ใหม่เกิดขึ้นใน incremental database จะต้องสแกน original database เพื่อหาค่า support count ให้กับ frequent itemetใหม่ที่พบ เนื่องจากส่วนใหญ่ incremental database จะมีขนาดเล็ก ดังนั้น อาจเกิดพบ itemset ที่มีค่า support count มากกว่าหรือเท่ากับค่า minimum support ของ incremental database จำนวนมาก ดังนั้น การสแกน original database จะต้องสแกน เพื่อหาค่า support count ให้กับ Frequent itemset ใหม่จำนวนมาก โดยที่ Frequent itemset ใหม่นี้อาจไม่สามารถจะกลายเป็น Frequent itemset ใน updated database ได้เลย

ในบางอัลกอริทึม เช่น negative border เมื่อพบว่ามีการเปลี่ยนแปลงจาก negative border itemset มาเป็น frequent itemset ต้องทำการสแกนฐานข้อมูลทั้ง 2 ส่วน ด้วยการไมน์นิ่งใหม่ทั้งหมดซึ่งทำให้ใช้เวลาในค้นหา frequent itemset และสร้างกฎความสัมพันธ์ใหม่ที่เกิดขึ้น

สำหรับ frequent k-itemset ที่พบใหม่นี้หมายถึง itemset ที่ปรากฏใน original database แต่มีค่า support count น้อยกว่าค่า minimum support ของ original database และมีค่า support count น้อยกว่า minimum expected frequent หรือเรียกว่าเป็น small itemset ดังนั้นจึงไม่ถูกจัดเก็บในส่วนของ frequent k-itemset และ expected frequent k-itemset

ในงานวิจัยนี้ได้นำเสนอวิธีการในการลดจำนวน frequent k-itemset ใหม่ที่พบใน incremental database ที่จะต้องนำไปสแกนใน original database เพื่อหาค่า support count ที่แท้จริง แนวคิดนี้จะทำการประมาณค่าให้กับ frequent k-itemset ใหม่ที่พบด้วยค่า minimum expected frequent ที่ได้กำหนดใน original database และนำมาใช้ในการประมาณค่าของ itemset ที่คาดว่าจะกลายเป็น frequent itemset ใน updated database

เนื่องจากค่าของ frequent k-itemset ที่พบใหม่เป็น small k-itemset ดังนั้นโอกาสที่ small itemset นี้จะเกิดใน original database นั้นสามารถประมาณได้ด้วยค่าสูงสุดคือ ค่าของ minimum expected frequent ลบ 1 เมื่อทราบค่าประมาณสูงสุดของ support count ของ small itemset สามารถนำมาใช้ในการกรอง itemset ที่มีโอกาสจะกลายเป็น frequent k-itemet ใหม่ใน updated database ได้โดยการหาผลรวมของ support count ของ frequent k-itemset ใหม่ ที่พบใน incremental database กับค่าประมาณการของ minimum expected frequent -1 ถ้าผลรวมที่ได้มีค่ามากกว่าหรือเท่ากับ minimum support ของ updated database จึงจะเก็บ frequent k-itemset ใหม่นั้นเพื่อไปสแกนใน original database ทั้งนี้จะทำการหา frequent itemset สำหรับทุก k-itemset แล้วจึงนำไปสแกนใน original database ในขั้นตอนสุดท้าย

ขั้นตอนการ update frequent itemset และ expected frequent Itemset หลักๆ ดังแสดงในอัลกอริทึมรูปที่ 5 ดังนี้คือ

เมื่อมี incremental database เพิ่มเข้ามาจะเริ่มจากการปรับปรุง support count ของ frequent itemset และ expected frequent itemset สำหรับ 1-itemset ดังแสดงในบรรทัดที่ 3 รูปที่ 3.5 และส่วนของการ updated 1-itemset เริ่มจากการสแกน incremental database เพื่อหา support count สำหรับ candidate 1-itemest ของ original database และ candidate 1-itemset ของ incremental database จากการทำงานในส่วนนี้จะทำให้ทราบค่าของ Frequent 1-itemset และ Expected Frequent itemset ใน updated database ดังแสดงในบรรทัดที่ 1- 6 ของรูปที่ 3.6

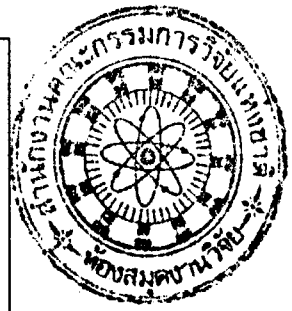
เมื่อค่า itemset ของ 1-itemset ถูกปรับปรุงแล้วจะทำการปรับปรุง Frequent itemset และ Expected Frequent itemset ของitemset ที่มากกว่า 1 itemset ถ้าพบว่ามี Frequent 1-itemset ของ updated database เกิดขึ้นใหม่จะทำการสร้าง Candidate 2-itemset โดยจะทำการ join Frequent k-itemset ของ updated database เข้าด้วยกันดังแสดงในรูปที่ 3.7 บรรทัดที่ 3 แต่ถ้า itemset มากกว่า 2 itemset ขึ้นไปจะทำการ join Frequent k-itemset (k>2) ด้วย Frequent k-itemset ใหม่ที่ปรากฏขึ้นใน incremental database ทั้ง k >=2 และ k>2 itemset จะถูกนำมาพิจารณาว่ามี frequent k-1 itemset ของ updated database เป็นสมาชิก ถ้าไม่ใช่ itemset นั้นจะถูก prune ออกไป

เมื่อได้ candidate k-itemset (k>=2) แล้วจะสแกน incremental database เพื่อหาค่า support count ของ Candidate itemset และ เพื่อ update support count สำหรับ itemset ที่เป็นสมาชิกของ Frequent itemset และ Expected Frequent itemset ดังรูปที่ 3.8

สำหรับ Candidate itemset เราจะทราบค่า support count ที่ itemset นั้นปรากฏใน incremental database ซึ่งถ้า Candidate itemset นี้มีค่า support ที่มากกว่าหรือเท่ากับค่า minimum support ของ incremental database แต่เนื่องจาก Candidate itemset นี้ไม่พบว่าเป็นสมาชิกของ Frequent itemset หรือ Expected Frequent itemset ในงานวิจัยนี้จะทำการตรวจสอบว่า candidate itemset นี้มีโอกาสที่จะกลายมาเป็น Frequent itemset ใน updated database หรือไม่โดยจะยังไม่ทำ

การสแกนเพื่อหาค่า support count ที่แท้จริงใน original database แต่จะใช้ค่า expected value ที่ได้จากการไมน์นิ่ง original database ในการประมาณค่า support count สูงสุดที่จะเกิดขึ้นในที่นี้คือ expected value -1 โดยถ้าผลรวมของ support count และ expected value-1 ของ candidate itemset มีค่ามากกว่าหรือเท่ากับค่า minimum support ของ updated database จะเก็บ candidate itemset นั้นๆ เพื่อนำไปหาค่า support ที่แท้จริงด้วยการสแกน original database ในขั้นตอนสุดท้ายดังรูปที่ 3.9

จากการทำงานของอัลกอริทึมจะพบว่า Frequent k-itemset ใหม่ (k>2) นั้น จะเกิดขึ้นได้ก็ต่อเมื่อ Frequent k-itemset นั้นอย่างน้อยจะต้องเป็น Frequent k-itemset ใน incremental database จึงจะมีโอกาสที่จะกลายเป็น Frequent k-itemset ของ updated database ได้



Algorithm1 : Main Algorithm

Input : DB, db, k, $\sigma^{UP}, \rho^{UP}, \rho^{DB}\ C_1^{DB}, F_1^{DB}, EF_1^{DB}$ *and their count*

Output : F_k^{UP}, EF_k^{UP}

1. $k = 1$
2. *if* $k = 1$
3. *Update* $1-$ *itemset*
4. $k = k + 1$
5. *else*
6. *for* $\left(k = 2; F_k^{UP} \neq \phi; k++\right)$ *do*
7. *Generate Candidate Itemset*
8. *Update* $k-$ *itemset (return m, Temp _ scanDB)*
9. *// m is the maximum itemset of Temp _ scanDB*
10. $k = k + 1$
11. *end do*
12. *end if*
13. $k = 2$
14. *while* $(Temp\_scanDBl_k \neq \phi$ *and* $(k \leq m)$ *do*
15. *Scan Original Database*$(Temp\_scanDB_k)$
16. $k = k + 1$
17. *end do*
18. *clear Temp _ scanDB*

รูปที่ 3.5 Main Algorithm

Algorithm 2 : Updating 1-itemsets

Input : $DB, db, \sigma^{UP}, \rho^{UP}, C_1^{DB}, F_1^{DB}, EF_1^{DB}, C_1^{db}$ *and their count*

Output : $F_1^{UP}, EF_1^{UP}, C_1^{UP}$ *and their count*

1. *Scan db and find count* $c(X, db)$ *for all* $X \in C_1^{DB} \cup C_1^{db}$
2. *for all* $X \in C_1^{DB} \cup C_1^{db}$ *do*
3. $c(X, UP) = c(X, DB) + c(X, db)$
4. *end do*
5. $F_1^{UP} = \left\{X \in C_1^{UP} \middle| c(X, UP) \geq \sigma^{UP}\right\}$
6. $EF_1^{UP} = \left\{X \in C_1^{UP} \middle| \rho^{UP} \leq c(X, UP) < \sigma^{UP}\right\}$

รูปที่ 3.6 Update 1-itemsest Algorithm

Algorithm 3: Generating Candidate k- itemsets

Input : $F_1^{UP}, F_{k-1}^{UP}, F_{k-1}^{db}, k$

Output : C_k^{new}

1. *if* $k = 2$ *then*
2. *if* $\left(length\left(F_1^{UP}\right) \geq 2\right)$ *then*
3. $C_2^{new} = F_1^{UP} * F_1^{UP}$
4. *end if*
5. *else if* $k > 2$ *then*
6. $C_k^{db} = F_{k-1}^{db} * F_{k-1}^{db}$
7. *for all* $X \in C_k^{db}$ *do*
8. $C_k^{new} = \left\{ X \in C_k^{db} \middle| X \in F_{k-1}^{UP} \right\}$
9. *end do*
10. *end if*

รูปที่ 3.7 Generating Candidate k- itemsets Algorithm

Algorithm 4 : Update ($k \geq 2$) itemset

Input: $DB, db, \sigma^{UP}, \rho^{UP}, \rho^{DB}, F_k^{DB}, EF_k^{DB}$ *and their count*

Ouiput: F_k^{UP} *and* EF_k^{UP}, F_k^{db}, *Temp_scanDB and their count, m*

1. *Scan db and find count* $c(X, db)$ *and* $c(Y, db)$
2. $\| X \in \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $Y \in C_k^{new}$
3. *for all* $X \in \left(F_k^{DB} \cup EF_k^{DB} \cup C_k^{new}\right)$ *do*
4. *if* $X \in \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $X \in C_k^{new}$ *then*
5. $c(X, UP) = c(X, DB) + c(X, db)$
6. *elseif* $X \in \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $X \notin C_k^{new}$ *then*
7. $c(X, UP) = c(X, DB)$
8. *elseif* $X \notin \left(F_k^{DB} \cup EF_k^{DB}\right)$ *and* $X \in C_k^{new}$ *then*
9. $Temp\_scanDB_k = \{ X \mid \left(c(X, db) + \left(\rho^{DB} - 1\right)\right) \geq \sigma^{UP} \}$
10. *end if*
11. *end do*
12. $F_k^{UP} = \left\{ X \middle| c(X, UP) \geq \sigma^{UP} \right\}$
13. $EF_k^{UP} = \left\{ X \middle| \rho^{UP} \leq c(X, UP) < \sigma^{UP} \right\}$

รูปที่ 3.8 Update ($k \geq 2$) itemset Algorithm

Algorithm 5 : Scanning an original database

Input : $Temp\_scanDB_k, \sigma^{UP}, \rho^{UP}, F_k^{UP}, EF_k^{UP}$ *and their count*

Output : F_k^{UP}, EF_k^{UP} *and their count*

1. *Scan DB and obtain count* $c(X, DB)$ *for all* $Temp\_scanDB_k$
2. *for all* $X \in Temp\_scanDB_k$ *do*
3. $\quad c(X, UP) = c(X, DB) + c(X, db)$
4. *end do*
5. $F_k^{new} = \{X \mid X \in Temp\_scanDB_k \text{ and } c(X, UP) \geq \sigma^{UP}\}$
6. $EF_k^{new} = \{X \mid X \in Temp\_scanDB_k \text{ and } \rho^{UP} \leq c(X, UP) < \sigma^{UP}\}$
7. $F_k^{UP} = F_k^{UP} \cup F_k^{new}$
8. $EF_k^{UP} = EF_k^{UP} \cup EF_k^{new}$

รูปที่ 3.9 Algorithm for Scanning an original database

จากขั้นตอนการค้นหา Frequent k-itemset ของงานวิจัยนี้เป็นการค้นหากฎความสัมพันธ์ที่มีประสิทธิภาพในกรณีที่มีการเพิ่ม incremental database เข้าไปใน original database โดยสามารถที่จะปรับปรุงค่า support count สำหรับ Frequent k-itemset และ Expected Frequent itemset และจะ prune itemset ที่ไม่อาจกลายเป็น Frequent k-itemset ออกไปทำให้ itemset ต่างๆ ได้มีการปรับปรุงให้มีความทันสมัยอยู่เสมอ และเมื่อมี Frequent k-itemset เกิดขึ้นสามารถที่จะทำการค้นหาได้อย่างถูกต้องและครบถ้วน และสามารถที่จะลดจำนวน itemset ที่จะถูกสแกนใน original database ได้ ทำให้ลดเวลาในการสแกน original database ที่มีขนาดของข้อมูลจำนวนมาก ดังแสดงการทดลองและผลการทดลองในบทที่ 4