รายงานวิจัยฉบับสมบูรณ์

การค้นหากฎความสัมพันธ์แบบเพิ่มขยายด้วยการประมาณค่าแบบ Pessimistic

Incremental Association Rule Discovery Using Pessimistic Estimation

วรพจน์ กรีสุระเดช

ได้รับทุนสนับสนุนงานวิจัยจากเงินงบประมาณแผ่นดิน ประจำปีงบประมาณ 2556
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

รายงานวิจัยฉบับสมบูรณ์

การค้นหากฎความสัมพันธ์แบบเพิ่มขยายด้วยการประมาณค่าแบบ Pessimistic

Incremental Association Rule Discovery Using Pessimistic Estimation

วรพจน์ กรีสุระเดช

ได้รับทุนสนับสนุนงานวิจัยจากเงินงบประมาณแผ่นดิน ประจำปีงบประมาณ 2556
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อโครงการ การค้นหากฎความสัมพันธ์แบบเพิ่มขยายด้วยการประมาณค่าแบบ Pessimistic
แหล่งเงิน แหล่งเงินรายได้
ประจำปีงบประมาณ 2556 **จำนวนเงินที่ได้รับการสนับสนุน** 50,000 บาท
ระยะเวลาทำการวิจัย 1 ปี 6 เดือน **ตั้งแต่** ตุลาคม 2555 **ถึง** มีนาคม 2557
ชื่อ-สกุล หัวหน้าโครงการ รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

บทคัดย่อ

งานวิจัยฉบับนี้ นำเสนออัลกอริทึมสำหรับการค้นหากฎความสัมพันธ์แบบเพิ่มขยายด้วยการประมาณค่าแบบ Pessimistic ซึ่งมีวัตถุประสงค์เพื่อใช้ในการค้นหากฎความสัมพันธ์เมื่อฐานข้อมูลมีการเปลี่ยนแปลง เมื่อมีการเพิ่มชุดข้อมูลใหม่จำนวนหนึ่งเขามาในฐานข้อมูลเดิมจะมีผลกระทบต่อกฎความสัมพันธ์ที่เคยได้ทำการค้นหาไว้แล้วก่อนหน้า นั่นคือฟรีเคว้นท์ไอเท็มเซตที่เคยถูกนำไปสร้างกฎความสัมพันธ์ในช่วงระยะเวลาก่อนหน้าอาจจะไม่เป็นไอเท็มเซ็ตตัวที่น่าสนใจสำหรับเวลาปัจจุบัน เพื่อแก้ไขปัญหาการเพิ่มขยายการค้นหากฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามา งานวิจัยฉบับนี้จึงนำแนวคิดและหลักการทำงานของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้ความน่าจะเป็น มาปรับปรุงให้สามารถประมวลผลได้โดยใช้ระยะเวลาน้อยลงแต่ยังได้ความครบถ้วนและถูกต้องของการค้นหาฟรีเคว้นท์ไอเท็มเซ็ตได้เหมือนเดิม ผลการทดลองพบว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic สามารถให้เวลาที่ใช้ในการประมวลผลที่ดีกว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น และยังช่วยลดจำนวนไอเท็มเซตที่จะถูกนำไปสแกนในฐานข้อมูลเดิม

คำสำคัญ : การค้นหากฎความสัมพันธ์แบบเพิ่มขยาย การค้นหากฎความสัมพันธ์ ดาต้าไมนิ่ง การประมาณค่าความน่าจะเป็นทวินามด้วยการแจกแจงปกติ

Research Title: Incremental Association Rule Discovery
Researcher: Associate Professor Dr.Worapoj Kreesuradej
Faculty: Faculty of Information Technology

ABSTRACT

This research paper proposes an incremental association rule discovery using the pessimistic error estimation. The objective of this research is to maintain association rules in dynamic databases. When a new set of transactions is inserted into the original database, an existing rule may be changed. The proposed algorithm is based on the probability-based algorithm. The experiment results show that the proposed algorithm has an execution times better than the probability-based algorithm. In addition, the proposed algorithm can decrease the number of the itemset rescanned in the original database.

Keywords: Incremental Association Rule Discovery, Association Rule Mining, Data Mining, Normal Approximation to Binomial

กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จได้ด้วยดี โดยการวิจัยครั้งนี้ได้รับทุนสนับสนุนการวิจัยจากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง จากแหล่งทุนประเภทรายได้ ประจำปีงบประมาณ พ.ศ. 2556 ผู้วิจัยต้องขอขอบพระคุณผู้ให้การสนับสนุนทุนวิจัยมา ณ โอกาสนี้

รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ข้อมูล เป็นทรัพยากรสำคัญและเป็นองค์ประกอบหลักที่ระบบสารสนเทศทุกระบบจำเป็นต้องมี ทุกองค์กรมีความจำเป็นต้องใช้ประโยชน์จากข้อมูลให้เกิดประสิทธิภาพสูงสุด เพื่อช่วยในการตัดสินใจดำเนินงานขององค์กรให้เกิดประสิทธิผล แต่ในปัจจุบันจะเห็นได้ว่า แม้เทคโนโลยีสารสนเทศ โดยเฉพาะอย่างยิ่งเทคโนโลยีการบันทึกข้อมูลมีความเจริญก้าวหน้ามากมายเพียงใด อัตราการใช้ประโยชน์จากข้อมูลจำนวนมหาศาลนั้นยังมีน้อยมาก ยังคงมีความรู้ (Knowledge) อีกมากมายที่ถูกซ่อนอยู่ในข้อมูลดังกล่าว แต่เรายังไม่ได้นำความรู้เหล่านั้นออกมาใช้

การทำเหมืองข้อมูล (Data mining) หรือบางครั้งเรียกว่ากระบวนการค้นหาความรู้ในฐานข้อมูล (Knowledge Discovery in Database: KDD) เป็นกระบวนการที่ใช้ในการค้นหารูปแบบหรือความสัมพันธ์ที่ซ่อนอยู่ในข้อมูลจำนวนมหาศาลโดยอัตโนมัติ ในปัจจุบันมีหลายองค์กรพยายามนำเอาหลักการของการทำเหมืองข้อมูลไปใช้ในระบบสารสนเทศเพื่อสร้างความได้เปรียบให้กับองค์กร เช่น การใช้เทคนิคของการทำเหมืองข้อมูลในการจัดกลุ่มลูกค้าเงินกู้ชั้นดีของธนาคาร หรือ การจำแนกกลุ่มลูกค้าที่มีความสามารถในการซื้อสินค้าราคาสูงจำพวกบ้านและรถยนต์ เป็นต้น

การค้นหากฎความสัมพันธ์ เป็นเทคนิคที่สำคัญของกระบวนการทำเหมืองข้อมูล (Data Mining) เพื่อค้นหาความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูล และสกัดรูปแบบข้อมูลที่น่าสนใจให้ออกมาอยู่ในภาพแบบของกฎความสัมพันธ์ if X then Y โดยมีหลักการทำงานหลัก 2 ขั้นตอนได้แก่ 1) การค้นหาไอเท็มเซ็ตที่เรียกว่าฟรีเคว้นท์ไอเท็มเซ็ต ซึ่งเป็นไอเท็มเซตที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ (minimum support) ที่ผู้ใช้กำหนด และ 2) การนำฟรีเคว้นท์ไอเท็มเซ็ตที่หาได้จากข้อแรกมาสร้างกฎความสัมพันธ์ ซึ่งกฎความสัมพันธ์ที่น่าสนใจ จะเป็นกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นมากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำ (minimum confidence) ที่ผู้ใช้กำหนด

โดยทั่วไปแล้ว อัลกอริทึมที่ได้รับการยอมรับและเป็นที่นิยมในการนำมาค้นหากฎความสัมพันธ์ คืออัลกอริทึม Apriori [1] ที่ถูกเสนอโดย Agrawal โดยในการค้นหาฟรีเคว้นไอเท็มเซ็ตของ Apriori จะประกอบด้วยขั้นตอนที่สำคัญ 2 ขั้นตอนหลัก คือ การเชื่อมไอเท็มเซ็ต (join) และการตัด (prune) ไอเท็มเซ็ตที่ไม่มีโอกาสเป็นฟรีเคว้นท์ไอเท็มเซ็ตทิ้งไป อย่างไรก็ดี เมื่อมีการเพิ่มชุดข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม การค้นหากฎความสัมพันธ์ด้วยวิธีการของ Apriori จะต้องทำประมวลผลข้อมูลทั้งหมดที่อยู่ฐานข้อมูล นั่นคือ การหาแคนดิเดตไอเท็มเซตใหม่ และสแกนหาสนับสนุนของแคนดิเดตไอเท็มเซ็ตใหม่ในทุกๆ รอบ ซึ่งทำให้เวลาที่ใช้ในการประมวลผลถูกใช้มากเกินความจำเป็นและไม่เกิดประสิทธิภาพในการทำงาน ยกตัวอย่างเช่น สมมติมีการชุดข้อมูลใหม่จำนวน 3 ชุดที่จะถูก

เพิ่มเข้าสู่ฐานข้อมูลเดิม นั่นหมายถึง หากต้องการปรับปรุงกฎความสัมพันธ์ในทุกๆ รอบที่มีการเพิ่มข้อมูลชุดใหม่เข้ามา Apriori จะต้องประมวลผลฐานข้อมูลทั้งเก่าและใหม่ตั้งแต่ต้นรวมเป็น 3 รอบ

ด้วยข้อด้อยดังกล่าวของ Apriori จึงได้มีผู้นำเสนอการปรับปรุงกฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิม Cheung และคณะ [2] ได้ เสนออัลกอริทึม FUP (Fast UPdate algorithm) ซึ่งเป็นอัลกอริทึมสำหรับการค้นหากฎความสัมพันธ์แบบเพิ่มขยาย เมื่อมีการเพิ่มข้อมูลใหม่เข้ามา โดยอาศัยองค์ความรู้เดิมจากการไมนิ่งในฐานข้อมูลเดิมมาช่วยเพิ่มประสิทธิภาพในการค้นหากฎความสัมพันธ์ นั่นคือ FUP จะใช้ฟรีเคว้นไอเท็มเซ็ตที่ได้จากการประมวลผลในฐานข้อมูลเดิม มาช่วยลดจำนวนแคนดิเดตไอเท็มเซ็ตที่จะต้องถูกนำไปสแกนในฐานข้อมูลเดิม ด้วยวิธีการประมวลผลดังกล่าวจึงทำให้ FUP ใช้เวลาในการประมวลผลน้อยกว่า Apriori

อย่างไรก็ตาม ในแต่ละรอบ k เมื่อมีการค้นพบแคนดิเดตไอเท็มเซตที่ไม่ได้เป็นสมาชิกของฟรีเคว้นท์ไอเท็มเซ็ตของฐานข้อมูลเดิม แคนดิเดตไอเท็มเซ็ตนั้นๆ จะถูกนำไปสแกนในฐานข้อมูลเดิมทุกรอบที่ k นั่นแปลว่า หากขนาดของลาร์จไอเท็มเซตในฐานข้อมูลปรับปรุงมีค่าเท่ากับ 5 (k=5) ย่อมหมายความว่า FUP จะต้องนำแคนดิเดตไอเท็มเซ็ตไปสแกนในฐานข้อมูลเดิมรวมจำนวนทั้งสิ้น 5 รอบ เช่นเดียวกับ Apriori ต่างกันตรงที่ จำนวนแคนดิเดตไอเท็มเซ็ตที่ถูกนำไปสแกนของ FUP จะมีจำนวนน้อยกว่า Apriori เท่านั้น

เพื่อลดจำนวนการสแกนฐานข้อมูลเดิมให้เหลือจำนวนรอบการสแกนที่น้อยที่สุดในงานวิจัยการค้นหากฎความสัมพันธ์แบบเพิ่มขยายโดยอาศัยหลักความน่าจะเป็น [3] ได้นำเสนอเทคนิคการประมาณค่าความน่าจะเป็นของไอเท็มเซ็ตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตสำหรับเก็บไว้เพื่อนำไปสแกนฐานข้อมูลเดิมเพียงครั้งเดียวในรอบสุดท้าย ซึ่งอัลกอริทึมดังกล่าวทำงานได้อย่างมีประสิทธิภาพให้ผลทางด้านเวลาที่รวดเร็วกว่า FUP และ Apriori อย่างไรก็ดี โดยพื้นฐานการหาความน่าจะเป็นของอัลกอริทึมดังกล่าวใช้หลักการหาความน่าจะเป็นเบอร์นูลลี ซึ่งจะมีปัญหาในการหาความน่าจะเป็นในกรณีที่ต้องคำนวณแฟคทอเรียลของจำนวนจริงที่มีค่ามาก งานวิจัยนี้จึงใช้หลักการประมาณค่าความน่าจะเป็นด้วยการเทียบค่า ทำให้ค่าความน่าจะเป็นที่ได้ไม่ตรงกับความเป็นจริง ส่งผลให้การทำนายไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตมีโอกาสคลาดเคลื่อนได้

ผู้วิจัยจึงได้ศึกษาค้นคว้าเพื่อปรับปรุงอัลกอริทึม การค้นหากฎความสัมพันธ์แบบเพิ่มขยายโดยอาศัยหลักการความน่าจะเป็น ให้สามารถทำนายไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตได้แม่นยำมากขึ้น โดยอาศัยหลักการประมาณค่าด้วยการแจกแจงปกติ และใช้แนวคิดด้าน Pessimistic และ Confidence Interval มาใช้ในการตัดสินใจการเลือกเก็บไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

งานวิจัยเรื่องการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic มีวัตถุประสงคือเพื่อ

1.2.1 เพื่อศึกษาค้นคว้าอัลกอริทึมเกี่ยวกับการเพิ่มขยายกฎความสัมพันธ์

1.2.2 เพื่อออกแบบและทดลองอัลกอริทึมกฎความสัมพันธ์แบบเพิ่มขยายด้วยการประมาณค่าแบบ Pessimistic

1.3 ทฤษฎีและแนวคิดที่ใช้ในการวิจัย

ทฤษฎีและแนวคิดต่างๆ ที่นำมาประยุกต์ใช้ในการวิจัยเกี่ยวกับการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ประกอบด้วย

1.3.1 การค้นหากฎความสัมพันธ์ (Association rule discovery)

1.3.2 การเพิ่มขยายกฎความสัมพันธ์ (Incremental association rule discovery)

1.3.3 ทฤษฎีการประมาณค่าทวินามด้วยการแจกแจงปกติ (Normal approximation to Binomial theory)

1.4 ขอบเขตของการวิจัย

งานวิจัยฉบับนี้เป็นการวิจัยเกี่ยวกับการนำเสนออัลกอริทึมที่ประยุกต์ใช้ทางด้านการเพิ่มขยายกฎความสัมพันธ์ เมื่อชุดข้อมูลใหม่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิม

1.5 ขั้นตอนของการศึกษา

งานวิจัยเรื่องการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic มีขั้นตอนของการศึกษาดังนี้

1.5.1 ศึกษาแนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้องกับงานวิจัย จากตำรา และเอกสารบทความต่างๆ

1.5.2 กำหนดหัวข้อ วัตถุประสงค์ และขอบเขตการทำงานของงานวิจัย

1.5.3 ออกแบบอัลกอริทึมใหม่ที่พัฒนาประสิทธิภาพการทำงานการเพิ่มขยายกฎความสัมพันธ์

1.5.4 พัฒนาโปรแกรมการทำงานของอัลกอริทึม ด้วยซอฟต์แวร์แมทแลป (MATLAB) รวมถึงการทดสอบการทำงานและแก้ไขข้อผิดพลาดของโปรแกรม

1.5.5 ทดลองการทำงานของอัลกอริทึมด้วยชุดข้อมูลสังเคราะห์ที่สร้างขึ้น เพื่อวัดประสิทธิภาพการทำงานของอัลกอริทึม

1.5.6 รวบรวมผลการทดลองจากการทำงานของอัลกอริทึม วิเคราะห์และสรุปผลการทดลอง

1.5.7 เรียบเรียงและจัดทำภาพเล่มวิทยานิพนธ์

1.6 นิยามศัพท์

ในงานวิจัยนี้ มีการใช้คำศัพท์เฉพาะในการศึกษาอัลกอริทึมด้านการเพิ่มขยายกฎความสัมพันธ์ เพื่อให้เข้าใจตรงกัน ผู้วิจัยได้นิยามศัพท์ที่สำคัญและนิยมใช้ในงานวิจัย ดังนี้

1.6.1 ฐานข้อมูลเดิม (Original database) หมายถึง ฐานข้อมูลที่ยังไม่มีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูล ในงานวิจัยฉบับนี้

1.6.2 ฐานข้อมูลใหม่ (Increment database) หมายถึง ฐานข้อมูลที่ประกอบด้วยข้อมูลชุดใหม่ที่จะถูกเพิ่มเข้าไปในฐานข้อมูลเดิม

1.6.3 ฐานข้อมูลปรับปรุง (Updated Database) หมายถึง ฐานข้อมูลที่ได้รับการปรับปรุงแล้ว นั่นคือ ฐานข้อมูลที่ได้รับการเพิ่มข้อมูลชุดใหม่เรียบร้อยแล้ว

1.6.4 ฟรีเคว้นท์ไอเท็มเซต (Frequent ไอเท็มเซต) หรืองานวิจัยบางฉบับจะเรียกว่า ฟรีเคว้นท์ไอเท็มเซต (Frequent ไอเท็มเซต) หมายถึง เซตของไอเท็มที่น่าสนใจ โดยเซตของไอเท็มใดๆ จะถูกเรียกว่าฟรีเคว้นท์ไอเท็มเซต ก็ต่อเมื่อ ค่าความสนับสนุนของชุดไอเท็มนั้นๆ มีค่ามากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ

1.6.5 อินไอเท็มเซต (Infrequent ไอเท็มเซต) หรืองานวิจัยบางฉบับจะเรียกว่า สมอลไอเท็มเซต (small ไอเท็มเซต) หมายถึง เซตของไอเท็มที่ไม่น่าสนใจ โดยเซตของไอเท็มใดๆ จะถูกเรียกว่าสมอลไอเท็มเซต ก็ต่อเมื่อ ค่าความสนับสนุนของชุดไอเท็มนั้นๆ มีค่าน้อยกว่าค่าสนับสนุนขั้นต่ำ

1.6.6 กฎความสัมพันธ์ (Association rule) เป็นกระบวนการค้นหารูปแบบของข้อมูลที่น่าสนใจในฐานข้อมูล แล้วแสดงออกมาในภาพของกฎความสัมพันธ์ ถ้า...แล้ว... (if-then rule)

1.6.7 แคนดิเดตไอเท็มเซต (Candidate ไอเท็มเซต) หมายถึง เซตของไอเท็มที่จะถูกนำไปคำนวณหาค่าสนับสนุน เพื่อนำมาทดสอบว่า เซตของไอเท็มใดบ้างจะจัดอยู่ในกลุ่มของฟรีเคว้นท์ไอเท็มเซต หรือสมอลไอเท็มเซต โดยทั่วไปการคำนวณหาแคนดิเดตไอเท็มเซต จะได้จากกระบวนการที่เรียกว่า join operation ยกเว้นเฉพาะไอเท็มเซตเดี่ยวที่หาได้ในรอบแรก

1.6.8 ค่าสนับสนุน (support count) หมายถึง ค่าความถี่ของไอเท็มเซต ที่เกิดขึ้นหรือปรากฏในฐานข้อมูล

1.6.9 ค่าสนับสนุนขั้นต่ำ (Minimum support) หมายถึง ค่าที่ใช้ทดสอบว่าไอเท็มเซตชุดใดจะถูกกำหนดให้เป็น ฟรีเคว้นท์ไอเท็มเซต โดยค่าสนับสนุนขั้นต่ำนี้ ผู้ใช้จะเป็นผู้กำหนดตามความเหมาะสม โดยค่าสนับสนุนขั้นต่ำจะกำหนดเป็นร้อยละของข้อมูลทรานแซคชันในฐานข้อมูลที่ปรากฏไอเท็ม A และ B ($A \cup B$) ในอีกความหมายหนึ่ง ค่าสนับสนุนขั้นต่ำจะหมายถึง ค่าความน่าจะเป็นที่ไอเท็ม A และ B จะปรากฏในฐานข้อมูลพร้อมกัน

1.6.10 ค่าความเชื่อมั่นขั้นต่ำ (Minimum confidence) หมายถึง ค่าที่ใช้ทดสอบว่ากฎความสัมพันธ์คู่ใดจะถูกกำหนดให้เป็น กฎที่น่าสนใจ (Interesting rule) หรือกฎที่เข้มแข็ง (Strong rule) โดยค่าความเชื่อมั่นขั้นต่ำนี้ ผู้ใช้จะเป็นผู้กำหนดตามความเหมาะสม และนิยมกำหนดเป็นร้อยละของข้อมูลทรานแซคชันในฐานข้อมูลที่เมื่อปรากฏไอเท็ม A แล้ว จะต้องปรากฏไอเท็ม B ในทรานแซคชันเดียวกันด้วย ซึ่งจะเป็นลักษณะความน่าจะเป็นแบบมีเงื่อนไข P(B|A)

1.6.11 k-ไอเท็มเซต (k-ไอเท็มเซต) หมายถึง เซตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน k ตัว โดย ค่า $k \geq 1$ เสมอ ตัวอย่างเช่น

k = 1 จะเรียกว่า 1-ไอเท็มเซต หมายถึง เซตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน 1 ตัว นิยมเขียนอยู่ในภาพ ของ $\{I_1, I_2, I_3, ..., I_n\}$ เมื่อ $I_1, I_2, I_3, ..., I_n$ คือไอเท็ม

k = 2 จะเรียกว่า 2-ไอเท็มเซต หมายถึง เซตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน 2 ตัว นิยมเขียนอยู่ในภาพของ $\{\{I_1, I_2\}, \{I_1, I_3\}, \{I_1, I_4\}, \{I_2, I_3\}, \{I_2, I_4\}, \{I_3, I_4\}\}$

k = 3 จะเรียกว่า 3-ไอเท็มเซต หมายถึง เซตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน 3 ตัว นิยมเขียนอยู่ในภาพของ $\{\{I_1, I_2, I_3\}, \{I_1, I_2, I_4\}, \{I_1, I_3, I_4\}, \{I_2, I_3, I_4\}\}$

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในงานวิจัยและงานวิจัยที่เกี่ยวข้อง

การค้นหากฎความสัมพันธ์ เป็นกระบวนการสำคัญของการทำเหมืองข้อมูล โดยจะค้นหารูปแบบของชุดข้อมูลที่น่าสนใจระหว่างรายการต่างๆ ทั้งหมดที่จัดเก็บไว้ในฐานข้อมูลมาใช้ในการทำนายความสัมพันธ์ระหว่างข้อมูลด้วยการสร้างให้อยู่ในภาพของกฎความสัมพันธ์ ซึ่งช่วยในการตัดสินใจ การวางแผนและการบริหารได้

ในบทนี้ จะกล่าวถึงแนวคิดและทฤษฎีพื้นฐานต่างๆ เกี่ยวข้องกับงานวิจัยการเพิ่มขยายกฎความสัมพันธ์ด้วยการแจกแจงแบบปกติ ซึ่งประกอบด้วย

1. การค้นหากฎความสัมพันธ์
2. การเพิ่มขยายกฎความสัมพันธ์
3. ทฤษฎีการประมาณค่าทวินามด้วยการแจกแจงปกติ

โดยในแต่ละแนวคิดและทฤษฎีต่างๆ รายละเอียดดังนี้

2.1 การค้นหากฎความสัมพันธ์

แนวคิดเรื่องการค้นหากฎความสัมพันธ์ ถูกนำเสนอขึ้นมาครั้งแรกในปี ค.ศ. 1993 โดย Agrawal et al. [4] เพื่อใช้ในการค้นรูปแบบความสัมพันธ์ระหว่างไอเท็ม (item) ในชุดข้อมูลที่เป็นลักษณะชุดข้อมูลแบบรายการ (Transactional dataset) ทั้งนี้ตัวแบบทางคณิตศาสตร์ที่ได้ถูกนำเสนอเพื่อใช้ในกระบวนการค้นหากฎความสัมพันธ์ เป็นดังนี้

กำหนดให้

I หมายถึง เซตของไอเท็ม โดย $I = \{I_1, I_2, I_3, ..., I_n\}$

T หมายถึง ทรานแซคชัน (transaction) หรือระเบียนรายการข้อมูล โดยแต่ละทรานแซคชั่น ประกอบด้วยเซ็ตของไอเท็ม นั่นคือ $T \subseteq I$ และ ทรานแซคชั่น T แต่ละรายการ จะสัมพันธ์กับตัวระบุ (identifier) ที่เรียกว่า TID (Transaction Identifier) นอกจากนี้ แต่ละไอเท็มจะต้องมีคุณลักษณะเป็นตัวแปรทวิ (binary variable) นั่นคือ มีค่าได้ 2 ค่า ได้แก่ ซื้อและไม่ซื้อ หรือการปรากฏ/ไม่ปรากฏไอเท็มเซ็ตในฐานข้อมูล เป็นต้น ทั้งนี้จำนวนการซื้อของแต่ละไอเท็มจะไม่ถูกนำมาพิจารณา เช่น กำหนดให้ไอเท็ม I_1 แทน ขนมปัง ในแต่ละรายการข้อมูลที่มีการซื้อขนมปัง จะนับว่า มีการซื้อขนมปังจำนวน 1 ครั้ง โดยไม่สนใจว่า ในรายการนั้นจะมีการซื้อขนมปัง จำนวนมากกว่า 1 ถุงหรือไม่

D หมายถึง เซตของทรานแซคชันในฐานข้อมูล

สมมติกำหนดให้ไอเท็ม X เป็นไอเท็มในเกิดขึ้นในทรานแซคชันหนึ่งๆ นั่นคือ $X \subseteq T$

ลักษณะของกฎความสัมพันธ์จะเป็นกฎความสัมพันธ์แบบ IF...THEN ซึ่งจะแสดงในภาพของ $X \Rightarrow Y$ โดย $X \subset I$, $Y \subset I$ และ $X \cap Y = \varnothing$

ไอเท็มเซตใดๆ ที่มีโอกาสจะนำไปสร้างกฎความสัมพันธ์ $X \Rightarrow Y$ จะต้องมีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำ หรือ minsup ที่ผู้ใช้กำหนด

ส่วนกฎความสัมพันธ์ $X \Rightarrow Y$ ใดๆ จะถูกเรียกว่าเป็นกฎที่น่าสนใจ จะต้องมีค่าความเชื่อมั่นมากกว่าค่าความเชื่อมั่นขั้นต่ำ หรือ minconf ที่ผู้ใช้เป็นผู้กำหนด

ทั้งนี้ แนวคิดเกี่ยวกับค่าสนับสนุนขั้นต่ำ และค่าความเชื่อมั่นขั้นต่ำ สำหรับการค้นหากฎความสัมพันธ์ สามารถแสดงให้อยู่ในภาพของความน่าจะเป็นได้ดังนี้

$$\text{Support}(X \Rightarrow Y) = P(X \cup Y)$$

$$\text{Confidence}(X \Rightarrow Y) = P(X|Y)$$

ตัวอย่าง ในฐานข้อมูลหนึ่งมีรายการทรานแซคชันจำนวน 10 ทรานแซคชัน พบว่าไอเท็ม A และ B ปรากฏขึ้นพร้อมในทรานแซคชันเดียวกันจำนวน 6 ทรานแซคชัน ซึ่งทางงานวิจัยจะเรียกว่า ไอเท็มเซต {A,B} มีค่าสนับสนุนเท่ากับ 6 และสมมติกำหนดค่าสนับสนุนขั้นต่ำ หรือ minsup = 40% ซึ่งหมายถึง ร้อยละ 40 ของทรานแซคชันทั้งหมดในฐานข้อมูลจะปรากฏไอเท็มเซตนั้นๆ คู่กัน

จากตัวอย่าง ฐานข้อมูลมีจำนวน 10 ทรานแซคชัน ดังนั้น ค่า minsup จะเท่ากับ 40% x 10 = 4 ไอเท็ม A และ B เกิดขึ้นพร้อมกันจำนวน 6 ทรานแซคชัน ซึ่งมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ นั่นคือ 6 > minsup ดังนั้น ไอเท็ม {A,B} จึงเรียกว่าฟรีเคว้นท์ไอเท็มเซต ซึ่งจะถูกนำไปสร้างเป็นกฎความสัมพันธ์ในลำดับถัดไป

จากที่ได้กล่าวข้างต้น การหากฎความสัมพันธ์จะประกอบด้วยขั้นตอนหลัก 2 ขั้นตอน คือ การหาฟรีเคว้นท์ไอเท็มเซต และการสร้างกฎความสัมพันธ์ ซึ่งกระบวนของการสร้างกฎความสัมพันธ์นั้นจะสามารถสร้างกฎความสัมพันธ์ได้ก็ต่อเมื่อกระบวนการคำนวณหาฟรีเคว้นท์ไอเท็มเซตกระทำเสร็จสิ้นแล้ว สำหรับงานวิจัยทางด้านการค้นหากฎความสัมพันธ์ อัลกอริทึมยอดนิยมที่ถูกนำมาใช้ในการหากฎความสัมพันธ์ได้แก่ อัลกอริทึมอะพริโอริ ซึ่งจะอธิบายในหัวข้อถัดไป

2.1.1 อัลกอริทึมอะพริโอริ (Apriori Algorithm) [1]

อัลกอริทึมอะพริโอริ เป็นอัลกอริทึมที่ค่อนข้างมีบทบาทและเป็นที่นิยมในงานวิจัยด้านการ mining association rules อัลกอริทึมนี้จะมีการคำนวณหา ฟรีเคว้นท์ไอเท็มเซต แสดงได้ดังภาพที่ 2.1 ในการทำงานอะพริโอริ จะสแกนแต่ละทรานแซคชันในฐานข้อมูล โดยมีการวนซ้ำหลายครั้ง โดยฟรีเคว้นท์ k-ไอเท็มเซต (L_k) ที่คำนวณได้แต่ละรอบ จะคำนวณจาก แคนดิเดต k-1 ไอเท็มเซต (C_{k-1}) ซึ่งเป็นในลักษณะการทำงานของ level-wise-step ในการคำนวณหา ฟรีเคว้นท์ไอเท็มเซต ของอัลกอริทึมนี้จะแบ่งการทำงานออกเป็น 2 ขั้นตอนหลัก ได้แก่

1. ขั้นตอนการ Join (join step)

เป็นขั้นตอนในการนำ L_{k-1} ($k \geq 2$) มาทำการ join operation เพื่อสร้าง C_k ซึ่งจะใช้ในการคำนวณหา L_k ต่อในรอบถัดไป เช่น C_2 ได้มาจากการทำ join operation ระหว่าง $L_1 * L_1$ จากนั้นจึงนำ C_2 ที่คำนวณได้ไปทำการหา L_2 (โดยการเปรียบเทียบกับค่า minimum support) เมื่อได้ L_2 ก็จะทำ join operation ระหว่าง $L_2 * L_2$ เพื่อให้ได้ C_3 เพื่อนำไปคำนวณหา L_3 เป็นลำดับถัดไป และจะทำเช่นไปเรื่อยๆ จนกระทั่ง $C_k = \varnothing$ การทำ join operation แสดงได้ดังภาพที่ 2.2

2. ขั้นตอนการ prune (prune step)

เป็นขั้นตอนที่ใช้ในการตัด (prune) ไอเท็มเซต ของ C_k เพื่อพิจารณาหา ไอเท็มเซต ที่มีคุณสมบัติเป็น L_k นั่นคือ ไอเท็มเซต ใดๆ ใน C_k ที่ X.support < min_sup จะถูก prune ออกไป และ ไอเท็มเซต ที่เหลือ (ซึ่งมีค่า X.support $\geq$ min_sup) จะจัดเป็น L_k ขั้นตอนการ prune step แสดงได้ดังภาพที่ 2.3 ทั้งนี้ยังได้มีการกำหนดคุณสมบัติของ ไอเท็มเซต ไว้ดังนี้ "ถ้าซับเซต ของ k-1 ไอเท็มเซต ใดๆ ใน C_k ที่ไม่ได้เป็นสมาชิกของ L_{k-1} แล้ว ไอเท็มเซต นั้นๆ จะไม่สามารถเป็น L_k ได้ ซึ่งสามารถ prune ไอเท็มเซต นั้นๆ ออกจาก C_k ได้"

```
1)  L_1 = {large 1-itemsets};
2)  for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
3)      C_k = apriori-gen(L_{k-1}); // New candidates
4)      forall transactions t ∈ D do begin
5)          C_t = subset(C_k, t); // Candidates contained in t
6)          forall candidates c ∈ C_t do
7)              c.count++;
8)      end
9)      L_k = {c ∈ C_k | c.count ≥ minsup}
10) end
11) Answer = ⋃_k L_k;
```

ภาพที่ 2.1 การคำนวณหาฟรีเคว้นท์ไอเท็มเซตของอัลกอริทึมอะพริโอริ

```
insert into C_k
select p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}
from L_{k-1} p, L_{k-1} q
where p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2},
      p.item_{k-1} < q.item_{k-1};
```

ภาพที่ 2.2 ขั้นตอนการทำ join step

```
forall itemsets c ∈ C_k do
    forall (k-1)-subsets s of c do
        if (s ∉ L_{k-1}) then
            delete c from C_k;
```

ภาพที่ 2.3 ขั้นตอนการทำ prune step

เมื่อเสร็จสิ้นขั้นตอนการหาฟรีเคว้นท์ไอเท็มเซต แล้ว ขั้นตอนถัดไปคือการหากฎความสัมพันธ์ที่เป็นไปตามเงื่อนไขของค่าสนับสนุนขั้นต่ำ (min_sup) และ ค่าความเชื่อมั่นขั้นต่ำ (min_conf) ที่กำหนดไว้ ซึ่งค่าทั้งสองดังกล่าวจะมีช่วงระหว่าง 0 – 100% หากกฎความสัมพันธ์ใดๆ ที่เป็นไปตามค่า min_sup และ min_conf ดังกล่าว จะจัดว่ากฎนั้นเป็นกฎที่น่าสนใจ หรือกฎที่เข้มแข็ง (strong rule) ในทางกลับกัน หากกฎความสัมพันธ์ใดๆ ที่ไม่เป็นไปตามค่า min_sup และ min_conf ดังกล่าว จะจัดว่ากฎนั้นเป็นกฎที่ไม่น่าสนใจหรือเป็นกฎที่อ่อนแอ (weak rule)

ข้อดีของ Apriori

1. มีการออกแบบเทคนิคในการลดจำนวนแคนดิเดตไอเท็มเซต ด้วยหลักการตัด (prune) ไอเท็มเซตที่ไม่มีโอกาสเป็นฟรีเคว้นท์ไอเท็มเซตออกไป โดยใช้ความรู้เดิมในรอบก่อนหน้า (k-1) เข้ามาช่วย

ข้อเสียของ Apriori

1. จำเป็นต้องมีการ scan ฐานข้อมูลหลายครั้ง และมีการสร้าง candidate ไอเท็มเซต จำนวนมาก ในการคำนวณแต่ละรอบ ซึ่งทำให้ใช้เวลาในการประมวลผลค่อนข้างนาน จึงทำให้มีการคิดค้นหาอัลกอริทึมที่ช่วยลดการสร้าง candidate ไอเท็มเซต และ ลดการ scan ฐานข้อมูล

2. ไม่เหมาะสมสำหรับการสร้างกฎความสัมพันธ์ ที่ข้อมูลเป็นลักษณะ Numeric หรือ Boolean

3. หากมีการเพิ่มรายการข้อมูลใหม่ เข้าไปในฐานข้อมูล จะต้องหา กฎความสัมพันธ์ใหม่ด้วยการ scan ฐานข้อมูลใหม่ทั้งหมด

2.2 การเพิ่มขยายการค้นหากฎความสัมพันธ์

ธรรมชาติของ Dynamic Database รายการข้อมูล (transaction) จะถูกเพิ่มเข้ามาในฐานข้อมูลอยู่ตลอดเวลา จึงทำให้เกิดกฎความสัมพันธ์ใหม่ที่น่าสนใจเกิดขึ้น ขณะเดียวกัน กฎความสัมพันธ์เดิมอาจจะกลายเป็นกฎที่ไม่น่าสนใจอีกต่อไป เหตุการณ์ที่เกิดขึ้นเมื่อมีการเพิ่มข้อมูลเข้ามาใหม่ ประกอบด้วย 4 เหตุการณ์ [5] ดังนี้

1. ไอเท็มเซต ที่เป็น ฟรีเคว้นท์ไอเท็มเซต ใน ฐานข้อมูลเดิม ยังคงเป็น ฟรีเคว้นท์ไอเท็มเซต ในฐานข้อมูลใหม่เช่นเดิม

2. ไอเท็มเซต ที่เป็น ฟรีเคว้นท์ไอเท็มเซต ใน ฐานข้อมูลเดิม เปลี่ยนเป็น อินฟรีเคว้นท์ไอเท็มเซต ในฐานข้อมูลใหม่

3. ไอเท็มเซต ที่เป็น อินฟรีเคว้นท์ไอเท็มเซต ใน ฐานข้อมูลเดิม เปลี่ยนเป็น ฟรีเคว้นท์ไอเท็มเซต ในฐานข้อมูลใหม่

4. ไอเท็มเซต ที่เป็น อินฟรีเคว้นท์ไอเท็มเซต ใน ฐานข้อมูลเดิม ยังคงเป็น อินฟรีเคว้นท์ไอเท็มเซต ในฐานข้อมูลใหม่เช่นเดิม

ในหัวข้อนี้ จะกล่าวถึงทฤษฎีด้านการเพิ่มขยายกฎความสัมพันธ์ที่เกี่ยวข้องกับงานวิจัย ซึ่งมีรายละเอียด ดังนี้

2.2.1 อัลกอริทึม FUP [2]

อัลกอริทึม FUP (Fast UPdate Algorithm) เป็นงานวิจัยแรกที่นำเสนอเทคนิคการทำ incremental updating technique เพื่อ maintenance association rules เมื่อมีการเพิ่มข้อมูลใหม่เข้ามาในฐานข้อมูล การทำงานของ FUP อาศัยหลักการเดียวกับ Apriori ซึ่งจะมีการทำงานหลายรอบ โดยรอบแรกจะเริ่มตั้งแต่ 1-ไอเท็มเซต ไปจนถึง k-ไอเท็มเซต ทั้งนี้อัลกอริทึมนี้จะทำงานภายใต้การอนุมานว่า ค่า minimum support และค่า minimum confidence คงที่

ความหมายของสัญลักษณ์ต่างๆ ที่ใช้ในอัลกอริทึม FUP มีรายละเอียดดังนี้

DB หมายถึง ฐานข้อมูลเก่า

db หมายถึง ฐานข้อมูลใหม่ที่ถูกเพิ่มเข้ามา

D หมายถึง จำนวน ทรานแซคชัน ที่มีอยู่ในฐานข้อมูลเดิม

d หมายถึง จำนวน ทรานแซคชัน ที่มีอยู่ในฐานข้อมูลใหม่

s หมายถึง ค่าสนับสนุนขั้นต่ำ (minimum support)

L_k หมายถึง ฟรีเคว้นท์ k-ไอเท็มเซต ใน original database เมื่อ k=1,2,...

L'_k หมายถึง ฟรีเคว้นท์ k-ไอเท็มเซต ใน ฐานข้อมูลปรับปรุง (DB$\cup$db) เมื่อ k=1,2,...n

$X.support_D$ หมายถึง ค่าสนับสนุนของ ไอเท็ม X ในฐานข้อมูลเดิม

$X.support_d$ หมายถึง ค่าสนับสนุนของ ไอเท็ม X ในฐานข้อมูลใหม่

$X.support_{UD}$ หมายถึง ค่าสนับสนุนของ ไอเท็ม X ในฐานข้อมูลปรับปรุง

FUP จะแบ่งการทำงานออกเป็น 2 ส่วนหลักคือ First iteration และ Second iteration and beyond รายละเอียดขั้นตอนการทำงานของ FUP ทั้งสองส่วน อธิบายดังนี้

1. First iteration

การทำงานในส่วนนี้ จะเป็นการตัด (prune) ไอเท็มที่เป็น loser item และ หาไอเท็มที่เป็น winner item ซึ่งเป็น ฟรีเคว้นท์ 1-ไอเท็มเซต

1.1 scan db เพื่อทำการปรับปรุงค่า support ของไอเท็ม เพื่อ prune ไอเท็มที่เป็น loser ออกไป และหาไอเท็มที่เป็น winner โดยมีหลักการพิจารณาดังนี้

1.1.1 กรณี $X \in L_1$ ให้นำค่า support ของไอเท็ม X ใน DB และ db มารวมกัน จะได้ $X.support_{UD} = X.support_D + X.support_d$ จากนั้นทำการตรวจสอบค่า support ที่ได้โดย

ถ้า $X.support_{UD} \geq s \times (D+d)$ แสดงว่า ไอเท็ม X นั้นๆ เป็น winner item และให้ $X \in L'_1$

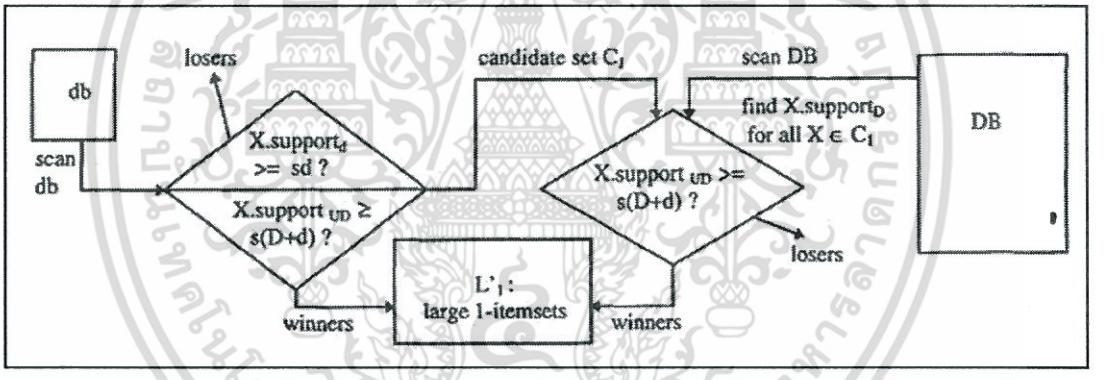
ถ้า $X.support_{UD} < s \times (D+d)$ แสดงว่า ไอเท็ม X นั้นๆ เป็น loser item และจะ prune ไอเท็ม X นั้นทิ้งไป

1.1.2 กรณี $X \notin L1$ จะมีการพิจารณา 2 ส่วน คือ

- prune ไอเท็มที่ไม่มีโอกาสเป็น L'_1 ได้ โดยพิจารณาจาก $X \notin L_1$ และ $X.support_d < s \times d$ แสดงว่าไอเท็ม X นั้น เป็น loser item จะทำการ prune ไอเท็ม X นั้นทิ้งไป ซึ่งจะช่วยลดจำนวนการ scan ไอเท็ม ในฐานข้อมูลเดิม
- ตรวจสอบไอเท็มที่มีโอกาสเป็น L'_1 โดยพิจารณาจาก $X \notin L_1$ และ $X.support_d \geq s \times d$ ให้นำ ไอเท็ม X ดังกล่าวไป scan ใน DB จากนั้นจึงนำค่า $X.support_{UD}$ ที่ได้มาตรวจสอบว่าเป็น loser item หรือ winner item โดย

- ถ้า $X.support_{UD} \geq s \times (D+d)$ แสดงว่า ไอเท็ม X นั้นๆ เป็น winner item และให้ $X \in L_1'$
- ถ้า $X.support_{UD} < s \times (D+d)$ แสดงว่า ไอเท็ม X นั้นๆ เป็น loser item จะทำการ prune ไอเท็ม X นั้นทิ้งไป

เมื่อเสร็จสิ้นกระบวนการทำงานในรอบนี้ จะได้ Large 1-ไอเท็มเซต (L'_1) ในฐานข้อมูลที่ปรับปรุงแล้ว (updated database) ทั้งนี้กระบวนการทำงานในรอบ first iteration เพื่อหา large 1-ไอเท็มเซต ของ FUP แสดงได้ดังภาพที่ 2.4 และอัลกอริทึมแสดงการทำงานในรอบ นี้ แสดงได้ดังภาพที่ 2.5



ภาพที่ 2.4 กระบวนการทำงานในรอบ First iteration ของ FUP

2. Second iteration and beyond

การทำงานในส่วนนี้จะเป็นการหา Large k-ไอเท็มเซต เมื่อ $k \geq 2$ ซึ่งจะมีส่วนหนึ่ง ที่มีหลักการทำงานคล้าย first iteration นั่นคือการหาไอเท็มที่เป็น loser ที่ไม่สามารถเป็น L_k เพื่อลดการ scan k-ไอเท็มเซต เมื่อ $k \geq 2$ ใน db โดยอาศัยแนวคิดที่ว่า “ถ้าไอเท็ม X ใดๆ ที่เป็น loser item ในการประมวลผลรอบที่ k-1 (เมื่อ $k \geq 2$) แล้ว ไอเท็มเซต ใดๆ ของ L_k ในฐานข้อมูลเดิมที่มีไอเท็ม X ดังกล่าวเป็น subset อยู่จะไม่สามารถเป็น winer item ในรอบที่ k” จากแนวคิดดังกล่าว ใน second iteration and beyond จะมีการทำงานดังนี้

2.1 หา ไอเท็มเซต ที่เป็นสมาชิกของ L_2 และ L'_2 นั่นคือ หา ไอเท็มเซต ที่เป็นฟรีเคว้นท์ไอเท็มเซต ทั้งในฐานข้อมูลเดิม และฐานข้อมูลที่ปรับปรุงแล้ว

ขั้นตอนนี้จะ prune loser item ที่ไม่สามารถเป็น L_2 เพื่อลดจำนวนไอเท็มที่จะ scan ใน db โดยพิจารณาจาก $Y = L_1 - L'_1$(ไอเท็ม Y ใดๆ ที่เป็นสมาชิกของ L_1 แต่ไม่เป็นสมาชิกของ L'_1) นั่นคือเมื่อ $X \in L_2$ ที่มีไอเท็ม Y เป็นซับเซต จะไม่สามารถเป็น ฟรีเคว้นท์ไอเท็มเซต ได้ ดังนั้น ไอเท็ม X ดังกล่าวจะถูก prune ทิ้งไป

ตัวอย่าง

$L_1 = \{A,B,C\}$ $L'_1 = \{A,C,D\}$ $L_2 = \{AB,AC\}$

$L_1 - L'_1 = \{B\}$

จากตัวอย่าง จะเห็นได้ {B} เป็นสมาชิกใน L_1 แต่ไม่เป็นสมาชิกใน L'_1 ดังนั้น ไอเท็มเซต ใด ๆ ใน L_2 ที่มี {B} เป็นสมาชิก ไอเท็มเซต นั้นๆ จะเรียกว่า loser item และถูก prune ออกไปโดยไม่ต้องนำไป scan ใน db ซึ่งจากตัวอย่าง ไอเท็มเซต ใน L_2 ที่มี {B} เป็นซับเซต คือ {AB} ดังนั้น {AB} จะถูก prune ออกจาก L_2 และ L_2 จะเหลือสมาชิกอยู่คือ {AC} ซึ่ง {AC} จะถูกนำไป scan ใน db เพื่อปรับปรุงค่า support

ในการ scan สมาชิกตัวที่เหลือใน L_2 หากพบว่า

- $X.support_{UD} \geq s \times (D+d)$ แล้วไอเท็มเซต นั้นๆ จะเรียกว่า winner item แล้วจะถูกนำไปเก็บใน L'_2
- $X.support_{UD} < s \times (D+d)$ แล้ว ไอเท็มเซต นั้นๆ จะเรียกว่า loser item ก็จะถูก prune ทิ้งไป

2.2 หา new frequent 2-ไอเท็มเซต (L'2)

ในขั้นตอนนี้จะเริ่มคำนวณ candidate 2-ไอเท็มเซต (C_2) ด้วยการ join operation ระหว่าง $L'_1 * L'_1$ เพื่อนำไป scan ใน db และหา L'2 โดยพิจารณาดังนี้

2.2.1 กรณี $X \in C_2$ และ $X \in L'_2$

ไม่ต้องนำ ไอเท็ม X ดังกล่าวไป scan ใน db เนื่องจาก X เป็นสมาชิกของ L'_2 แล้ว (ซึ่งคำนวณได้จากขั้นตอนที่ 2.1)

2.2.2 กรณี $X \in C2$ และ $X \notin L'2$

ให้นำ ไอเท็ม X ดังกล่าวไป scan ใน db แล้วตรวจสอบ

- ถ้า $X.support_d \geq s \times d$

ให้นำ ไอเท็ม X ดังกล่าวไป scan ใน DB แล้วปรับปรุงค่า support แล้วตรวจสอบหากพบว่า $X.support_{UD} \geq s \times (D+d)$ แล้ว ให้เพิ่มไอเท็ม X นั้นเป็นสมาชิกของ L'_2

- ถ้า $X.support_d < s \times d$

ให้ prune ไอเท็ม X ออกจาก C_2 และไม่ต้องนำ X ไป scan ใน DB

เมื่อเสร็จสิ้นขั้นตอนที่ 2.2 ผลลัพธ์ที่ได้คือ L'_2

2.3 ทำการวนซ้ำเช่นเดียวกับข้อที่ 2.1 เพื่อหา k-ไอเท็มเซต (k≥3)ในรอบถัดไป

จุดเด่นของ FUP

1. มีการนำผลลัพธ์จากการไมนิ่งในฐานข้อมูลเดิมมาใช้ร่วมกับฐานข้อมูลใหม่ที่เพิ่มเข้ามา

2. สามารถลดจำนวนแคนดิเดตไอเท็มเซต ที่จะนำไปใช้ scan ในฐานข้อมูลเดิม

ข้อเสียของ FUP

1. สามารถใช้งานได้ในกรณีของการเพิ่มข้อมูลใหม่เข้าไปในฐานข้อมูลเดิมเท่านั้น ยังไม่สามารถใช้ได้กับกรณีการลบ และการ modification ในฐานข้อมูลได้

2. ยังจำเป็นต้องมีการสแกนฐานข้อมูลเดิม เพื่อหา new large k-ไอเท็มเซต ในทุกรอบ k

Algorithm 1 FUP: A fast update algorithm for maintenance of association rules on database updates.

Input: (1) DB: the original database (with its size, i.e., the total number of transactions, equal to D); (2) L_k: the set of all large k-itemsets in DB, where $k = 1, \ldots, r$; (3) db: an increment database (with its size equal to d); and (4) s: the minimum support threshold.

Output: L': The set of all large itemsets in $DB \cup db$.

Method:

The 1st iteration: /* find L'_1, the set of all large 1-itemsets in $DB \cup db$ */

$W = L_1$; $C = \emptyset$; $L'_1 = \emptyset$; $P = \emptyset$;
/* W: winners, C: candidate sets,
L'_1: initialized, P: for optimization */
for_all $T \in db$ do /* scan db */
 for_all 1-itemset $X \subseteq T$ do {
  if $X \in W$ then $X.support_d$++;
  else {
   if $X \notin C$
   then { $C = C \cup \{X\}$; $X.support_d = 0$; }
   /*init the support count and add X into C */
   $X.support_d$++; }
 };
for_all $X \in W$ do /*put winners into L'_1 */
 if $X.support_{UD} \geq s \times (D + d)$
  then $L'_1 = L'_1 \cup \{X\}$;
for_all $X \in C$ do /*prune candidate sets in C */
 if $X.support_d < s \times d$
  then { $C = C - \{X\}$; $P = P \cup \{X\}$; }
  /* P will be used for optimization. */
for_all $T \in DB$ do /* scan DB */
 for_all 1-itemset $X \subseteq T$ do {
  if $X \in C$ then $X.support_D$++;
  if $X \in P$ then removes X from T;
  /* Transaction T is reduced */
 };
for_all $X \in C$ do /*put winners into L'_1*/
 if $X.support_{UD} \geq s \times (D + d)$
  then $L'_1 = L'_1 \cup \{X\}$;
return L'_1. /* end of the 1st iteration */

ภาพที่ 2.5 อัลกอริทึม FUP ใน First iteration

The k-th iteration: /* for $k = 2$ or larger, repeat this program fragment to find L'_k, the set of all large k-itemsets in the updated database, until either L'_k returned is empty or $db = \emptyset$ */

$W = L_k$; $L'_k = \emptyset$;
/* W: winners; L'_k initialized */
C = apriori-gen(L'_{k-1}) $- L_k$;
/* the size-k candidate sets */
for_all k-itemset $X \in W$ do
/* prune off losers in W */
for_all (k-1)-itemset $Y \in L_{k-1} - L'_{k-1}$ do
if $Y \subseteq X$ then { $W = W - \{X\}$; break; }
for_all $T \in db$ do { /* scan db */
for_all $X \in$ Subset(W, T) do $X.support_d$++;
/* Subset(W, T) returns all the sets in W contained in T [2] */
for_all $X \in$ Subset(C, T) do $X.support_d$++;
/* find support of all $X \in C$ */
Reduce_db(T);
/*Some items in transactions in db can be removed, discussed in next section*/
}
for_all $X \in W$ do
/*put the winners from W into L'_k */
if $X.support_{UD} \geq s \times (D + d)$
then $L'_k = L'_k \cup \{X\}$;
for_all $X \in C$ do /* prune candidate sets in C */
if $X.support_d < s \times d$ then $C = C - \{X\}$;
for_all $T \in DB$ do { /* scan DB */
for_all $X \in$ Subset(C, T) do $X.support_D$++;
Reduce_Db(T); }
/* Some items in transactions in DB can be removed, discussed in next section */
for_all $X \in C$ do
/* put the winners from C into L'_k */
if $X.support_{UD} \geq s \times (D + d)$
then $L'_k = L'_k \cup \{X\}$;
return L'_k. /* The end of the k-th iteration */

ภาพที่ 2.6 อัลกอริทึม FUP ใน Second iteration and beyond

2.2.2 Negative border [7]

อัลกอริทึม Negative border เป็นงานวิจัยที่ศึกษาเกี่ยวกับปัญหาการ maintenance association rules โดยอัลกอริทึมนี้สามารถกระทำได้ใน 2 กรณีคือ การเพิ่มข้อมูลใหม่เข้าและ และการลบข้อมูลเก่าออกไปจากฐานข้อมูลเดิม ทั้งนี้อัลกอริทึมนี้จะทำงานภายใต้การอนุมานว่า ค่าสนับสนุนขั้นต่ำ (minimum support) และค่าความเชื่อมั่นขั้นต่ำ (minimum confidence) คงที่

หลักการของ Negative border คือ จะมีการเก็บค่า C_k ทั้ง $C_k \in L_k$ และ $C_k \notin L_k$ โดย $C_k \notin L_k$ จะเรียกว่า negative border ตัวอย่างเช่น

$C_1 = \{A, B, C, D, E\}$ $L_1 = \{A, B, E\}$

ดังนั้น negative border ของ L_1 หรือ $NBd(L_1) = \{C,D\}$

จากตัวอย่าง สามารถเขียนในภาพของสมการได้ดังนี้

$NBd(L_k) = C_k - L_k$ หรือ

$C_k = L_k \cup NBd(L_k)$

ความหมายของสัญลักษณ์ที่ใช้ในอัลกอริทึม Negative border มีรายละเอียดดังนี้

DB	หมายถึง	ฐานข้อมูลเดิม
db	หมายถึง	ฐานข้อมูลใหม่
DB^+	หมายถึง	ฐานข้อมูลปรับปรุง
L^{DB}	หมายถึง	ฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลเดิม
L^{db}	หมายถึง	ฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่
L^{DB+}	หมายถึง	ฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุง
$NBd(L_k)$	หมายถึง	Negative border ของ large k-ไอเท็มเซต เมื่อ $k \geq 1$
$NBd(L^{DB})$	หมายถึง	Negative border ใน original database
$NBd(L^{db})$	หมายถึง	Negative border ใน increment database
$NBd(L^{DB+})$	หมายถึง	Negative border ใน updated database
s	หมายถึง	ไอเท็มเซต ใดๆ ในฐานข้อมูลทั้ง DB, db และ DB^+
s.count	หมายถึง	ค่าความถี่ของไอเท็ม s ที่เกิดขึ้น
$t_{DB}(s)$	หมายถึง	จำนวน transaction ใน DB ที่มีไอเท็ม s เป็นสมาชิก
$t_{db}(s)$	หมายถึง	จำนวน transaction ใน db ที่มีไอเท็ม s เป็นสมาชิก

ขั้นตอนการทำงานของ Negative border จะแบ่งการทำงานออกเป็น 2 ส่วน คือส่วนของการเพิ่มข้อมูลใหม่ (Addition of new transaction) และในส่วนของการลบข้อมูลเก่าออกไป (Deletion of existing transaction) อัลกอริทึม Negative border แสดงดังภาพที่ 2.7 โดยในแต่ละส่วนมีรายละเอียดการทำงานดังนี้

1. Addition of new transactions

อัลกอริทึม Negative border แสดงดังภาพที่ 2.7 ซึ่งมีรายละเอียดการทำงานดังนี้

1.1 ปรับปรุงค่า support ให้กับ ไอเท็มเซต ที่เป็นสมาชิกของ L_k และ $NBd(L_k)$ ในฐานข้อมูลเดิม โดยเริ่มจากสแกนทรานแซคชันที่เข้ามาในฐานข้อมูลเพื่อ คำนวณหา ฟรีเคว้นท์ไอเท็มเซต ใน db (L_k^{db}) ในขณะเดียวกันให้ทำการปรับปรุงค่า support ของ ไอเท็มเซต ที่เป็น L_k และ $NBd(L_k)$ ในฐานข้อมูลเดิมด้วย จากนั้นให้ทำการพิจารณาดังนี้

1.1.1 กรณี $s \in L^{DB}$

ถ้า $t_{DB}(s) + t_{db}(s) < min\_sup \times (t_{DB}+t_{db})$ ให้ prune ไอเท็ม s นั้นออกจาก L^{DB}

ถ้า $t_{DB}(s) + t_{db}(s) \geq min\_sup \times (t_{DB}+t_{db})$ ให้เพิ่มไอเท็ม s เข้าเป็นสมาชิกของ L^{DB+}

1.1.2 กรณี $s \in Ldb$ และ $s \notin LDB$ และ $s \in NBd(LDB)$

ถ้า $t_{DB}(s) + t_{db}(s) \geq min\_sup \times (t_{DB}+t_{db})$ ให้เพิ่มไอเท็ม s เข้าเป็นสมาชิกของ L^{DB+}

1.2 หลังจากทำการปรับปรุงค่า support ให้แก่ Lk และ NBd(Lk) ในฐานข้อมูลเดิมเรียบร้อยแล้ว (จากขั้นตอนที่ 1) ผลลัพธ์ที่ได้คือ ฟรีเคว้นท์ไอเท็มเซต ในฐานข้อมูลปรับปรุง (LDB+) จากนั้นจะทำการเปรียบความแตกต่างของ ไอเท็มเซต ใน LDB และ LDB+ ดังนี้

1.2.1 กรณี LDB = LDB+ (ไม่มีการเปลี่ยนของ ไอเท็มเซต ใน original database และ updated database) หมายถึง ไอเท็มเซต ที่เป็นสมาชิกของ negative border ไม่มีการเปลี่ยนแปลง นั่นคือ NBd(LDB) = NBd(LDB+)

1.2.2 กรณี LDB $\neq$ LDB+ (มีการเปลี่ยนของ ไอเท็มเซต ใน original database และ updated database) หมายถึง ไอเท็มเซต ที่เป็นสมาชิกของ negative border มีการเปลี่ยนแปลง นั่นคือ NBd(LDB) $\neq$ NBd(LDB+) ให้ทำการคำนวณค่า ใหม่โดยใช้ฟังก์ชั่น Negativeborder-gen(LDB+) แสดงได้ดังภาพ 2.8 โดยการนำเอา LkDB+ ไปคำนวณหา NBd(LkDB+) ในแต่ละรอบ k ตาม level-wise

1.3 จากขั้นตอนที่ 2 เมื่อคำนวณหาค่า NBd(LkDB+) เรียบร้อยแล้ว จะนำค่า LkDB $\cup$ NBd(LkDB) ของฐานข้อมูลเดิมมาเปรียบ LkDB+ $\cup$ NBd(LkDB+) เทียบกับฐานข้อมูลใหม่ เพื่อดูความเปลี่ยนแปลง ซึ่งถ้า LkDB $\cup$ NBd(LkDB) $\neq$ LkDB+ $\cup$ NBd(LkDB+) จะทำการหาค่า negative border closure ของ LDB+ และจะทำการ scan ฐานข้อมูลเดิมอีกครั้งเพื่อปรับปรุงค่า LkDB และ NBd(LkDB)

function Update-Large-Itemset(L^{DB}, $\mathcal{NB}d(L^{DB})$, *db*)

//DB and *db* denote the number of transactions in the original database and the increment database respectively.
Compute L^{db}
for each itemset $s \in L^{DB} \cup \mathcal{NB}d(L^{DB})$ *do*
  $t_{db}(s)$ = number of transactions in *db* containing s
$L^{DB+} = \phi$
for each itemset $s \in L^{DB}$ *do*
  if $(t_{DB}(s) + t_{db}(s)) \geq minsup * (DB + db)$ *then*
    $L^{DB+} = L^{DB+} \cup s$
for each itemset $s \in L^{db}$ *do*
  if $s \notin L^{DB}$ and $s \in \mathcal{NB}d(L^{DB})$ and $(t_{DB}(s) + t_{db}(s)) \geq minsup * (DB + db)$ *then*
    $L^{DB+} = L^{DB+} \cup s$
if $L^{DB} \neq L^{DB+}$ *then*
  $\mathcal{NB}d(L^{DB+})$ = negativeborder-gen(L^{DB+})
else $\mathcal{NB}d(L^{DB+}) = \mathcal{NB}d(L^{DB})$
if $L^{DB} \cup \mathcal{NB}d(L^{DB}) \neq L^{DB+} \cup \mathcal{NB}d(L^{DB+})$ *then*
  $S = L^{DB+}$
  repeat
    compute $S = S \cup \mathcal{NB}d(S)$
  until S does not grow
  $L^{DB+} = \{x \in S | support(x) \geq minsup\}$
  //support(x) is the support count of x in $DB \cup db$
  $\mathcal{NB}d(L^{DB+})$ = negativeborder-gen(L^{DB+})

ภาพที่ 2.7 อัลกอริทึมของ Negative Border

function negativeborder-gen(L)
Split L into $L_1, L_2, \ldots, L_n$ where n is the size of the largest itemset in L
forall $k = 1, 2, \ldots, n$ *do*
  compute C_{k+1} using *apriori-gen*(L_k)
$L \cup \mathcal{NB}d(L) = \bigcup_{i=2,\ldots,n+1} C_k \cup I_1$ where I_1 is the set of 1-itemsets.

ภาพที่ 2.8 ฟังก์ชั่น Negative Border-gen

2. Deletion of existing transactions

ในกรณีที่มีข้อมูลถูกลบออกจาก ฐานข้อมูลเดิม จะทำการคำนวณค่า L_k^{DB-} และ $NBd(L_k^{DB-})$ ใหม่ ด้วยการนำเอาค่า s.count ใน db มาลบออกจาก L_k^{DB} และ $NBd(L_k^{DB})$ แล้วตรวจสอบค่า $s.count_{DB-db}$ กับค่า min_sup * (DB – db) เช่นเดียวกับขั้นตอนที่ 1-3 ในส่วนของการเพิ่มข้อมูลใหม่ (Addition of new transactions)

จุดเด่นของ Negative border

1. มีการใช้ ไอเท็มเซต ที่เป็น negative border เป็นตัวตัดสินใจในการ scan ฐานข้อมูลเดิม

2. มีการ scan ฐานข้อมูลเดิมเพียงครั้งเดียว ในกรณีที่ $L_k^{DB} \cup NBd(L_k^{DB}) \neq L_k^{DB+} \cup NBd(L_k^{DB+})$

ข้อเสียของ negative border

1. ใช้ได้เฉพาะในกรณีที่ไม่มี new item เกิดขึ้น

2. การหา negative border closer ใช้เวลานานในการหา L_k เพราะต้องมีการวนซ้ำหลายรอบเพื่อให้ได้ $L = L \cup NBd(L)$ ทั้งหมด

3. ต้องมีใช้พื้นที่เก็บข้อมูลเพิ่มขึ้นเพื่อใช้เก็บค่า negative border

2.2.3 Promising frequent ไอเท็มเซต algorithm [8]

Promising Frequent ไอเท็มเซต Algorithm เป็นอัลกอริทึมที่ศึกษาด้าน maintenance of association rules ใน dynamic database โดยพยายามหลีกเลี่ยงการ scan ฐานข้อมูลเดิม หลักการของอัลกอริทึมนี้คือการใช้ค่า maximum support count ของ 1-ไอเท็มเซต ที่ได้มาจากการคำนวณในฐานข้อมูลเดิมมาทำการพยากรณ์หาค่า small ไอเท็มเซต ที่มีโอกาสเป็นฟรีเคว้นท์ไอเท็มเซต เมื่อมีทรานแซคชันใหม่เพิ่มเข้ามา ซึ่งจะเรียก ไอเท็มเซต ดังกล่าวว่า promising frequent ไอเท็มเซต

ในการคำนวณหา ฟรีเคว้นท์ไอเท็มเซต และ promising frequent ไอเท็มเซต ของอัลกอริทึมนี้จะใช้หลักการเช่นเดียวกับ Apriori แต่จะมีความแตกต่างในส่วนของการทำ join operation ซึ่งเดิม Apriori จะทำ join operation โดย $C_k = L_{k-1} * L_{k-1}$ แต่ในส่วนของ Promising Frequent ไอเท็มเซต Algorithm จะทำ join operationโดย $C_k = (L_{k-1} \cup PL_{k-1}) * (L_{k-1} \cup PL_{k-1})$

ความหมายของสัญลักษณ์ที่ใช้ใน Promising Frequent ไอเท็มเซต Algorithm มีรายละเอียดดังนี้

min_PL หมายถึง ค่าที่ใช้ตรวจสอบว่า infrequent ไอเท็มเซต ใดๆ ที่มีโอกาสเป็น frequent ไอเท็มเซต ได้ ซึ่งคำนวณได้จากสมการ (2.1)

$$\min\_sup_{DB} - \left\lceil \frac{\max sup}{totalsize} \times inc\_size \right\rceil \leq \min\_PL < \min\_sup_{DB} \qquad (2.1)$$

เมื่อกำหนดให้

$\min\_sup_{DB}$ คือ ค่า min_sup ของฐานข้อมูลเดิม

maxsup คือ ค่า support ของ ไอเท็มเซต ที่มีค่าสูงที่สุด

totalsize คือ จำนวนทรานแซคชันที่มีอยู่ในฐานข้อมูลเดิม

inc_size คือ จำนวนทรานแซคชันที่มีอยู่ในฐานข้อมูลปรับปรุง

L^{k}_{DB} หมายถึง ฟรีเคว้นท์ k-ไอเท็มเซตในฐานข้อมูลเดิม เมื่อ $k \geq 1$

PL^{k}_{DB} หมายถึง promising frequent k-ไอเท็มเซต ใน original database เมื่อ $k \geq 1$

$L^{k}_{(DB \cup db)}$ หมายถึง frequent k-ไอเท็มเซตใน updated database เมื่อ $k \geq 1$

$PL^{k}_{(DB \cup db)}$ หมายถึง promising frequent k-ไอเท็มเซตใน updated database เมื่อ $k \geq 1$

C^{k}_{DB} หมายถึง candidate k-ไอเท็มเซต ใน original database เมื่อ $k \geq 1$

C^{k}_{db} หมายถึง candidate k-ไอเท็มเซต ใน increment database เมื่อ $k \geq 1$

X.support หมายถึง ค่า support ของ ไอเท็ม X ใดๆ

อัลกอริทึมนี้แบ่งการทำงานออกเป็น 2 ส่วน คือ 1) ส่วนของการประมวลผลในส่วนของฐานข้อมูลเดิม (Original database discovery) และ 2) ส่วนของการประมวลผลใน increment database (Updating frequent and promising frequent ไอเท็มเซต) รายละเอียดการทำงานของทั้ง 2 ส่วน มีดังนี้

1. Original Database Discovery

การทำงานในส่วนนี้ มีวัตถุประสงค์เพื่อคำนวณหา frequent ไอเท็มเซต และ promising frequent ไอเท็มเซต รายละเอียดการทำงานมีดังนี้

1.1 คำนวณหาค่า min_PL ตามสมการ (1) เพื่อใช้ในการทำสอบ infrequent ไอเท็มเซต ที่มีคุณสมบัติเป็น PL^{k}_{DB}

1.2 คำนวณหา frequent 1-ไอเท็มเซต (L^{1}_{DB}) และ promising 1-frequent (PL^{1}_{DB}) ด้วยวิธีการเช่นเดียวกับ Apriori โดยค่า min_sup ที่กำหนดโดยผู้ใช้ จะใช้ทดสอบความเป็น L^{1}_{DB} ส่วนค่า min_PL ที่คำนวณได้จากขั้นตอนที่ 1.1 จะใช้ทดสอบความเป็น PL^{1}_{DB} เสร็จสิ้นขั้นตอนนี้ จะได้ ไอเท็มเซต ที่ $X \in L^{1}_{DB}$ และ $X \in PL^{1}_{DB}$

1.3 ทำ join operation เพื่อหา C^{2}_{DB} จาก $C^{2}_{DB} = (L^{1}_{DB} \cup PL^{1}_{DB1}) * (L^{1}_{DB} \cup PL^{1}_{DB1})$

1.4 ทำซ้ำข้อ 1.1 – 1.3 เพื่อหา L^{k}_{DB} และ PL^{k}_{DB} ในรอบที่ $k \geq 2$ ในรอบถัดไป ผลลัพธ์ที่ได้จากการทำงานในส่วนของ Original Database Discovery คือ frequent ไอเท็มเซต (L^{k}_{DB}) และ promising frequent ไอเท็มเซต (PL^{k}_{DB}) ของ ฐานข้อมูลเดิม

2. Updating frequent and promising frequent ไอเท็มเซต

การทำงานในส่วนนี้ มีวัตถุประสงค์เพื่อทำการปรับปรุงค่าสนับสนุนของ frequent ไอเท็มเซต และ promising frequent ไอเท็มเซต โดยผลลัพธ์จากการประมวลผลในส่วนนี้ อาจจะแสดงให้เห็นถึงผลการเปลี่ยนแปลงของ ไอเท็มเซต เช่น promising frequent ไอเท็มเซต อาจจะเปลี่ยนเป็น new frequent ไอเท็มเซต ใน increment database เป็นต้น

การทำงานในส่วนที่ 2 จะแบ่งออกเป็น 2 ส่วนย่อยได้แก่ 2.1) ส่วนการปรับปรุงค่าสนับสนุนของ ฟรีเคว้นท์ไอเท็มเซต และ promising frequent ไอเท็มเซต กรณี 1-ไอเท็มเซต และ 2.2) ส่วนการปรับปรุงค่า support ของ frequent ไอเท็มเซต และ promising frequent k-ไอเท็มเซต กรณี k-ไอเท็มเซต เมื่อ k≥2รายละเอียดการทำงานมีดังนี้

2.1 Updating frequent and promising frequent 1-ไอเท็มเซต

การทำงานในส่วนของการปรับปรุงค่า support ของ frequent ไอเท็มเซต และ promising frequent ไอเท็มเซต กรณี 1-ไอเท็มเซต แสดงได้ดังภาพที่ 2.9 ซึ่งมีรายละเอียดการทำงานดังนี้

2.1.1 ทำการ scan ฐานข้อมูลใหม่ (db) เพื่อหาค่า support ของ C^1_{db} จากนั้นให้ทำการรวมค่า support ของ C^1_{db} ที่คำนวณได้ เข้ากับ C^1_{DB} ในฐานข้อมูลเดิม (DB) จากนั้นให้ทำการพิจารณาเพื่อหา ไอเท็มเซต ที่เป็น $L^1_{(DB\cup db)}$ และ $PL^1_{(DB\cup db)}$ ดังนี้

(1) กรณี $X\in C^1_{DB}$ และ $X\notin L^1_{DB}$ หรือ $X\notin PL^1_{DB}$

ถ้า $X.support_{(DB\cup db)} \geq min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $L^1_{(DB\cup db)}$ และ Temp1

(2) กรณี $X\in C^1_{DB}$ และ $X\in L^1_{DB}$

ถ้า $X.support_{(DB\cup db)} \geq min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $L^1_{(DB\cup db)}$

ถ้า $min\_PL_{(DB\cup db)} \leq X.support_{(DB\cup db)} < min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $PL^1_{(DB\cup db)}$ ทั้งนี้ $min\_PL_{(DB\cup db)}$ คำนวณจากสมการ (2.2)

$$\min\_PL_{DB\cup db} = \min\_\sup_{DB\cup db} - \left\lceil \frac{\max\sup}{totalsize} \times inc\_size \right\rceil \qquad (2.2)$$

(3) กรณี $X\in$ C1DB และ $X\in$PL1DB

ถ้า $X.support_{(DB\cup db)} \geq min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $L^1_{(DB\cup db)}$ และ Temp1

ถ้า $min\_PL_{(DB\cup db)} \leq X.support_{(DB\cup db)} < min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $PL^1_{(DB\cup db)}$

สำหรับทั้ง 3 กรณี (1, 2 และ 3) จะเป็นการตรวจสอบ ไอเท็มเซต ในกรณีเป็น ไอเท็มเซต เดิมปรากฏอยู่ในฐานข้อมูลเดิมส่วนกรณีถัดมาคือ 2.1.1.4 จะเป็นกรณีที่เป็น ไอเท็มเซต ใหม่ที่เพิ่งปรากฏขึ้นมาใน increment database

(4) กรณี $X\notin$ C1DB (new ไอเท็มเซต)

ถ้า $X.support_{(db)} \geq min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $L^1_{(DB\cup db)}$ และ Temp1

ถ้า $min\_PL(DB \cup db) \leq X.support(db) < min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน PL1(DB∪db) และ Temp1

2.1.2 Temp1 จะเป็นตัวแปรที่ใช้เก็บ ไอเท็มเซต ที่เป็น new ฟรีเคว้นท์ไอเท็มเซต และ new promising frequent ไอเท็มเซต ในฐานข้อมูลปรับปรุง เพื่อทำการสร้าง $C^2_{(DB \cup db)}$ ตัวใหม่จากฟังก์ชัน gen_new candidate ( ) แสดงดังภาพที่ 2.10 ซึ่งจะใช้หลักการ prune และการ join operation เช่นเดียวกับ Apriori จากนั้นจะทำการเก็บค่า $C^2_{(DB \cup db)}$ ไว้ใน Temp_newCk เพื่อใช้คำนวณในส่วน Updating frequent and promising frequent k-ไอเท็มเซต เมื่อ $k \geq 2$ ซึ่งอยู่ในขั้นตอนที่ 2.2 เป็นลำดับถัดไป

2.2 Updating frequent and promising frequent k-ไอเท็มเซต เมื่อ $k \geq 2$

การทำงานในส่วนนี้เป็นการปรับปรุงค่า support ของ frequent ไอเท็มเซต และ promising frequent k-ไอเท็มเซต กรณี k-ไอเท็มเซต เมื่อ $k \geq 2$ แสดงได้ดังภาพที่ 2.12 ซึ่งมีรายละเอียดการทำงานแต่ละรอบที่ k ดังนี้

2.2.1 นำ ไอเท็มเซต ทุกๆ ตัว ที่เป็นสมาชิกของ L^2_{DB}, PL^2_{DB} และ Temp_newCk มา scan ใน db เพื่อปรับปรุงค่า support ของฐานข้อมูลปรับปรุง ($X.support_{(DB \cup db)}$)

2.2.2 จากนั้นให้ทำการพิจารณาเพื่อหา ไอเท็มเซต ที่เป็น $L^2_{(DB \cup db)}$ และ $PL^2_{(DB \cup db)}$ ดังนี้

- กรณี $X \in L^2_{DB}$

ถ้า $X.support_{(DB \cup db)} \geq min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $L^2_{(DB \cup db)}$

ถ้า $min\_PL_{(DB \cup db)} \leq X.support_{(DB \cup db)} < min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $PL^2_{(DB \cup db)}$ และ Temp1

- กรณี $X \in PL^2_{DB}$

ถ้า $X.support_{(DB \cup db)} \geq min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $L^2_{(DB \cup db)}$ และ Temp1

ถ้า $min\_PL_{(DB \cup db)} \leq X.support_{(DB \cup db)} < min\_sup * (DB+db)$ ให้เพิ่มไอเท็ม X นั้นเข้าไปใน $PL^2_{(DB \cup db)}$

- กรณี $Y \in$ Temp_newCk

ถ้า $Y.support_{db} \geq min\_sup * (db)$ ให้เพิ่มไอเท็ม Y นั้นเข้าไปใน Temp_scanDB($L^2_{(DB \cup db)}$) และ Temp1($L^2_{(DB \cup db)}$)

- ถ้า $Y.support_{db} \geq min\_PL_{(DB \cup db)}$ หรือ $Y.support_{db} \geq min\_PL_{DB}$ ให้เพิ่มไอเท็ม Y นั้นเข้าไปใน Temp_scanDB($PL^2_{(DB \cup db)}$) และ Temp1($PL^2_{(DB \cup db)}$)

ทั้งนี้ ไอเท็มเซต ทุกตัวที่ถูกเก็บไว้ใน Temp_scanDB จะถูกนำไป scan ในฐานข้อมูลเดิมในอัลกอริทึม Find_SuppcountDB (แสดงดังภาพที่ 2.11) เพื่อปรับปรุงค่า support และคำนวณหาค่า $L^{k}_{(DB\cup db)}$ และ $PL^{k}_{(DB\cup db)}$ ที่เกิดขึ้นใหม่ เมื่อ $k\geq 2$

2.2.3 จากนั้นให้นำ ไอเท็มเซต ทุกตัวที่เก็บไว้ใน Temp1 มาทำการสร้าง $C^{3}_{(DB\cup db)}$ ตัวใหม่จากฟังก์ชัน gen_new candidate ( ) เพื่อใช้คำนวณในรอบที่ k ถัดไปโดยวนซ้ำตั้งแต่ขั้นตอนที่ 2.2.1 – 2.2.3

ข้อดีของ Promising frequent ไอเท็มเซต algorithm

1. ลดจำนวนครั้งในการ scan ฐานข้อมูลเดิม
2. มีการใช้พื้นที่ในการจัดเก็บ L_k และ PL_k น้อยกว่าเดิม เมื่อเทียบกับ negative border

ข้อเสียของ Promising frequent ไอเท็มเซต algorithm

1. ใช้ได้เฉพาะในกรณีที่มีการเพิ่มข้อมูลเท่านั้น
2. ในการคำนวณค่า min_PL จะต้องมีการคาดคะเนขนาดของ increment database (|db|) ไว้ล่วงหน้า ซึ่งในสถานการณ์จริง |db| ที่คาดคะเนไว้ล่วงหน้าอาจจะไม่เท่ากับ |db| ที่เกิดขึ้นจริง ซึ่งอาจทำให้ promising frequent ไอเท็มเซต (PL) ที่คำนวณได้มีความผิดพลาด

Algorithm Updating frequent and promising frequent 1-itemset

Input :

(1) L^1_{DB}: the set of all frequent 1-itemset in original database,
(2) PL^1_{DB} : the set of all promising frequent 1-itemset original database,
(3) C^1_{DB} : candidate 1-itemset of original database,
(4) C^1_{db}: candidate 1-itemset of incremental database.

Output :

(1) $L^1_{(DB\cup db)}$: frequent itemset in updated database,
(2) $PL^1_{(DB\cup db)}$: promising frequent itemset in updated database,
(3) new frequent itemsets : new frequent itemset in updated database ,
(4) new promising frequent itemsets : new promising frequent itemset in updated database
(5) Temp_newCk : new candidate 2-itemset in updated database

```
1   C^1_db = all 1-itemsets in db with support >0
2   k=1
3   While C^1_db > 0 do
4   For each X ∈ C^1_DB do
5       X.support_(DB∪db) = X.support_DB + X.support_db
6       If (X ∉ L^1_DB or X ∈ PL^1_DB) and
7         (X.support_(DB∪db)) >= min_sup_(DB∪db) Then
8          Add X to L^1_(DB∪db)
9          Add X to temp1
10      For each X ∈ L^1_DB do
11        If X.support_(DB∪db) >= min_sup_(DB∪db) Then
12          Add X to L^1_(DB∪db)
13        Else
14          If X.support_(DB∪db) >= min_PL_(DB∪db) Then
15            Add X to PL^1_(DB∪db)
16      For each X ∈ PL^1_DB do
17        If X.support_(DB∪db) >= min_sup_(DB∪db) Then
18          Add X to L^1_(DB∪db)
19          Add X to temp1
20        Else
21          If X.support_(DB∪db) >= min_PL_(DB∪db) Then
22            Add X to PL^1_(DB∪db)
23  For each X ∉ C^1_DB do
24      Add X to C^1_(DB∪db)
25      If X.support_db >= minsup_(DB∪db) (new item in db) Then
26          Add X to L^1_(DB∪db)
27          Add X to temp1
28      Else
29        If X.support_(db) >= min_PL_(DB∪db) Then
30          Add X to PL^1_(DB∪db)
31          Add X to temp1
32   If temp1 ≠ {} Then
33      Y ∈ temp1
34      C^2_(DB∪db) (new) = gen_newcandidate (Y)
35      Clear temp1
36      Add C^2_(DB∪db) (new) to Temp_newCk
37  k=k+1
```

ภาพที่ 2.9 อัลกอริทึม ปรับปรุงค่า support ของ frequent ไอเท็มเซต และ promising frequent ไอเท็มเซต กรณี 1-ไอเท็มเซต

Algorithm Gen_newcandidate

Input :

(1)$L^k_{(DB \cup db)}$: frequent k-itemset in updated database,.

(2) $PL^k_{DB \cup db)}$: promising k-itemset in updated database.

(3) Temp1 : new frequent k-itemset in updated database

Output :

(1) new C^{k+1}: new candidate k+1-itemset in updated database.

```
1    If k <= (length(L) + length(PL))
2      For each Y ∈ Temp1
3        C^{k+1}(new) = Y *(L^k_{(DB ∪db)})∪PL^k_{(DB ∪db)})
4      For c ∈ C^{k+1}(new)
5        Delete c from C^{k+1}_{DB}(new) if all subset of c is in L^k or PL^k
```

ภาพที่ 2.10 อัลกอริทึม Gen_newcandidate

Algorithm Find_SuppcountDB

Input :

(1) $L^k_{(DB \cup db)}$ ∈ Temp_scanDB : Estimated frequent k-itemset,

(2) $PL^k_{(DB \cup db)}$ ∈ Temp_scanDB : Estimated promising frequent k-itemset

Output :

(1) $L_{(DB \cup db)}$: frequent k-itemset in updated database,

(2) $PL_{(DB \cup db)}$: promising frequent k-itemset in updated database

```
1    For each W ∈ Temp_scanDB
2      Scan DB for W
3      W.support_{(DB∪db)} = W.support_DB + W.support_db
4        If W.support_{(DB∪db)} >= min_sup_{(DB∪db)} Then
5          Add W to L^k_{(DB∪db)}
6        Else
7          If W.support_{(DB∪db)} >= min_PL_{(DB∪db)} Then
8            Add W to PL^k_{(DB∪db)}
9      Clear Temp_scamDB
```

ภาพที่ 2.11 อัลกอริทึม Find_Suppcount DB

Algorithm Update frequent and promising frequent itemsets for k>= 2 itemset

Input :

(1) L^k_{DB} : frequent k-itemset in original database,
(2) $PL^k_{DB,}$: promising frequent k-itemset in original, database
(3) Temp_newCk : new candidate k-itemset in updated database.

Output :

(1) $L^k_{(DB\cup db)}$: frequent k-itemset in updated database,
(2) $PL^k_{DB\cup db)}$: Promising frequent k-itemset in updated database,
(3) Temp_scanDB : estimated frequent k-itemset and estimated promising frequent k-itemset in updated database
(4) Temp1 : new estimated frequent k-itemset and new estimated promising frequent k-itemset in updated database
(5) Temp_newCk : new candidate k+1-itemset in updated database.

```
1   k=2
2   While k <= (length(L^k) +length(PL^k)) do
3    Scan db for ∀(L^k) , ∀(PL^k) and ∀(items) ∈ Temp_newCk
4      X.support (DB∪db) = X.support_DB + X.support_db
5    For each X ∈  L^k_DB do
6      If  X.support_(DB∪db) >= min_sup_(DB∪db) Then
7       Add  X to L^k_(DB∪db)
8      Else
9      If  X.support_(DB∪db) >= min_PL_(DB∪db) Then
10      Add  X to PL^k_(DB∪db)
11   For each X ∈  PL^k_DB do
12     If  X.support_(DB∪db) >= min_sup_(DB∪db) Then
13      Add  X to L^k_(DB∪db)
14      Add X to temp1
15     Else
16      If  X.support_(DB∪db) >= min_PL_(DB∪db) Then
17       Add  X to PL^1_(DB∪db)
18   For each Y ∈ Temp_newCk do
19    If  Y.support_(db) >= min_sup_(db) Then
20      Add Y to Temp_scanDB(L^k_(DB∪db))
21      Add Y to Temp1(L^k_(DB∪db))
22    Else
23      If Y.support_(db) >= (min_PL_(DB∪db))
24       or  Y.support_(db) >= min_PL_(DB) )Then
25        Add Y to Temp_scanDB(PL^k_(DB∪db))
26        Add Y to Temp1(PL^k_(DB∪db))
27   Clear Temp_newCk
28   If  Temp1 ≠ {} Then
29      Y ∈ Temp1
30      C^(k+1)_(DB∪db) (new) = gen_newcandidate (Y)
31      Clear Temp1
32      Add C^(k+1)_(DB∪db) (new) to Temp_newCk
33  k=k+1
    If  Temp_scanDB ≠ {}  Then Find_SuppcountDB
```

ภาพที่ 2.12 อัลกอริทึม ปรับปรุงค่า support ของ frequent ไอเท็มเซต และ promising frequent ไอเท็มเซต กรณี $k \geq 2$ ไอเท็มเซต

2.2.4 Probability-based incremental association rule discovery [3]

การเพิ่มขยายการค้นหากฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น หรืออัลกอริทึม Probability-based incremental association rule discovery เป็นอัลกอริทึมที่ถูกพัฒนาขึ้นมาโดยอาศัยหลักการหาความน่าจะเป็นด้วยทฤษฎีเบอร์นูลลีในการทำนายไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต ซึ่งไอเท็มดังกล่าวจะช่วยในการค้นหาฟรีเคว้นท์ไอเท็มเซ็ตเมื่อมีฐานข้อมูลใหม่เพิ่มเข้ามา โดยจะนำไอเท็มเซตดังกล่าวไปสแกนในฐานข้อมูลเดิมเพียงครั้งเดียวเพื่อปรับปรุงค่าสนับสนุน ซึ่งเป็นข้อดีที่ช่วยลดจำนวนครั้งของการสแกนฐานข้อมูลเดิม เมื่อเทียบกับอัลกอริทึม FUP ที่จะต้องสแกนฐานมูลทุกๆ รอบ k ซึ่งทำให้สูญเสียเวลาในการสแกนฐานข้อมูลเดิม

การทำงานของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักการความน่าจะเป็นจะแบ่งกระบวนการทำงานออกเป็น 2 กระบวนหลัก ได้แก่ (1) กระบวนการค้นหาฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลเดิมและ (2) กระบวนการปรับปรุงและค้นหาฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่ โดยทั้ง 2 กระบวนการ มี ขั้นตอนหลักที่จะต้องกระทำเหมือนกันคือการคำนวณหาค่าความน่าจะเป็นของไอเท็มที่มีโอกาสจะเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุงใหม่ ซึ่งคำนวณจาก

$$P(x \geq k) = 1 - \sum_{x=0}^{k-1} \binom{n+m}{x} . p^x (1-p)^{m+n-x} \qquad (2.3)$$

โดย $P(x \geq k)$ หมายถึงความน่าจะเป็นที่ไอเท็มเซ็ตจะมีค่าสนับสนุน x มากกว่าหรือเท่ากับค่าสนับสนุน k

k หมายถึงค่าสนับสนุนน้อยที่สุดที่จะทำให้ไอเท็มเซตนั้นกลายเป็นฟรีเคว้นท์ไอเท็มเซ็ตในฐานข้อมูลปรับปรุง

n หมายถึงขนาดหรือจำนวนทรานแซคชันในฐานข้อมูลเดิม

m หมายถึงขนาดหรือจำนวนทรานแซคชันในฐานข้อมูลใหม่

p หมายถึงความน่าจะเป็นของการเกิดไอเท็มเซตที่พิจารณาในฐานข้อมูลซึ่งคำนวณจากจำนวนทรานแซคชันทั้งหมดของฐานข้อมูลเดิมที่ปรากฏไอเท็มเซตที่พิจารณาหารด้วยจำนวนทรานแซคชันทั้งหมดของฐานข้อมูลเดิม

ตัวอย่าง กำหนดให้มีจำนวนฐานข้อมูลเดิม (n) 10 ทรานแซคชัน และจำนวนฐานข้อมูลใหม่ (m) จำนวน 5 ทรานแซคชัน กำหนดให้มีค่าสนับสนุนขั้นต่ำ 40% และในฐานข้อมูลเก่ามีไอเท็มเซต A ปรากฏอยู่จำนวน 2 ทรานแซคชัน

จากตัวอย่าง ค่า k คือค่าสนับสนุนน้อยที่สุดที่ไอเท็มเซตนั้นจะกลายเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุง คำนวณจาก $k = (n+m) * 40\% = 6$ ส่วนค่า p คำนวณจากจำนวนทรานแซคชันในฐานข้อมูลเดิมที่ปรากฏไอเท็ม A ซึ่งมีค่าเท่ากับ 2 หารด้วยจำนวนทรานแซคชันทั้งหมดของฐานข้อมูลเดิมซึ่งมีค่าเท่ากับ 10 ดังนั้น $p = 2 \div 10 = 0.2$ จากนั้นจึงนำค่า k และ p ที่ได้มาคำนวณหาความน่าจะเป็นที่ไอเท็ม A จะเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุง โดยใช้สมการที่ 2.3 คำนวณได้ดังนี้

$$P(x \geq 6)_A = 1 - \sum_{x=0}^{6-1} \binom{10+5}{x} .0.2^x (1-0.2)^{10+5-x} = 0.06$$

จากนั้นให้นำค่าความน่าจะเป็นที่คำนวณได้มาเปรียบเทียบกับค่า $prob_{PL}$ ซึ่งเป็นค่าทดสอบอีกตัวหนึ่งที่ผู้ใช้จะต้องกำหนด เช่นกำหนดให้ค่า $prob_{PL} = 0.1$ จะเห็นได้ว่า ค่าความน่าจะเป็นของไอเท็ม A ซึ่งมีค่าเท่ากับ 0.06 มีค่าน้อยกว่าค่า $prob_{PL}$ ซึ่งมีค่าเท่ากับ 0.1 ดังนั้น ไอเท็ม A จะไม่จัดว่าเป็นไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต และจะถูกตัดทิ้งไป ไม่นำไปคำนวณในรอบถัดไป

รายละเอียดขั้นตอนกระบวนการค้นหาฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลเดิมและกระบวนการปรับปรุงและค้นหาฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่ มีรายละเอียดดังนี้

1. การค้นหาฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลเดิม

1.1 สแกนฐานข้อมูลเดิมเพื่อหาค่าสนับสนุนของไอเท็มเซตทุกตัว

1.2 คำนวณหาค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตให้แก่ไอเท็มทุกตัว ด้วยการคำนวณจากสมการที่ 2.3

1.3 หาฟรีเคว้นท์ไอเท็มเซต โดยพิจารณาจากค่าสนับสนุนของไอเท็มเซตที่จะต้องมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ (min_sup) ที่ผู้ใช้กำหนด

1.4 ไอเท็มเซตตัวใดที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนับสนุนขั้นต่ำ (min_sup) และมีค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตมากกว่าค่า $prob_{PL}$ จะถูกเรียกว่า ไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต

1.5 คำนวณหาค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นทไอเท็มเซตของฐานข้อมูลเดิม แทนค่าด้วยสัญลักษณ์ ρ^{DB} ด้วยการหาค่าสนับสนุนที่มีค่าน้อยที่สุดจากไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตทั้งหมด

1.6 ตั้งแต่รอบที่ k มีค่า มากกว่าเท่ากับ 2 ขึ้นไป ให้ดำเนินการสร้างแคนดิเดตไอเท็มเซตด้วยการจอย (join) ระหว่าง $(\text{แ} \cup EF_{k-1}^{DB})*(F_{k-1}^{DB} \cup EF_{k-1}^{DB})$

1.7 จากนั้นให้ทำตามขั้นตอนที่ 1.1 – 1.6 จนกว่าจะไม่สามารถสร้างแคนดิเดตไอเท็มเซตได้อีก

2. การปรับปรุงและค้นหาฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่

ขั้นตอนการปรับปรุงและการค้นหาฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่

2.1 สแกนฐานข้อมูลใหม่เพื่อหาค่าสนับสนุนของไอเท็มเซตทุกตัว

2.2 ปรับปรุง 1-ไอเท็มเซตที่เป็นฟรีเคว้นท์ไอเท็มเซต F_1^{DB} และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท 1-ไอเท็มเซตของฐานข้อมูลเดิม EF_1^{DB}

2.3 คำนวณหาค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตให้แก่ไอเท็มทุกตัว ด้วยการคำนวณจากสมการที่ 2.3

2.4 หาฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง F_1^{UD} โดยพิจารณาจากค่าสนับสนุนของไอเท็มเซตที่จะต้องมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ (min_sup) ที่ผู้ใช้กำหนด

2.5 ไอเท็มเซตตัวใดที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนับสนุนขั้นต่ำ (min_sup) และมีค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตมากกว่าค่า $prob_{PL}$ จะถูกเรียกว่า ไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง EF_1^{UD}

2.6 คำนวณหาค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง แทนค่าด้วยสัญลักษณ์ ρ^{UD} ด้วยการหาค่าสนับสนุนที่มีค่าน้อยที่สุดจากไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตทั้งหมด

2.7 ตั้งแต่รอบที่ k มีค่า มากกว่าเท่ากับ 2 ขึ้นไป ให้ดำเนินการสร้างแคนดิเดตไอเท็มเซตด้วยการจอย (join) โดย

- กรณีที่ k =2 ให้สร้างแคนดิเดตไอเท็มเซตจากการจอยระหว่าง $F_1^{UD} * F_1^{UD}$

- กรณีที่ k > 2 ให้สร้างแคนดิเดตไอเท็มเซตจากการจอยระหว่างระหว่าง $(F_{k-1}^{db} \cup EF_{k-1}^{db}) * (F_{k-1}^{db} \cup EF_{k-1}^{db})$

2.8 นำแคนดิเดตไอเท็มเซตที่ได้มาสแกนฐานข้อมูลใหม่เพื่อคำนวณหาค่าสนับสนุน

2.9 ปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เป็นสมาชิกของ F_k^{DB} และ EF_k^{DB} แล้วคำนวณค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต เช่นเดียวกับ ข้อ 2.3

2.10 หาฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง F_k^{UD} โดยพิจารณาจากค่าสนับสนุนของไอเท็มเซตที่จะต้องมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ (min_sup) ที่ผู้ใช้กำหนด

2.11 ไอเท็มเซตตัวใดที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนับสนุนขั้นต่ำ (min_sup) และมีค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตมากกว่าค่า $prob_{PL}$ จะถูกเรียกว่า ไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง EF_k^{UD}

2.12 แคนดิเดตไอเท็มเซตตัวใดที่ไม่ได้เป็นสามาชิกของ F_k^{DB} และ EF_k^{DB} ให้นำค่าสนับสนุนของแคนดิเดตไอเท็มเซตนั้นบวกด้วยค่า ρ^{DB} -1 แล้วนำไปทดสอบกับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด หาก มีค่ามากกว่าหมายความว่าไอเท็มเซตนั้นมีโอกาสที่จะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง ดังนั้นไอเท็มเซตนี้จะถูกเก็บไว้ในตัวแปร $Temp\_scanDB$ เพื่อนำไปใช้ในการสแกนฐานข้อมูลเก่าอีกครั้งหนึ่ง หลังจากที่ได้ ฟรีเคว้นท์ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตครบทุกตัวแล้ว

2.13 ทำซ้ำข้อ 2.7 – 2.12 จนกว่าจะไม่สามารถสร้างแคนดิเดตไอเท็มเซตได้อีก

2.14 นำไอเท็มที่ถูกเก็บไว้ในตัวแปร $Temp\_scanDB$ ไปสแกนในฐานข้อมูลเดิมเพื่อปรับปรุงค่าสนับสนุน จากนั้นจึงหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตด้วยเงื่อนไขเช่นเดียวกับข้อ 2.10 และ ข้อ 2.11

ข้อดีของ Probability-based incremental association rule discovery

1. ลดจำนวนครั้งของการสแกนฐานข้อมูลเดิม โดยอัลกอริทึมนี้จะมีการสแกนฐานข้อมูลเดิมเพียงครั้งเดียว

2. ใช้หลักความน่าจะเป็นในการทำนายหาไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตทำให้มีจำนวนไอเท็มเซตที่ถูกเก็บไว้ในการประมวลผลรอบถัดไปน้อยกว่าเมื่อเปรียบเทียบกับอัลกอริทึม Negative Border

ข้อเสียของ Probability-based incremental association rule discovery

1. ในการคำนวณหาค่าความน่าจะเป็นจะมีปัญหาในการคำนวณค่าแฟคทอเรียลในกรณีที่จำนวนเต็มมีค่ามาก

2. ในการคำนวณหาค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นฟรีเคว้นท์ไอเท็มเซตจำเป็นจะต้องทราบจำนวนทรานแซคชันที่แน่นอนของฐานข้อมูลใหม่ที่จะถูกเพิ่มเข้ามา ซึ่งเป็นไปได้ว่า ในบางครั้ง จำนวนทรานแซคชันในข้อมูลใหม่แต่ละครั้งอาจจะมีจำนวนไม่เท่ากัน หรือบางครั้งอาจจะไม่ทราบจำนวนทรานแซคชันของฐานข้อมูลใหม่ ซึ่งในกรณีหลังนี้จะทำให้ไม่สามารถคำนวณค่าความน่าจะเป็นได้

2.3 ทฤษฎีการประมาณค่าทวินามด้วยการแจกแจงปกติ

จากทฤษฎีบทลิมิตของเดอมัวร์และลาปลาซ (Demoivre-Laplace Limit Theorem) [9, 10] กล่าวว่า เมื่อ n มีค่ามาก ตัวแปรสุ่มทวินามที่มีพารามิเตอร์ n และ p จะมีการแจกแจงใกล้เคียงกับการแจกแจงของตัวแปรสุ่มปกติที่มีค่าเฉลี่ยและความแปรปรวนเท่ากับของตัวแปรสุ่มทวินาม ซึ่งในปี ค.ศ.1733 เดมัวร์ได้พิสูจน์ได้ในกรณีที่ p = 0.5 ต่อมาลาปลาซได้พิสูจน์ในกรณี p ใดๆ ในปี ค.ศ. 1812 ว่า ในการแปลงตัวแปรสุ่มทวินามเป็นตัวแปรสุ่มมาตรฐาน (Standardized Random Variable) โดยนำค่าเฉลี่ย np มาลบออก แล้วหารด้วยค่าเบี่ยงเบนมาตรฐาน $\sqrt{np(1-p)}$ แล้วฟังก์ชั่นการแจกแจงของตัวแปรสุ่มมาตรฐานนี้ จะลู่เข้าสู่ฟังก์ชั่นการแจกแจงปกติมาตรฐานเมื่อ $n \to \infty$

ทฤษฎีบทที่ 1 ทฤษฎีบทลิมิตของเดอมัวร์และลาปลาซ (DeMoivre-Laplace Limit Theorem)

จากทฤษฎีแนวโน้มเข้าสู่ส่วนกลาง (Central Limit Theorem) เมื่อค่า n, np และ nq มีขนาดใหญ่มาก ตัวแปรสุ่มของการแจกแจงแบบทวินาม สามารถประมาณค่าได้ดีด้วยการแจกแจงปกติดังสมการ

$$P(X=k) = \binom{n}{k} p^k q^{n-k} \cong \frac{1}{\sqrt{2\pi npq}} e^{-\frac{(k-np)^2}{2npq}}$$

เมื่อกำหนดให้

n	แทน	จำนวนครั้งของการทดลอง
p	แทน	จำนวนครั้งที่เกิดความสำเร็จในการทดลอง n ครั้ง
q	แทน	จำนวนครั้งที่เกิดความไม่สำเร็จในการทดลอง n ครั้ง
k	แทน	จำนวนครั้งของความสำเร็จ
$P(X=k)$	แทน	ความน่าจะเป็นที่ตัวแปรสุ่ม X จะประสบความสำเร็จจำนวน k ครั้ง

สำหรับการพิสูจน์ว่าการแจกแจงปกติสามารถนำมาใช้ประมาณค่าการแจกแจงทวินามแสดงดังภาคผนวก ก

อย่างไรก็ตาม การแจกแจงทวินามนั้นเป็นการแจงแจงแบบไม่ต่อเนื่อง (Discrete distribution) ซึ่งใช้กับจำนวนเต็ม (Integer number) ในขณะที่ การแจกแจงแบบปกติเป็นการแจกแจงแบบต่อเนื่อง (Continuous distribution) ซึ่งใช้กับจำนวนจริง (real number) จึงทำให้มีโอกาสเกิดค่าคาดเคลื่อนในการประมาณค่า เพื่อลดค่าคาดเคลื่อนดังกล่าว จำเป็นต้องมีการปรับค่าโดยการแก้ความต่อเนื่อง (continuity correction)

บทแทรกที่ 1 การปรับแก้ความต่อเนื่องจากตัวแปรสุ่มทวินามซึ่งเป็นการแจกแจงแบบไม่ต่อเนื่อง โดยใช้การประมาณค่าแบบแจกแจงปกติซึ่งเป็นการแจกแจงแบบต่อเนื่อง [10]

$$P(a \le X \le b) \approx P\left(\frac{a - \frac{1}{2} - np}{\sqrt{np(1-p)}} \le z \le \frac{b + \frac{1}{2} - np}{\sqrt{np(1-p)}} \right)$$

เมื่อ

$$z = \frac{(x - \mu)}{\sigma}$$

ดังนั้น ในการประมาณค่าการแจกแจกทวินามด้วยการแจกแจงแบบปกติ การนำค่า ½ มาบวกเข้าและลบออกในการคำนวณหาค่าความน่าจะเป็น จะช่วยในการลดค่าความคาดเคลื่อนดังกล่าวได้

ในงานวิจัยนี้ได้นำหลักการประมาณค่าแบบแจกแจงปกติเข้ามาใช้ในการค้นหากฎความสัมพันธ์แบบเพิ่มขยาย เพื่อใช้ในการแก้ปัญหาการคำนวณหาค่าความน่าจะเป็นที่อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายโดยอาศัยความน่าจะเป็นประสบอันเนื่องมาจากการคำนวณค่าแฟคทอเรียล เมื่อจำนวนจริงมีค่ามาก ซึ่งจะทำให้ผลการทำนายได้ค่าที่ความน่าจะเป็นที่แม่นยำกว่า

บทที่ 3
การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic

การเพิ่มขยายกฎความสัมพันธ์เป็นอัลกอริทึมที่ใช้ในการปรับปรุงความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ เมื่อมีการเพิ่มชุดรายการข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิม อาจส่งผลให้กฎความสัมพันธ์เดิมที่มีอยู่ไม่มีความถูกต้อง นั่นคืออาจมีบางกฎที่ยังคงอยู่ บางกฎอาจจะไม่คงอยู่ และอาจจะมีกฎความสัมพันธ์ใหม่เกิดขึ้นมา สำหรับงานวิจัยนี้ จะนำเสนอวิธีแก้ปัญหาการปรับปรุงกฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามา โดยอาศัยหลักการประมาณค่าแบบการแจกแจงแบบปกติ เพื่อลดจำนวนไอเท็มเซตที่จะนำไปสแกนในฐานข้อมูลเดิม โดยในส่วนขั้นตอนการทำงานของอัลกอริทึมที่นำเสนอ รายละเอียดดังนี้

3.1 การประมาณค่าไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตด้วยการประมาณค่าแบบการแจกแจกปกติ

งานวิจัยนี้ เป็นการนำเสนอแนวคิดของการค้นหากฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ เพื่อนำมาแก้ไขปัญหาการคำนวณค่าความน่าจะเป็นของโอกาสที่จะเกิดฟรีเคว้นท์ไอเท็มเซตด้วยวิธีการหาความน่าจะเป็นแบบเบอร์นูลลี ซึ่งจะมีปัญหาในการคำนวณในกรณีที่ต้องคำนวณหาแฟคทอเรียลจากจำนวนเต็มที่มีค่ามาก ในงานวิจัยก่อนหน้า [9] แก้ไขปัญหาการคำนวณความน่าจะเป็นด้วยการเทียบค่า ซึ่งทำให้ค่าความน่าจะเป็นที่ได้ไม่ใช่ค่าความน่าจะเป็นที่แท้จริง ส่งผลให้อัลกอริทึมนั้นมีโอกาสทำนายความความน่าจะเป็นผิดพลาดไป ในหัวข้อนี้จะนำเสนอเกี่ยวกับการนำหลักการประมาณค่าแบบแจกแจงปกติมาคำนวณหาความน่าจะเป็นของการเกิดฟรีเคว้นท์ไอเท็มเซต

จากหลักการความน่าจะเป็นแบบเบอร์นูลลีที่ประกอบด้วยการทดลองจำนวน n ครั้งที่มีความเป็นอิสระต่อกัน และการเกิดเหตุการณ์ในการทดลองแต่ละครั้งจะประกอบด้วยผลสำเร็จของการทดลอง และผลความล้มเหลวของการทดลอง เมื่อนำหลักการของเบอร์นูลลี่มาประยุกต์ใช้กับการค้นหากฎความสัมพันธ์ สามารถนำมาหลักการดังกล่าวมาพิจารณาการเกิดของไอเท็มเซตในฐานข้อมูลได้คือ ให้การทดลอง n ครั้ง หมายถึง จำนวนทรานแซคชันในฐานข้อมูลจำนวน n ทรานแซคชั่น และผลการทดลองได้แก่ การเกิดหรือการปรากฏขึ้นของไอเท็มเซตนั้นแต่ละทรานแซคชั่นของฐานข้อมูล ซึ่งถ้าไอเท็มเซตที่พิจารณาปรากฏอยู่ในทรานแซคชัน จะหมายถึงผลการทดลองที่สำเร็จ ในทางกลับกัน หากไอเท็มเซตที่พิจารณาไม่ปรากฏอยู่ในทรานแซคชัน จะหมายถึงผลการทดลองที่ล้มเหลว

จาก ฟังก์ชั่นการกระจายตัวของตัวแปรสุ่มทวินาม ดังสมการที่ 1

$$f(x)=\binom{n}{x}p^{x}(1-p)^{x} \qquad , \qquad x=0,1,2,3,\ldots,n \tag{1}$$

ค่าความน่าจะเป็นด้วยการแจกแจงแบบทวินามที่ตัวแปรสุ่ม X จะให้ผลสำเร็จจำนวนน้อยกว่า k ครั้ง หรือ $P(X<k)$ ด้วยการทดลองจำนวน n ครั้ง สามารถคำนวณหาได้จากสมการที่ 2

$$P(X<k)=\sum_{x=0}^{k-1}\binom{n}{x}p^x(1-p)^{n-x} \tag{2}$$

จากสมการที่ 2 ความน่าจะเป็นที่ตัวแปรสุ่ม X จะให้ผลสำเร็จจำนวนมากกว่าหรือเท่ากับ k ครั้ง หรือ $P(X\geq k)$ ด้วยการทดลองจำนวน n ครั้ง สามารถคำนวณหาได้จากสมการที่ 3

$$P(X\geq k)=1-\sum_{x=0}^{k-1}\binom{n}{x}p^x(1-p)^{n-x} \tag{3}$$

จากสมการที่ 3 เมื่อนำมาประยุกต์ใช้ในการหากฎความสัมพันธ์ จะให้นิยามตัวแปรต่างๆ ดังนี้

n แทน จำนวนทรานแซคชันทั้งหมดในฐานข้อมูล

p แทน ความน่าจะเป็นของการเกิดไอเท็มเซตที่พิจารณาในฐานข้อมูล

q แทน ความน่าจะเป็นของการไม่เกิดไอเท็มเซตที่พิจารณาในฐานข้อมูล

k แทน จำนวนครั้งของเกิดไอเท็มเซตที่พิจารณา ในที่นี้จะหมายถึงค่า •สนับสนุนขั้นต่ำหรือ min_sup ที่ผู้ใช้เป็นกำหนดตั้งแต่เริ่มแรก

ทั้งนี้ ค่าความน่าจะเป็นของการเกิดไอเท็มเซตที่พิจารณาในฐานข้อมูล หรือค่า p สามารถคำนวณได้ดังนี้

$$p=\frac{\delta_x^{DB}}{|DB|} \tag{4}$$

เมื่อ δ_x^{DB} แทน จำนวนทรานแซคชั่นที่ปรากฏไอเท็มเซตที่พิจารณา และ $|DB|$ แทน จำนวนทรานแซคชันทั้งหมดของฐานข้อมูล

ส่วนค่าความน่าจะเป็นของการไม่เกิดไอเท็มเซตที่พิจารณาในฐานข้อมูล หรือ ค่า q สามารคำนวณได้จาก $1-p$

ด้วยปัญหาที่ค้นพบจากการวิจัยก่อนหน้า ว่าด้วยปัญหาที่เกิดจากการคำนวณค่าแฟคทอเรียลจากจำนวนเต็มที่มีค่ามากๆ ทำให้งานวิจัยฉบับนี้ได้นำเอาหลักการประมาณค่าความน่าจะเป็นแบบทวินามด้วยการแจกแจงปกติมาประยุกต์ใช้ โดยอ้างอิงจากทฤษฎีบทลิมิตของเดมัวร์และลาปลาซ และแนวคิดการปรับแก้ความต่อเนื่องจากตัวแปรสุ่มทวินามโดยใช้การประมาณค่าแบบแจกแจงปกติ ค่าความน่าจะเป็นที่ตัวแปรสุ่ม สุ่ม X จะให้ผลสำเร็จจำนวนน้อยกว่า k ครั้ง หรือ $P(X<k)$ ด้วยการทดลองจำนวน n ครั้ง ดังสมการที่ 5

$$P(X < k) \approx \sum_{x=0}^{k-1} \int_{x-0.5}^{x+.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \quad (5)$$

จากสมการที่ 5 จะได้ว่า ความน่าจะเป็นที่ตัวแปรสุ่ม X จะให้ผลสำเร็จจำนวนมากกว่าหรือเท่ากับ k ครั้ง หรือ $P(X \geq k)$ ด้วยการทดลองจำนวน n ครั้ง สามารถแสดงได้ดังสมการที่ 6

$$\begin{aligned} P(X \geq k) &\approx 1 - \sum_{x=0}^{k-1} \int_{x-0.5}^{x+0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \\ &= 1 - \int_{0-0.5}^{k-1+0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \\ &= 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \end{aligned} \quad (6)$$

จากสมการที่ 6 เมื่อนำมาประยุกต์ใช้การงานวิจัยด้านการค้นหากฎความสัมพันธ์ ค่าความน่าจะเป็นที่ไอเท็มเซตจะมีโอกาสเป็นฟรีเคว้นท์ไอเท็มเซต หรือ ไอเท็มเซตมีโอกาสที่จะมีค่าสนับสนุน X มีค่ามากกว่าค่าสนับสนุนขั้นต่ำ k จะสามารถคำนวณหาได้จากสมการที่ 7

$$\Pr obEF_X = P(X \geq k) = 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx, \quad (7)$$

เมื่อ $\mu = np$ และ $\sigma = \sqrt{npq}$ และ n คือจำนวนทรานแซคชันของฐานข้อมูลปรับปรุง |UD| ซึ่งคำนวณได้จาก |UD| = |DB|+|db|

จากสมการที่ 7 ซึ่งเป็นสมการที่จะนำมาใช้คำนวณหาค่าความน่าจะเป็นที่ไอเท็มเซตจะมีโอกาสเป็นฟรีเคว้นไอเท็มเซตในงานวิจัยนี้ ยังจำเป็นต้องทราบค่าความน่าจะเป็นของการไอเท็มเซตในฐานข้อมูลหรือค่า p เพื่อนำมาใช้ในการคำนวณ ซึ่ง สามารถคำนวณหาได้โดยนำหลักการประมาณค่าสัดส่วนกลุ่มตัวอย่าง (sample proportion) มาประยุกต์ใช้ดังนี้

กำหนดให้ ค่าสัดส่วนกลุ่มตัวอย่าง หรือ $\hat{p}$ คำนวณได้ $\frac{x}{n}$ เมื่อ x คือจำนวนครั้งที่การทดลองที่น่าสนใจจะให้ผลสำเร็จ และ n คือ จำนวนครั้งของการทดลองทั้งหมด ซึ่งในงานวิจัย x จะหมายถึงจำนวนทรานแซคชั่นที่ปรากฏไอเท็มเซตที่พิจารณา และให้ n หมายถึง จำนวนทรานแซคชันทั้งหมดในฐานข้อมูลเดิม ตัวอย่างเช่น ในฐานข้อมูลเดิมมีทรานแซคชันทั้งหมด 120 ทรานแซคชัน และ ปรากฏว่ามีไอเท็มเซต A ซึ่งเป็นไอเท็มเซตที่พิจารณาปรากฏอยู่ในฐานข้อมูลทั้งหมด 30 ทรานแซคชัน ดังนั้น $\hat{p} = \frac{x}{n} = \frac{30}{120} = 0.25$

อย่างไรก็ดีค่าสัดส่วนกลุ่มตัวอย่าง $\hat{p}$ จัดว่าเป็นค่าที่จากการประมาณค่าแบบจุด (point estimation) ซึ่งอาจทำให้ค่าความน่าจะเป็นที่ของการเกิดไอเท็มเซตที่คำนวณได้จากการประมาณค่าแบบจุดคาดเคลื่อนไปจากค่าความน่าจะเป็นที่แท้จริง ดังนั้น ในงานวิจัยนี้ ได้เสนอการนำเอาหลักการการประมาณค่าประชากรแบบช่วงและของช่วงความเชื่อมั่น (Confidence interval) เข้ามาช่วยในการเพิ่มค่าความน่าจะเป็นของการเกิดไอเท็มเซต ดังนี้

ค่าเฉลี่ยของสัดส่วนตัวอย่าง ที่ระดับความเชื่อมั่น $(1-\alpha)\%$ คือ

$$\hat{p} - z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \le p \le \hat{p} + z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \quad (8)$$

จากสมการที่ 8 การประมาณค่าเฉลี่ยของสัดส่วนตัวอย่างที่ระดับความเชื่อมั่น $(1-\alpha)\%$ นั้น จะมีช่วงขอบเขตบนและขอบเขตล่าง แต่ด้วยการประมาณค่าไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตนั้น ผู้วิจัยต้องการเพิ่มขอบเขตความน่าจะเป็นของการเกิดไอเท็มเซตดังกล่าว ดังนั้น จะได้ว่า ค่าความน่าจะเป็นที่ยอมให้เกิดไอเท็มเซต (Probability tolerance threshold of ไอเท็มเซต: ptt_X) ซึ่งจะนำไปคำนวณแทนค่า p ในสมการที่ 7 ดังนี้

$$ptt_X = p + z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} \quad (9)$$

ค่า p ในสมการที่ 9 คำนวณได้จากสมการที่ 4 ค่าความน่าจะเป็นที่ยอมให้เกิดให้เท็มเซต (ptt_X) นี้ จะช่วยเพิ่มค่าความน่าจะเป็นที่ไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตมากขึ้น ทำให้อัลกอริทึมเก็บไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นไอเท็มเซตมากขึ้น ส่งผลให้ลดจำนวนไอเท็มเซตที่จะถูกนำไปสแกนในฐานข้อมูลเดิมมีน้อยลง

3.2 อัลกอริทึมการค้นหากฎความสัมพันธ์แบบเพิ่มขยายด้วยการประมาณค่าแบบการแจงแจงปกติ

การค้นหากฎความสัมพันธ์แบบเพิ่มขยายในงานวิจัยฉบับนี้ จะใช้หลักการหากฎความสัมพันธ์โดยใช้อัลกอริทึมอะพริโอริเป็นฐาน ซึ่งเป็นการค้นหาข้อมูลตามลำดับจำนวนสมาชิกของไอเท็มเซตจากน้อยไปหามาก (k = 1, 2, 3,... n) โดยนำเอาไอเท็มเซตที่เป็นฟรีเคว้นท์ k-1 ไอเท็มเซตมาใช้ในการสร้าง แคนดิเดต k ไอเท็มเซต ด้วยขั้นตอนการจอย (join) และการตัด (prune) ไอเท็มที่ไม่สามารถเป็นฟรีเคว้นท์ไอเท็มเซตออกไป นอกจากนี้ยังได้นำหลักการความน่าจะเป็นที่ใช้ในงานวิจัยการเพิ่มขยายการค้นหากฎความสัมพันธ์โดยใช้หลักความน่าจะเป็นเข้ามาประยุกต์ร่วมด้วย ในการหาความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุง เพื่อจะนำไปใช้ในการค้นหาฟรีเคว้นท์ไอเท็มเซตในรอบถัดไป และลดจำนวนไอเท็มเซตที่จะถูกนำไปสแกนในฐานข้อมูลเดิม เพื่อช่วยลดระยะเวลาในการประมวลผล

อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ จะแบ่งการทำงานออกเป็น 2 ขั้นตอนหลักด้วยกัน ได้แก่

1. การค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลเดิม

2. การค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่และปรับปรุงให้เป็นปัจจุบัน

รายละเอียดการทำงานของอัลกอริทึมทั้ง 2 ขั้นตอนหลัก มีตัวแปรหรือสัญลักษณ์ต่างๆ กำหนดไว้ ดังนี้

ตารางที่ 3.1 สัญลักษณ์ที่ใช้ในอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ

สัญลักษณ์	*ความหมาย*
$\|DB\|$	ขนาดหรือจำนวนทรานแซคชั่นของฐานข้อมูลเดิม (Original database size)
$\|db\|$	ขนาดหรือจำนวนทรานแซคชั่นของฐานข้อมูลใหม่ (increment database size)
$\|UD\|$	ขนาดหรือจำนวนทรานแซคชั่นของฐานข้อมูลใหม่ (Updated database size)
F_k^{DB}	ฟรีเคว้นท์ k - ไอเท็มเซต ของฐานข้อมูลเดิม
F_k^{db}	ฟรีเคว้นท์ k - ไอเท็มเซต ของฐานข้อมูลใหม่
F_k^{UD}	ฟรีเคว้นท์ k - ไอเท็มเซต ของฐานข้อมูลปรับปรุง
EF_k^{DB}	ไอเท็มที่คาว่าจะเป็นฟรีเคว้นท์ k - ไอเท็มเซตในฐานข้อมูลเดิม
EF_k^{UD}	ไอเท็มที่คาว่าจะเป็นฟรีเคว้นท์ k - ไอเท็มเซตในฐานข้อมูลปรับปรุง
δ_x^{DB}	ค่าสนับสนุนของไอเท็มที่พิจารณา ในฐานข้อมูลเดิม
δ_x^{db}	ค่าสนับสนุนของไอเท็มที่พิจารณา ในฐานข้อมูลใหม่
δ_x^{UD}	ค่าสนับสนุนของไอเท็มที่พิจารณา ในฐานข้อมูลปรับปรุง
C_1^{DB}	แคนดิเดต 1 - ไอเท็มเซต ของฐานข้อมูลปรับปรุง
C_1^{db}	แคนดิเดต 1 - ไอเท็มเซต ของฐานข้อมูลใหม่
C_1^{UD}	แคนดิเดต 1 ไอเท็มเซต ของฐานข้อมูลปรับปรุง
s	ค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด
ρ^{DB}	ค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นทไอเท็มเซตของฐานข้อมูลเดิม
ρ^{UD}	ค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นทไอเท็มเซตของฐานข้อมูลปรับปรุง
Z	ค่าความเชื่อมั่น กำหนดโดยผู้ใช้
$prob_{PL}$	ค่าความน่าจะเป็นที่น้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นท์ไอเท็มเซต

3.2.1 การค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นทไอเท็มเซตในฐานข้อมูลเดิม

โดยทั่วไปฟรีเคว้นท์ไอเท็มเซตจะพิจารณาจากไอเท็มเซตใดๆ ที่มีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำที่ผู้ใช้เป็นผู้กำหนด แต่สิ่งสำคัญที่งานวิจัยนี้นำเสนอคือการค้นหาไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต (Expected frequent ไอเท็มเซต) ซึ่งแทนด้วยสัญลักษณ์ EF

การค้นหาไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของอัลกอริทึมที่นำเสนอในงานวิจัยนี้ ผู้ใช้จะต้องกำหนดค่าทดสอบขึ้นมาอีกสองค่า ได้แก่ (1) ค่าความน่าจะเป็นน้อยที่สุดที่คาดว่าไอเท็มจะกลายเป็นฟรีเคว้นท์ไอเท็มเซต ซึ่งแทนด้วยสัญลักษณ์ $prob_{PL}$ โดยค่าที่กำหนดจะอยู่ช่วงระหว่าง 0 – 1 และ (2) ค่าความเชื่อมั่น ซึ่งแทนด้วยสัญลักษณ์ Z โดยค่าที่กำหนดจะอยู่ช่วงระหว่าง 0 – 100 %

ในการค้นหาไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต ไอเท็มเซตแต่ละตัวจะถูกคำนวณค่าความน่าที่ไอเท็มเซตที่พิจารณาจะมีโอกาสเป็นฟรีเคว้นท์ไอเท็มเซต ($\mathrm{Pr}obEF_X$) ตามสมการ 7 จากนั้นค่าที่ได้จะถูกนำมาทดสอบว่ามีค่ามากกว่าค่าของ $prob_{PL}$ หรือไม่ หากมีค่ามากกว่าไอเท็มเซตนั้นจะถูกเรียกว่าไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต ส่วนค่า Z จะถูกนำไปใช้คำนวณ ค่าความน่าจะเป็นที่ยอมให้เกิดไอเท็มเซต (ptt_X) ซึ่งคำนวณได้จากสมการที่ 9

การทำงานของอัลกอริทึมการค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นทไอเท็มเซตในฐานข้อมูลเดิมแสดงดังภาพที่ 3.1

ในรอบแรก (k = 1) ไอเท็มเซตที่พิจารณาทุกตัวจะถูกนำไปสแกนในฐานข้อมูลเดิม (DB) เพื่อคำนวณหาค่าสนับสนุน (δ_x^{DB}) ของแต่ละไอเท็มเซต จากนั้นจึงนำค่าสนับสนุนที่ได้มาเปรียบเทียบกับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด s หากไอเท็มเซตใดที่มีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำ ไอเท็มนั้นจะถูกเก็บไว้ในเซตของฟรีเคว้นท์ไอเท็มเซต ส่วนไอเท็มที่ไม่ใช่ฟรีเคว้นท์ไอเท็มเซต จะถูกนำมาคำนวณหาไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต (F_k^{DB}) ดังบรรทัดที่ 4-10 โดยเริ่มจาก คำนวณค่าความน่าจะเป็นที่ยอมให้เกิดไอเท็มเซต (ptt_X) ซึ่งคำนวณได้จากสมการที่ 9 ในบรรทัดที่ 5 จากนั้นจึงนำค่า ptt_X ที่ได้ ไปคำนวณหาค่าความน่าที่ไอเท็มเซตที่พิจารณาจะมีโอกาสเป็นฟรีเคว้นท์ไอเท็มเซต ($\mathrm{Pr}obEF_X$) ในบรรทัดที่ 6

ในบรรทัดที่ 7 ไอเท็มใดที่มีค่า $\mathrm{Pr}obEF_X$ มากกว่าค่า $prob_{PL}$ ถูกนำมาคำนวณหาค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นทไอเท็มเซตของฐานข้อมูลเดิม หรือ ρ^{DB} ซึ่งคำนวณจาก ค่าสนับสนุนของไอเท็มที่มีค่า $\mathrm{Pr}obEF_X$ น้อยที่สุด จากนั้น จึงนำค่า ρ^{DB} และค่า $\mathrm{Pr}obEF_X$ มาเปรียบเทียบกัน โดยไอเท็มเซตใดๆ ที่มีค่าสนุนมากกว่าหรือเท่ากับค่า ρ^{DB} จะถูกนำไปเก็บไว้ในเซตของไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต หรือ EF_k^{DB} ส่วนไอเท็มที่มีค่า $\mathrm{Pr}obEF_X$ น้อยกว่าค่า ρ^{DB} จะถูกตัดทิ้งไป ไม่นำมาคิดคำนวณอีก

ตั้งแต่รอบที่ 2 ขึ้นไป ($k \geq 2$) ไอเท็มเซตที่ที่เป็นฟรีเคว้นท์ไอเท็มเซต F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต EF_k^{DB} จะถูกนำมาดำเนินการจอย (join) เพื่อสร้าง แคนดิเดต k – ไอเท็มเซ็ต ตามบรรทัดที่ 13 เมื่อได้แคนดิเดต k – ไอเท็มเซ็ต เรียบร้อยแล้ว ก็จะอัลกอริทึมทำการหาไอเท็มเซตที่ที่เป็นฟรีเคว้นท์ไอเท็มเซต F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต EF_k^{DB} เช่นเดียวกับรอบแรก (บรรทัดที่ 2-10)

Algorithm1: Original Mining Phase

Input: DB, |db|, $Prob_{pl}$, s, Z

Output: F_k^{DB} , EF_k^{DB} , C_1^{DB} , ρ^{DB}

1 k =1

2 scan DB for all X∈ C_k and obtain δ_x^{DB}

3 F_k^{DB} ={X | $\delta_x^{DB} \geq$ s×|DB|}

4 for {X | δ_x^{DB} < s×|DB|}

5 calculate ptt_x //using equation (9)

6 calculate probability of expected ไอเท็มเซต X ($ProbEF_X$) //using eq. 7

7 ρ^{DB} = min(δ_x^{DB} |$ProbEF_x \geq Prob_{pl}$)

8 EF_k^{DB} = {X | s×|DB| > $\delta_x^{DB} \geq$ ρ^{DB} }

9 C_1^{DB} = {X| X∈($F_1^{DB} \cup EF_1^{DB} \cup (F_1^{DB} \cup EF_1^{DB})^C$) }

10 end

11 k =2

12 while | $F_k^{DB} \cup EF_k^{DB}$ | > 1

13 C_k = ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$)*($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$)

14 repeat line 2-8

15 k++

16 end while loop

17 Return F_k^{DB} , EF_k^{DB} , C_1^{DB} , ρ^{DB}

ภาพที่ 3.1 การทำงานของอัลกอริทึมการค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นทไอเท็มเซตในฐานข้อมูลเดิม

3.2.2 การค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นทไอเท็มเซตในฐานข้อมูลใหม่และปรับปรุงให้เป็นปัจจุบัน

ในการค้นหากฎความสัมพันธ์แบบเพิ่มขยายเมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิมจะส่งผลให้ฟรีเคว้นท์ไอเท็มเซ็ตที่หาได้ก่อนหน้าเปลี่ยนแปลงไป เช่น ฟรีเคว้นท์ไอเท็มเซตบางตัวอาจจะไม่ใช่ฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุง หรือ ไอเท็มเซตที่ไม่ได้เป็นฟรีเคว้นท์ไอเท็มเซตก่อนหน้านี้อาจจะกลายมาเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุงก็ได้

งานวิจัยทางด้านการเพิ่มขยายการค้นหากฎความสัมพันธ์ส่วนใหญ่มีทั้งการสแกนฐานข้อมูลเดิมและฐานข้อมูลใหม่ ดังเช่นอัลกอริทึมเอฟยูพี ซึ่งจะมีการปรับปรุงค่าสนับสนุนที่ปรากฏในฐานข้อมูลใหม่ให้แก่ฟรีเคว้นท์ไอเท็มเซตตัวเดิมที่หาได้ก่อนหน้า ในกรณีแบบนี้อัลกอริทึมเอฟยูพีไม่จำเป็นต้องนำไอเท็มเซตดังกล่าวไปสแกนในฐานข้อมูลเดิม เนื่องจากทราบค่าสนับสนุนเดิมและสนับสนุนใหม่ จึงสามารถปรับปรุงได้ทันที แต่ในกรณีที่ไอเท็มเซตที่ไม่ได้เป็นฟรีเคว้นท์ไอเท็มเซตก่อนหน้า แต่เป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่ ในกรณีนี้อัลกอริทึมเอฟยูพีจะต้องนำไอเท็ม

ดังกล่าวไปสแกนในฐานข้อมูลเดิมเพื่อหาค่าสนับสนุน จากนั้นจึงดำเนินการปรับปรุงค่าสนับสนุนให้เป็นค่าปัจจุบัน เนื่องจากส่วนใหญ่ข้อมูลชุดใหม่จะมีขนาดเล็กกว่าฐานข้อมูลเดิม ดังนั้นอาจจะพบจำนวนไอเท็มที่เป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่จำนวนมาก ดังนั้น การสแกนฐานข้อมูลเดิมเพื่อหาค่าสนับสนุนให้กับฟรีเคว้นท์ไอเท็มเซตใหม่จำนวนมากจะทำให้ใช้เวลานาน โดยฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลใหม่นี้อาจจะไม่สามารถกลายเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุงได้เลย เป็นต้น

Algorithm2: Incremental Mining Phase

Input: DB, db, $Prob_{pl}$, s, F_k^{DB}, EF_k^{DB}, C_1^{DB}, ρ^{DB}, $\mathcal{Z}$

Output: F_k^{UD}, EF_k^{UD}, C_1^{UD}, ρ^{UD}

```
1     k=1
2         Updating 1-ไอเท็มเซต ( )        // call algorithm 3
3     for k = 2
4         while F_{k-1}^{UD} ≠ ∅
5             Generating Candidate ไอเท็มเซต ( )    // call algorithm 4
6             Updating k-ไอเท็มเซต ( )       // call algorithm 5
7             k++
8         end while loop
9         if Temp_scanDB ≠ ∅
10            Rescanning original database ( )      // call algorithm 6
11        endif
12        clear Temp_scanDB
13        Return F_k^{UD}, EF_k^{UD}, C_1^{UD}, ρ^{UD}
```

ภาพที่ 3.2 การทำงานของอัลกอริทึมการค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นทไอเท็มเซตในฐานข้อมูลใหม่

จากภาพ 3.2 แสดง การทำงานของอัลกอริทึมการค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นทไอเท็มเซตในฐานข้อมูลใหม่ โดยในการทำงานของอัลกอริทึมนี้จำเป็นต้องทราบฟรีเคว้นท์ไอเท็มเซต F_k^{DB} ไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต EF_k^{DB} แคนดิเดต 1-ไอเท็มเซต C_1^{DB} และค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นทไอเท็มเซตของฐานข้อมูลเดิม หรือ ρ^{DB} ของฐานข้อมูลเดิมเสียก่อน ซึ่งหาได้จากอัลกอริทึมที่แสดงดังภาพที่ 3.1

ในการค้นหาฟรีเคว้นท์ไอเท็มเซต และไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลใหม่และปรับปรุงให้เป็นปัจจุบันของงานวิจัยนี้สามารถแบ่งการทำงานได้เป็น 4 ส่วนหลัก ดังนี้

3.2.2.1 การปรับปรุงค่าสนับสนุนของฟรีเคว้นท์ 1- ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ 1-ไอเท็มเซต

ในรอบแรก เมื่อมีข้อมูลชุดใหม่ถูกเพิ่มเข้ามา อัลกอริทึมจะเริ่มปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เป็นฟรีเคว้นท์ไอเท็มเซตและไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต โดยเริ่มสแกนฐานข้อมูลใหม่เพื่อหาค่าสนับสนุนของไอเท็มเซต จากนั้นจึงปรับปรุงค่าสนับสนุนให้แก่ไอเท็มเซตทุกตัวที่เป็นฟรีเคว้นท์ 1- ไอเท็มเซตและไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ 1- ไอเท็มเซต ซึ่งจะถูกกำหนดให้เป็น แคนดิเดต 1 – ไอเท็มเซตของฐานข้อมูลเดิมทุกตัว แทนด้วย C_1^{DB} จากนั้นจะทำการปรับปรุงค่าสนับสนุน ดังบรรทัดที่ 2-6

Algorithm3: Updating 1-itemset

Input: DB, db, $Prob_{pl}$, s, C_1^{DB}, Z

Output: F_k^{UD}, EF_k^{UD}, C_1^{UD}, ρ^{UD}

1 scan db for all X to obtain δ_x^{db}

2 if $X \in C_1^{DB}$

3 $\delta_x^{UD} = \delta_x^{DB} + \delta_x^{db}$

4 else

5 $\delta_x^{UD} = \delta_x^{db}$

6 endif

7 $F_1^{UD} = \{X \mid \delta_x^{UD} \geq s \times |UD|\}$ // $|UD| = |DB| + |db|$

8 for $\{X \mid \delta_x^{UD} < s \times |UD|\}$

9 calculate ptt_x //using eq.9

10 calculate probability of expected ไอเท็มเซต X ($ProbEF_X$) //using eq.7

11 $\rho^{UD} = \min(\delta_x^{DB} | ProbEF_x \geq Prob_{pl})$

12 $EF_1^{UD} = \{X \mid s \times |UD| > \delta_x^{UD} \geq \rho^{UD}\}$

13 $C_1^{UD} = \{X \mid X \in (F_1^{UD} \cup EF_1^{UD} \cup (F_1^{UD} \cup EF_1^{UD})^c)\}$

14 end

15 Return F_1^{UD}, EF_1^{UD}, C_1^{UD}, ρ^{UD}

ภาพที่ 3.3 การปรับปรุงค่าฟรีเคว้นท์ 1- ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ 1-ไอเท็มเซต

จากนั้นไอเท็มเซตทุกตัวที่ได้รับการปรับปรุงค่าสนับสนุนเรียบร้อยแล้ว จะถูกนำมาเปรียบเทียบกับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด s หากไอเท็มเซตใดที่มีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำ ไอเท็มนั้นจะถูกเก็บไว้ในเซตของฟรีเคว้นท์ไอเท็มเซต ส่วนไอเท็มที่ไม่ใช่ฟรีเคว้นท์ไอเท็มเซต จะถูกนำมาคำนวณหาไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต (F_k^{UD}) ดังบรรทัดที่ 7 โดยเริ่มจาก คำนวณค่าความน่าจะเป็นที่ยอมให้เกิดไอเท็มเซต (ptt_X) ซึ่งคำนวณได้จากสมการที่ 9 ใน

บรรทัดที่ 9 จากนั้นจึงนำค่า ptt_X ที่ได้ ไปคำนวณหาค่าความน่าที่ไอเท็มเซตที่พิจารณาจะมีโอกาสเป็นฟรีเคว้นท์ไอเท็มเซต ($\mathrm{Pr}obEF_X$) ในบรรทัดที่ 10

ในบรรทัดที่ 11 ไอเท็มใดที่มีค่า $\mathrm{Pr}obEF_X$ มากกว่าค่า $prob_{PL}$ ถูกนำมาคำนวณหาค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นทไอเท็มเซตของฐานข้อมูลเดิม หรือ ρ^{UD} ซึ่งคำนวณจาก ค่าสนับสนุนของไอเท็มที่มีค่า $\mathrm{Pr}obEF_X$ น้อยที่สุด จากนั้น จึงนำค่า ρ^{UD} และค่า $\mathrm{Pr}obEF_X$ มาเปรียบเทียบกัน โดยไอเท็มเซตใดๆ ที่มีค่าสนุนมากกว่าหรือเท่ากับค่า ρ^{UD} จะถูกนำไปเก็บไว้ในเซตของไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต หรือ EF_k^{UD} ส่วนไอเท็มที่มีค่า $\mathrm{Pr}obEF_X$ น้อยกว่าค่า ρ^{UD} จะถูกตัดทิ้งไป ไม่นำมาคิดคำนวณอีก เมื่ออัลกอริทึมการปรับปรุงฟรีเคว้นท์ 1- ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ 1- ไอเท็มเซต นี้ทำงานเสร็จสิ้น ผลลัพธ์ที่ได้คือ ฟรีเคว้นท์ 1- ไอเท็มเซต F_k^{UD} และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ 1- ไอเท็มเซต EF_k^{UD} และแคนดิเดต 1 – ไอเท็มเซต C_1^{UD} ของฐานข้อมูลปรับปรุง ซึ่งจะถูกนำไปใช้อีกครั้งเมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามา

3.2.2.2 การสร้างแคนดิเดตไอเท็มเซต

การสร้างแคนดิเดตไอเท็มเซตของงานวิจัยนี้ อยู่บนพื้นฐานหลักการสร้างแคนดิเดตไอเท็มเซตของอัลกอริทึมอะพริโอริ แต่สิ่งที่แตกต่างจากอัลกอริทึมอะพริโอริคือไอเท็มเซตที่จะนำมาเข้ากระบวนการจอย (join) เพื่อให้ได้มาซึ่งแคนดิเดตไอเท็มเซต ในกรณีอะพริโอริจะทำการจอยระหว่างฟรีเคว้นท์ k-1 ไอเท็มเซต กับ ฟรีเคว้นท์ k-1 ไอเท็มเซตด้วยกัน แต่ในอัลกอริทึมที่นำเสนอในงานวิจัยนี้จะใช้ไอเท็มในการจอยเหมือนอัลกอริทึมอะพริโอริ แค่เพียงรอบที่ k=2 เท่านั้น เนื่องจากเราทราบค่าสนับสนุนที่แน่นอนของ 1-ไอเท็มเซตทุกตัว แสดงดังบรรทัดที่ 1-3 ในภาพที่ 3.4 จากนั้น เมื่อได้แคนดิเดต 2- ไอเท็มเซตเรียบร้อยแล้ว อัลกอริทึมจะคำนวณหาแคนดิเดต 2-ไอเท็มเซตตัวใหม่ หรือ C_2^{new} ตามบรรทัดที่ 3 เพื่อนำไปใช้ในการหาไอเท็มเซตที่ควรจัดเก็บอยู่ในเซตของ $Temp\_scanDB$ ในอัลกอริทึมถัดไป โดย C_k^{new} จะเป็นไอเท็มเซตที่ไม่ได้เป็นสมาชิกของฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลเดิม

ตั้งแต่รอบที่ k = 3 เป็นต้นไป การสร้างแคนดิเดตไอเท็มเซตของงานวิจัยนี้จะนำไอเท็มเซ็ตที่เป็น ฟรีเคว้นท์ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลเดิม และแคนดิเดต k-1 ไอเท็มเซตของฐานข้อมูลใหม่ มาตรวจสอบค่าสนับสนุนว่า มีค่ามากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดหรือไม่ ดังบรรทัดที่ 6 หากมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ ไอเท็มเซตนั้นๆ จะถูกเก็บไว้ในเซตชั่วคราว แทนด้วยสัญลักษณ์ FX_{k-1} ตามบรรทัดที่ 7 จากนั้นในบรรทัดที่ 8 จึงนำเอาไอเท็มเซตที่เป็นสมาชิกของ FX_{k-1} มาดำเนินการจอย เพื่อสร้างแคนดิเดตไอเท็มเซตขึ้นมา

เมื่อได้แคนดิเดตไอเท็มเซตเรียบร้อยแล้ว อัลกอริทึมจะคำนวณหาแคนดิเดตไอเท็มเซตตัวใหม่ หรือ C_k^{new} ตามบรรทัดที่ 9 เพื่อนำไปใช้ในการหาไอเท็มเซตที่ควรจัดเก็บอยู่ในเซตของ $Temp\_scanDB$ ในอัลกอริทึมถัดไป

Algorithm4: Generating Candidate itemset

Input: C_k^{db}, F_1^{UD}, F_k^{DB}, EF_k^{DB}, s, |db|

Output: C_k, C_k^{new}

1 if k = 2

2 $C_2^{db} = F_1^{UD} * F_1^{UD}$

3 $C_2^{new} = \{X \in C_2^{db} / X \notin (F_2^{DB} \cup EF_2^{DB})\}$

4 else if $k \geq 3$

5 $X = F_{k-1}^{DB} \cup EF_{k-1}^{DB} \cup C_{k-1}^{db}$

6 $\delta_x^{UD} = \delta_x^{DB} + \delta_x^{db}$

7 $FX_{k-1} = \{X | \delta_x^{UD} \geq s^*|db|\}$

8 $C_k^{db} = FX_{k-1} * FX_{k-1}$

9 $C_k^{new} = \{X \in C_k^{db} / X \notin (F_k^{DB} \cup EF_k^{DB})\}$

10 end if

11 Return C_k^{db}, C_k^{new}

ภาพที่ 3.4 การสร้างแคนดิเดตไอเท็มเซต

3.2.2.3 การปรับปรุงค่าสนับสนุนของฟรีเคว้นท์ k- ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ k-ไอเท็มเซต เมื่อ k≥2

ในการปรับปรุงค่าสนับสนุนของฟรีเคว้นท์ k- ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ k-ไอเท็มเซต เมื่อ k≥2 มีกระบวนการทำงานตามภาพที่ 3.5 จะดำเนินการต่อหลังจากที่ได้แคนดิเดตไอเท็มเซต หาได้จากอัลกอริทึมที่ 4 ซึ่งได้อธิบายไปในหัวข้อก่อนหน้า

สำหรับตั้งแต่รอบที่ 2 เป็นต้นไป (k≥2) การปรับปรุงค่าสสนับสนุนของฟรีเคว้นท์ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตจะเริ่มจาก การนำฟรีเคว้นท์ k- ไอเท็มเซต F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ k-ไอเท็มเซต EF_k^{DB} ของฐานข้อมูลเดิม และ แคนดิเดตไอเท็มเซตใหม่ C_k^{new} ที่หาได้จากอัลกอริทึม 4 มาทำการสแกนในฐานข้อมูลข้อมูลใหม่เพื่อหาค่าสนับสนุน δ_x^{db} ตามบรรทัดที่ 1 จากนั้นจะทำการปรับปรุงค่าสนับสนุนให้แก่ไอเท็มเซตเป็นสมาชิกของฟรีเคว้นท์ k- ไอเท็มเซต F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ k-ไอเท็มเซต EF_k^{DB} ของฐานข้อมูลเดิม แต่ไม่เป็นสมาชิกของแคนดิเดตไอเท็มเซตใหม่ C_k^{new} ดังบรรทัดที่ 2-4 จากนั้นจึงนำไอเท็มเซตที่ได้รับการปรับปรุงค่าสนับสนุนเรียบร้อยแล้วมาทดสอบกับค่าสนับสนุนขั้นต่ำเพื่อหาฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง F_k^{UD} ดังบรรทัดที่ 5 ส่วนไอเท็มตัวที่เหลือที่ไม่จัดเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง จะถูกนำไปหาไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง EF_k^{UD} ในบรรทัดที่ 6-12 ซึ่งเป็นขั้นตอนเดียวกับการหาไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของอัลกอริทึมที่ 1

ส่วนไอเท็มเซตที่เป็นสมาชิกของแคนดิเดตใหม่ C_k^{new} จะถูกนำไปคำนวณด้วย ค่าสนับสนุนบวกด้วย ρ^{DB}-1 แล้วนำไปทดสอบกับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด หาก มีค่ามากกว่า หมายความว่าไอเท็มเซตนั้นมีโอกาสที่จะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง ดังนั้นไอเท็มเซตนี้จะถูกเก็บไว้ในตัวแปร *Temp_scanDB* เพื่อนำไปใช้ในการสแกนฐานข้อมูลเก่าอีกครั้งหนึ่ง หลังจากที่ได้ ฟรีเคว้นท์ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตครบทุกตัวแล้ว

Algorithm5: Updating k-itemset

Input: DB, db, $Prob_{pl}$, s, F_k^{DB}, EF_k^{DB}, ρ^{DB}, Z

Output: F_k^{UD}, EF_k^{UD}, C_1^{UD}, ρ^{UD}

1 scan db for all X $\in$ ($F_k^{DB} \cup EF_k^{DB} \cup C_k^{new}$) to obtain δ_x^{db}

2 if X $\in$ ($F_k^{DB} \cup EF_k^{DB}$) and X$\notin C_k^{new}$

3 $\delta_x^{UD} = \delta_x^{DB} + \delta_x^{db}$

4 endif

5 F_k^{UD} ={X | $\delta_x^{UD} \geq$ s×|UD|} // |UD| = |DB|+|db|

6 for {X | $\delta_x^{UD} <$ s×|UD|}

7 calculate ptt_x //using eq.9

8 calculate probability of expected ไอเท็มเซต X ($ProbEF_X$) //using eq.7

9 ρ^{UD} = min(δ_x^{DB}|$ProbEF_X \geq Prob_{pl}$)

10 EF_k^{UD} = {X | s×|UD| > $\delta_x^{UD} \geq \rho^{UD}$ }

12 end for

13 for X $\notin$ ($F_k^{DB} \cup EF_k^{DB}$) and X $\in C_k^{new}$

14 *Temp_scanDB* = {X | (δ_x^{db} +(ρ^{DB} -1)) $\geq$ s×|UD|}

15 end for

16 Return F_k^{UD}, EF_k^{UD}, *Temp_scanDB*

ภาพที่ 3.5 การปรับปรุงค่าสนับสนุนของฟรีเคว้นท์ k- ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ k-ไอเท็มเซต เมื่อ k≥2

3.2.2.3 การสแกนฐานข้อมูลเดิมซ้ำ

จากอัลกอริทึมที่ 2 – 5 ซึ่งเป็นอัลกอริทึมของการค้นหาฟรีเคว้นท์ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง เมื่ออัลกอริทึมที่ 5 ทำงานเสร็จสิ้นจะได้ ฟรีเคว้นท์ไอเท็มเซต F_k^{UD} และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต EF_k^{UD} ของฐานข้อมูลปรับปรุง ที่ปรับปรุงเรียบร้อยในระดับหนึ่ง ยังคงเหลือการหาฟรีเคว้นท์ไอเท็มเซ็ตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตครั้งสุดท้ายจากไอเท็มเซตที่ถูกเก็บไว้ในตัวแปร

Temp_scanDB ไอเท็มเซตดังกล่าว เป็นไอเท็มเซตชุดใหม่ที่เกิดขึ้นในฐานข้อมูลใหม่ ที่คาดว่ามีโอกาสจะเป็นฟรีเคว้นท์ไอเท็มเซตหรือไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต

ในอัลกอริทึมที่ 6 แสดงดังภาพที่ 3.6 จะนำเสนอขั้นตอนการนำไอเท็มเซตที่อยู่ในตัวแปร *Temp_scanDB* มาสแกนฐานข้อมูลเดิม เพื่อปรับปรุงค่าสนับสนุน ซึ่งในการสแกนฐานข้อมูลเดิมนี้จะกระทำเพียงครั้งเดียว หลังจากอัลกอริทึมที่ 2-5 ทำงานครบ k รอบเป็นที่เรียบร้อยแล้ว ไอเท็มเซตที่เป็นสมาชิกของ *Temp_scanDB* จะถูกนำไปสแกนในฐานข้อมูลเพื่อหาค่าสนับสนุนตามบรรทัดที่ 1 จากนั้นค่าสนับสนุนจะถูกปรับปรุง ตามบรรทัดที่ 2 เมื่อได้ค่าสนับสนุนที่ปรับปรุงแล้ว ไอเท็มทุกตัวจะถูกนำไปทดสอบกับค่าสนับสนุนขั้นต่ำเพื่อหาฟรีเคว้นท์ไอเท็มเซตใหม่ของฐานข้อมูลปรับปรุง ตามบรรทัดที่ 3 และ ไอเท็มตัวที่เหลือจะถูกนำไปทดสอบกับค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลเดิม หรือ ρ^{UD} เพื่อหาไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตของฐานข้อมูลปรับปรุง ตามบรรทัดที่ 4

Algorithm6: Rescanning original database

Input: DB, db, s, F_k^{DB}, EF_k^{DB}, C_1^{DB}, ρ^{UD}

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

1 scan DB for all X ∈ *Temp_scanDB* to obtain δ_x^{DB}

2 $\delta_x^{UD} = \delta_x^{DB} + \delta_x^{db}$

3 $F_k^{new} = \{X \mid X \in Temp\_scanDB \text{ and } \delta_x^{UD} \geq s^* |UD|\}$

4 $EF_k^{new} = \{X \mid X \in Temp\_scanDB \text{ and } s^* |UD| > \delta_x^{UD} \geq \rho^{UD}\}$

5 $F_k^{UD} = F_k^{UD} \cup F_k^{new}$

6 $EF_k^{new} = EF_k^{UD} \cup EF_k^{new}$

ภาพที่ 3.6 การสแกนฐานข้อมูลเดิมซ้ำ

ในบรรทัดที่ 5 และ 6 จะเป็นการนำเอาฟรีเคว้นท์ไอเท็มเซตใหม่และไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตใหม่ของฐานข้อมูลปรับปรุงไปผนวกเข้ากับตัวแปรฟรีเคว้นท์ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตเดิม ตามลำดับ ซึ่งตัวแปรเหล่านี้จะถูกนำไปใช้อีกครั้งเมื่อมีการเพิ่มข้อมูลชุดใหม่ชุดถัดไปเข้ามา

บทที่ 4
ผลการทดลอง

เพื่อแสดงให้เห็นถึงประสิทธิภาพการทำงานของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ซึ่งเป็นอัลกอริทึมที่ปรับปรุงประสิทธิภาพการทำงานของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น ที่ช่วยในการค้นหากฎความสัมพันธ์แบบเพิ่มขยาย ในบทนี้จะกล่าวถึงวัตถุประสงค์ของการทดลอง วิธีการทดลอง และผลการทดลอง ของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic

4.1 วัตถุประสงค์ของการทดลอง

เพื่อแสดงให้เห็นถึงประสิทธิภาพการทำงานของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic สำหรับการเพิ่มขยายการค้นหากฎความสัมพันธ์ โดยเป็นการปรับปรุงประสิทธิภาพอัลกอริทึมอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น ในหัวข้อนี้จะกล่าวถึงวัตถุประสงค์ของการทดลอง ซึ่งประกอบด้วย 2 วัตถุประสงค์หลัก ดังนี้

1. **เพื่อทดสอบความถูกต้องของผลลัพธ์ที่ได้จากการเพิ่มฐานข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม** อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic แม้ว่าจะเป็นอัลกอริทึมที่พัฒนามาเพื่อปรับปรุงประสิทธิภาพของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น อย่างไรก็ดี การทำงานของทั้งอัลกอริทึมอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic และ อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น ล้วนมีฐานการทำงานมาจากอัลกอริทึม Apriori ดังนั้นผู้วิจัยจะออกแบบการทดลองเพื่อทดสอบความถูกต้องของผลลัพธ์โดยเปรียบเทียบกับอัลกอริทึม Apriori เป็นหลัก
2. **เพื่อทดสอบประสิทธิภาพในการทำงานของอัลกอริทึมในการเพิ่มฐานข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม** การทดสอบประสิทธิภาพของอัลกอริทึมในงานวิจัยนี้ จะเป็นการทดสอบเพื่อวัดประสิทธิภาพการเพิ่มขยายการค้นหากฎความสัมพันธ์โดยวัดจากเวลาที่ใช้ในการประมวลผล (Execution Time) โดยเปรียบเทียบเวลาที่ใช้ในการประมวลผลระหว่างอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessmistic อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น และอัลกอริทึม Apriori

4.2 วิธีการทดลอง

การทดลองอัลกอริทึมเพื่อค้นหากฎความสัมพันธ์แบบเพิ่มขยาย เมื่อมีการเพิ่มฐานข้อมูลหรือชุดข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม ซึ่งอาจทำให้ฟรีเคว้นท์ไอเท็มเซ็ตที่ได้จากการค้นหาจากฐานข้อมูลเดิมมีการเปลี่ยนแปลงไป ในการทดลองสำหรับงานวิจัยนี้ จะเป็นการทดลองโดยการเพิ่มโดยเพิ่มเข้าไปในฐานข้อมูลเดิมจำนวน 2 ฐานข้อมูล ที่จำนวนทรานแซคชั่น 3,000 และ 5,000 ทรานแซคชั่น ตามลำดับ ด้วยค่าสนับสนุนขั้นต่ำ (minimum support) ในระดับที่ไม่แตกต่างกัน คือ minsup = 0.005% ค่า ProbPL = 0.1 และค่า confidence interval 15%

สำหรับชุดข้อมูลที่นำมาทดลอง เป็นชุดข้อมูลสังเคราะห์ (Synthesis Dataset) ซึ่งเป็นชุดข้อมูลที่นำเสนอโดย Agrawal และคณะ [1] ซึ่งได้เสนอวิธีการสร้างชุดข้อมูลสังเคราะห์เพื่อใช้ในการประเมินประสิทธิภาพของอัลกอริทึม โดยอาศัยหลักการทางสถิติมาใช้ในการสร้างชุดข้อมูล สำหรับการทดลองในงานวิจัยนี้ ชุดข้อมูลที่ใช้คือชุดข้อมูล I10T4D100K สำหรับฐานข้อมูลเดิม 10,000 ทรานแซคชั่น และฐานข้อมูลใหม่ 3,000 ทรานแซคชัน และ 5,000 ทรานแซคชัน

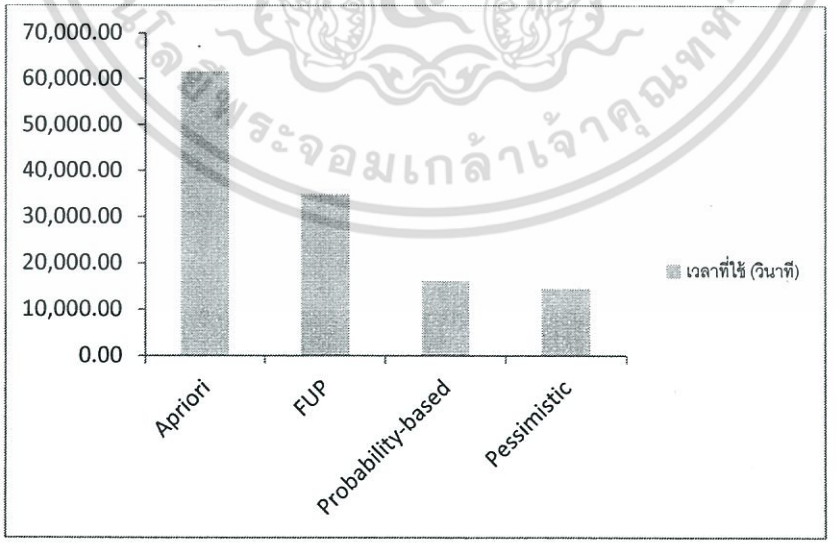
4.3 ผลการทดลอง

เมื่อนำข้อมูลทั้ง 2 ชุดดังกล่าวข้างต้นมาทำการทดลองด้วยโปรแกรม MATLAB 7.6 ผลการทดลองเป็นดังนี้

ผลการทดลองการเพิ่มข้อมูลจำนวน 3,000 ทรานแซคชัน ด้วยการทดสอบกับค่าสนับสนุนต่ำสุดจำนวน 0.005% และทดสอบประสิทธิภาพของอัลกอริทึมโดยเปรียบเทียบกับอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น (Probability-Based) อัลกอริทึม FUP และอัลกอริทึม Apriori ผลการทดลองแสดงตามตารางที่ 4.1 และภาพที่ 4.1

ตารางที่ 4.1 ผลการเปรียบเทียบเวลาที่ใช้ในการประมวลผลเมื่อมีการเพิ่มข้อมูล 3,000 ทรานแซคชัน

อัลกอริทึม	Apriori	FUP	Probability-based	Pessimistic
เวลาที่ใช้ (วินาที)	61,452.0120	34,951.3102	16,252.6708	14,656.8360
จำนวนฟรีเคว้นทไอเท็มเซต	1,105	1,105	1,105	1,105
จำนวนฟรีเคว้นท์ที่คาดว่าจะเป็นไอเท็มเซต	-	-	40	115
ขนาดสูงสุดของฟรีเคว้นทไอเท็มเซต (k)	F_5	F_5	F_5	F_5
จำนวน TempscanDB	-	-	95	31

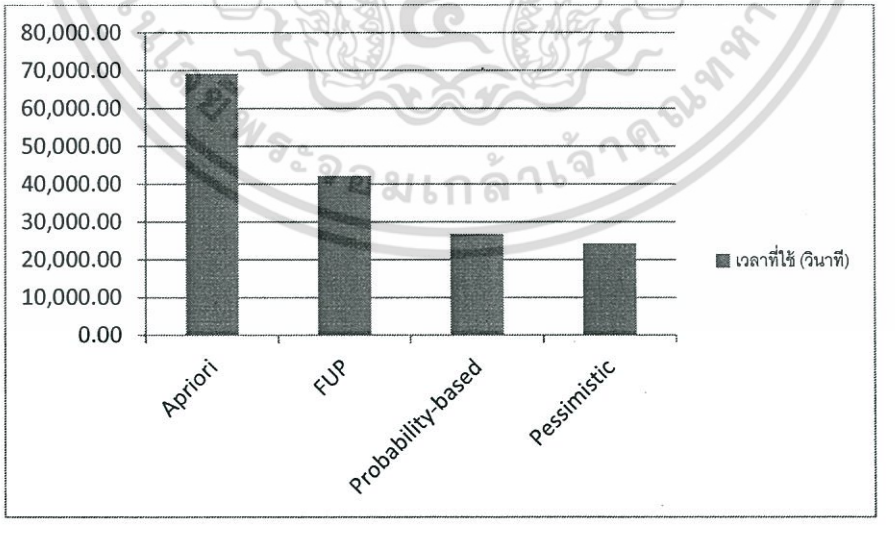


ภาพที่ 4.1 แผนภูมิแท่งเปรียบเทียบเวลาที่ใช้ในการประมวลผลเมื่อมีข้อมูลเพิ่ม 3,000 ทรานแซคชัน

จากตารางที่ 4.1 และภาพที่ 4.1 แสดงผลการเปรียบเทียบเวลาที่ใช้ในการประมวลผลของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น (Probability-Based) อัลกอริทึม FUP และอัลกอริทึม Apriori จะเห็นได้ว่า เมื่อมีการเพิ่มข้อมูลจำนวน 3,000 ทรานแซคชั่น เข้าไปในฐานข้อมูลเดิมจำนวน 10,000 ทรานแซคชั่น ด้วยค่าสนับสนุนขั้นต่ำ 0.005% เมื่อพิจารณาถึงความถูกต้องของความถูกต้องในการประมวลผลโดยเปรียบเทียบกับอัลกอริทึม Apriori ผลการทดสอบพบว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessmistic ให้ผลลัพธ์ที่ถูกต้องคือมีจำนวน ฟรีเคว้นท์ไอเท็มเซ็ตเท่ากับจำนวนฟรีเคว้นท์ไอเท็มเซตที่ประมวลผลได้จากอัลกอริทึม Apriori FUP และ Probability-Based นอกจากนี้ เมื่อพิจารณาถึงเวลาที่ใช้ในการประมวลผล จะพบว่า ใช้ระยะเวลาในการประมวลผลน้อยกว่าอัลกอริทึม Apriori และ FUP อย่างเห็นได้ชัด และใช้เวลาในการประมวลผลน้อยกว่าอัลกอริทึม Probability-Based ซึ่งถือว่า อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic สามารถปรับปรุงอัลกอริทึม Probability-based ให้สามารถใช้เวลาในการประมวลน้อยลง

ตารางที่ 4.2 ผลการเปรียบเทียบเวลาที่ใช้ในการประมวลผลเมื่อมีการเพิ่มข้อมูล 5,000 ทรานแซคชัน

อัลกอริทึม	Apriori	FUP	Probability-based	Pessimistic
เวลาที่ใช้ (วินาที)	69,215.6714	42,219.5093	26,826.7789	24,316.6723
จำนวนฟรีเคว้นทไอเท็มเซต	1,097	1,097	1,097	1,097
จำนวนฟรีเคว้นท์ที่คาดว่าจะเป็นไอเท็มเซต	-	-	50	137
ขนาดสูงสุดของฟรีเคว้นทไอเท็มเซต (k)	F_4	F_4	F_4	F_4
จำนวน TempscanDB	-	-	85	39



ภาพที่ 4.2 แผนภูมิแท่งเปรียบเทียบเวลาที่ใช้ในการประมวลผลเมื่อมีข้อมูลเพิ่ม 5,000 ทรานแซคชัน

จากตารางที่ 4.2 และภาพที่ 4.2 แสดงผลการเปรียบเทียบเวลาที่ใช้ในการประมวลผลของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น (Probability-Based) อัลกอริทึม FUP และอัลกอริทึม Apriori จะเห็นได้ว่า เมื่อมีการเพิ่มข้อมูลจำนวน 5,000 ทรานแซคชั่น เข้าไปในฐานข้อมูลเดิมจำนวน 10,000 ทรานแซคชั่น ด้วยค่าสนับสนุนขั้นต่ำ 0.005% เมื่อพิจารณาถึงความถูกต้องของความถูกต้องในการประมวลผลโดยเปรียบเทียบกับอัลกอริทึม Apriori ผลการทดสอบพบว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessmistic ให้ผลลัพธ์ที่ถูกต้อง คือมีจำนวน ฟรีเคว้นท์ไอเท็มเซ็ตเท่ากับจำนวนฟรีเคว้นท์ไอเท็มเซตที่ประมวลผลได้จากอัลกอริทึม Apriori FUP และ Probability-Based นอกจากนี้ เมื่อพิจารณาถึงเวลาที่ใช้ในการประมวลผล จะพบว่า ใช้ระยะเวลาในการประมวลผลน้อยกว่าอัลกอริทึม Apriori และ FUP อย่างเห็นได้ชัด และใช้เวลาในการประมวลผลน้อยกว่าอัลกอริทึม Probability-Based ซึ่งถือว่า อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic สามารถปรับปรุงอัลกอริทึม Probability-based ให้สามารถใช้เวลาในการประมวลน้อยลง

4.4 สรุปผลการทดลอง

จากการทดลองเพิ่มข้อมูลจำนวน 2 ชุด คือ 3,000 ทรานแซคชัน และ 5,000 ทรานแซคชัน เข้าไปในฐานข้อมูลเดิมจำนวน 10,000 ทรานแซคชัน ด้วยค่าสนับสนุนขั้นต่ำ minsup = 0.005% ซึ่งข้อมูลที่ใช้ทดสอบเป็นข้อมูลที่ได้จากการสังเคราะห์ข้อมูล I4T10D100K ในการทดลองจะแบ่งการวัดผลการทดลองออกเป็น 2 ประเด็น คือประเด็นของความถูกต้องในการประมวลผล และการวัดประสิทธิภาพด้านเวลาที่ใช้ในการประมวลผล สามารถสรุปผลการทดลองทั้ง 2 ประเด็นได้ดังนี้

1. ด้านความถูกต้อง ในการทดสอบความถูกต้องของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ผู้วิจัยได้ออกแบบการทดสอบความถูกต้องโดยการเปรียบเทียบกับผลลัพธ์ที่ได้จากการประมวลผลอัลกอริทึม Apriori ซึ่งผลการทดสอบ พบว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ให้ผลลัพธ์ในการประมวลผลที่ถูกต้อง นั่นคือ มีจำนวนฟรีเคว้นท์ไอเท็มเซ็ตที่ได้จากการประมาลผลและขนาดสูงสุดของ k-itemset เท่ากับอัลกอริทึม Apriori

2. ด้านประสิทธิภาพด้านเวลาที่ใช้ในการประมวลผล ในการทดสอบ ผู้วิจัยได้ทดสอบประสิทธิภาพด้านเวลาที่ใช้ในการประมวลผล โดยการเปรียบเทียบกับ 3 อัลกอริทึมอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น (Probability-Based) อัลกอริทึม FUP และอัลกอริทึม Apriori ด้วยค่าสนับสนุนต่ำสุด 0.005% ผลการทดลอง พบว่า อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ใช้เวลาในการประมวลผลน้อยกว่าอัลกอริทึม Apriori และ FUP อย่างเห็นได้ชัด และใช้เวลาประมวลผลน้อยกว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้ความน่าจะเป็น ซึ่งเมื่อพิจารณาจากจำนวน TempscanDB ที่เก็บไว้เพื่อสแกนฐานข้อมูลเดิม จะเห็นได้ว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ประมวลผลเร็วกว่า เนื่องจากจำนวนไอเท็มที่จะถูกนำไปสแกนในฐานข้อมูลเดิมมีจำนวนน้อยกว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์แบบเพิ่มขยายโดยใช้หลักความน่าจะเป็น

บทที่ 5
สรุปและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

การค้นหากฎความสัมพันธ์ เป็นเทคนิคที่สำคัญของกระบวนการทำเหมืองข้อมูล (Data Mining) เพื่อค้นหาความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูล และสกัดรูปแบบข้อมูลที่น่าสนใจให้ออกมาอยู่ในรูปแบของกฎความสัมพันธ์ if X then Y โดยมีหลักการทำงานหลัก 2 ขั้นตอนได้แก่ 1) การค้นหาไอเท็มเซ็ตที่เรียกว่าฟรีเคว้นท์ไอเท็มเซ็ต ซึ่งเป็นไอเท็มเซตที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ (minimum support) ที่ผู้ใช้กำหนด และ 2) การนำฟรีเคว้นท์ไอเท็มเซ็ตที่หาได้จากข้อแรกมาสร้างกฎความสัมพันธ์ ซึ่งกฎความสัมพันธ์ที่น่าสนใจ จะเป็นกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นมากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำ (minimum confidence) ที่ผู้ใช้กำหนด

โดยทั่วไปแล้ว อัลกอริทึมที่ได้รับการยอมรับและเป็นที่นิยมในการนำมาค้นหากฎความสัมพันธ์ คืออัลกอริทึม Apriori [3] ที่ถูกเสนอโดย Agrawal อย่างไรก็ดี เมื่อมีการเพิ่มชุดข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม การค้นหากฎความสัมพันธ์ด้วยวิธีการของ Apriori จะต้องทำประมวลผลข้อมูลทั้งหมดที่อยู่ฐานข้อมูล นั่นคือ การหาแคนดิเดตไอเท็มเซตใหม่ และสแกนหาสนับสนุนของแคนดิเดตไอเท็มเซ็ตใหม่ในทุกๆ รอบ ซึ่งทำให้เวลาที่ใช้ในการประมวลผลถูกใช้มากเกินความจำเป็นและไม่เกิดประสิทธิภาพในการทำงาน

ด้วยข้อด้อยดังกล่าวของ Apriori จึงได้มีผู้นำเสนอการปรับปรุงกฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิม Cheung และคณะ [4] ได้ เสนออัลกอริทึม FUP (Fast UPdate algorithm) ซึ่งเป็นอัลกอริทึมสำหรับการค้นหากฎความสัมพันธ์แบบเพิ่มขยาย เมื่อมีการเพิ่มข้อมูลใหม่เข้ามา โดยอาศัยองค์ความรู้เดิมจากการไมนิ่งในฐานข้อมูลเดิมมาช่วยเพิ่มประสิทธิภาพในการค้นหากฎความสัมพันธ์ นั่นคือ FUP จะใช้ฟรีเคว้นไอเท็มเซ็ตที่ได้จากการประมวลผลในฐานข้อมูลเดิม มาช่วยลดจำนวนแคนดิเดตไอเท็มเซ็ตที่จะต้องถูกนำไปสแกนในฐานข้อมูลเดิม ด้วยวิธีการประมวลผลดังกล่าวจึงทำให้ FUP ใช้เวลาในการประมวลผลน้อยกว่า Apriori

อย่างไรก็ตาม ในแต่ละรอบ k เมื่อมีการค้นพบแคนดิเดตไอเท็มเซตที่ไม่ได้เป็นสมาชิกของฟรีเคว้นท์ไอเท็มเซ็ตของฐานข้อมูลเดิม แคนดิเดตไอเท็มเซ็ตนั้นๆ จะถูกนำไปสแกนในฐานข้อมูลเดิมทุกรอบที่ k นั่นแปลว่า หากขนาดของฟรีเคว้นไอเท็มเซตในฐานข้อมูลปรับปรุงมีค่าเท่ากับ 5 (k=5) ย่อมหมายความว่า FUP จะต้องนำแคนดิเดตไอเท็มเซ็ตไปสแกนในฐานข้อมูลเดิมรวมจำนวนทั้งสิ้น 5 รอบ เช่นเดียวกับ Apriori ต่างกันตรงที่ จำนวนแคนดิเดตไอเท็มเซ็ตที่ถูกนำไปสแกนของ FUP จะมีจำนวนน้อยกว่า Apriori เท่านั้น

เพื่อลดจำนวนการสแกนฐานข้อมูลเดิมให้เหลือจำนวนรอบการสแกนที่น้อยที่สุดในงานวิจัยการค้นหากฎความสัมพันธ์แบบเพิ่มขยายโดยอาศัยหลักความน่าจะเป็น [3] ได้นำเสนอเทคนิคการประมาณค่าความน่าจะเป็นของไอเท็มเซ็ตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตสำหรับเก็บไว้เพื่อนำไปสแกนฐานข้อมูลเดิมเพียงครั้งเดียวในรอบสุดท้าย ซึ่งอัลกอริทึมดังกล่าวทำงานได้อย่างมีประสิทธิภาพให้ผลทางด้านเวลาที่รวดเร็วกว่า FUP และ Apriori อย่างไรก็ดี โดยพื้นฐานการหาความน่าจะเป็นของอัลกอริทึมดังกล่าวใช้หลักการหาความน่าจะเป็นเบอร์นูลลี ซึ่งจะมีปัญหาในการหาความน่าจะเป็นในกรณีที่ต้องคำนวณแฟคทอเรียลของจำนวนจริงที่มีค่ามาก งานวิจัยนี้จึงใช้หลักการประมาณค่าความ

น่าจะเป็นด้วยการเทียบค่า ทำให้ค่าความน่าจะเป็นที่ได้ไม่ตรงกับความเป็นจริง ส่งผลให้การทำนายไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตมีโอกาสคลาดเคลื่อนได้

ผู้วิจัยจึงได้ศึกษาค้นคว้าเพื่อปรับปรุงอัลกอริทึม การค้นหากฎความสัมพันธ์แบบเพิ่มขยายโดยอาศัยหลักการความน่าจะเป็น ให้สามารถทำนายไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซตได้แม่นยำมากขึ้น โดยอาศัยหลักการประมาณค่าด้วยการแจกแจงปกติ และใช้แนวคิดด้าน Pessimistic และ Confidence Interval มาใช้ในการตัดสินใจการเลือกเก็บไอเท็มเซตที่คาดว่าจะเป็นฟรีเคว้นท์ไอเท็มเซต

ในการทดลองผล ผู้วิจัยทดลองผลกับข้อมูลสังเคราะห์คือ ชุดข้อมูล I10T4D50K สำหรับฐานข้อมูลเดิม 10,000 ทรานแซคชั่น และฐานข้อมูลใหม่ 2 ชุด คือ 3,000 ทรานแซคชันและ 5,000 ทรานแซคชัน ด้วยค่าสนับสนุนขั้นต่ำ (minimum support) ในระดับที่ไม่แตกต่างกัน คือ minsup = 0.005% ค่า ProbPL = 0.1 และค่า confidence interval 15%

ผลการทดลองแบ่งออกเป็น 2 ประเด็น คือประเด็นของความถูกต้องในการประมวลผลและการวัดประสิทธิภาพด้านเวลาที่ใช้ในการประมวลผล สามารถสรุปผลการทดลองทั้ง 2 ประเด็นได้ดังนี้

1. ด้านความถูกต้อง ในการทดสอบความถูกต้องของอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ผู้วิจัยได้ออกแบบการทดสอบความถูกต้องโดยการเปรียบเทียบกับผลลัพธ์ที่ได้จากการประมวลผลอัลกอริทึม Apriori ซึ่งผลการทดสอบ พบว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ให้ผลลัพธ์ในการประมวลผลที่ถูกต้อง นั่นคือ มีจำนวนฟรีเคว้นท์ไอเท็มเซ็ตที่ได้จากการประมาลผลและขนาดสูงสุดของ k-itemset เท่ากับอัลกอริทึม Apriori

2. ด้านประสิทธิภาพด้านเวลาที่ใช้ในการประมวลผล ในการทดสอบผู้วิจัยได้ทดสอบประสิทธิภาพด้านเวลาที่ใช้ในการประมวลผล โดยการเปรียบเทียบกับ 3 อัลกอริทึมอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักความน่าจะเป็น (Probability-Based) อัลกอริทึม FUP และอัลกอริทึม Apriori ด้วยค่าสนับสนุนต่ำสุด 0.005% ผลการทดลอง พบว่า อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ใช้เวลาในการประมวลผลน้อยกว่าอัลกอริทึม Apriori และ FUP อย่างเห็นได้ชัด และใช้เวลาประมวลผลน้อยกว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์โดยใช้ความน่าจะเป็น ซึ่งเมื่อพิจารณาจากจำนวน TempscanDB ที่เก็บไว้เพื่อสแกนฐานข้อมูลเดิม จะเห็นได้ว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบ Pessimistic ประมวลผลเร็วกว่า เนื่องจากจำนวนไอเท็มที่จะถูกนำไปสแกนในฐานข้อมูลเดิมมีจำนวนน้อยกว่าอัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์แบบเพิ่มขยายโดยใช้หลักความน่าจะเป็น

5.2 ข้อเสนอแนะ

ในงานวิจัยฉบับนี้ ในการคำนวณค่าความน่าจะเป็นเพื่อหาว่าไอเท็มเซตใดสามารถจะเป็นฟรีเคว้นท์ไอเท็มเซตในฐานข้อมูลปรับปรุงได้ จำเป็นจะต้องทราบจำนวนฐานข้อมูลใหม่ที่จะถูกเพิ่มเข้ามาอย่างแน่นอน ซึ่งในความเป็นจริงเราไม่สามารถทราบได้ว่าจะมีฐานข้อมูลใหม่ถูกเพิ่มเข้ามาจำนวนกี่ทรานแซคชัน ดังนั้นในงานวิจัยครั้งต่อไป จึงควรจะค้นหาวิธีการคำนวณความน่าจะเป็นโดยไม่จำเป็นต้องทราบจำนวนทรานแซคชันที่จะถูกเพิ่มเข้ามา

บรรณานุกรม

[1] Agrawal, R. and Srikant, R., "Fast algorithms for mining association rules." **Proceedings of 20 th VLDB Conference Santiago.** Chile, 1994. pp.487-499

[2] Cheung, D.W., Han, J., Ng, V.T. and Wong, C.Y., "Maintenance of Discovered Association Rules in Large Database: An incremental updating technique" **In 12 th IEEE International Conference on Data Engineering, 1996.**

[3] Amornchewin, R., and Kreesuradej, W., "Mining Dynamic Databases using Probability-Based Incremental Association Rule Discovery Algorithm." **Journal of Universal Computer Science,** pp. 2409 – 2428 .vol 15, no.12, 2009.

[4] Agrawal, R., Imielinski, T., and Swami, A. "Mining association rules between sets of items in large databases." **Proceeding of the 1993 ACM SIGMOD Conference on Washington DC**, USA, May 1993.

[5] S M Tsai Paulry, Lee Chin-Chong, L P Chen Arbee, "An Efficient Approach for Incremental Association Rule Mining," **Proceedings of the third pacific-Asia Conference on methodologies for Knowledge Discovery and Data Mining,** Lecture notes in Computer Science, Vol. 1574 archive, 1999.

[2] W.-G. Teng and M.-S. Chen. "Incremental Mining on Association Rules." **Foundations and Advances in Data Mining,** pp. 125-162, edited by W. Chu and T.-Y. Lin, Springer, 2005.

[6] Cheung, D.W., Lee, S.D. and Kao,, B., "A general Incremental Technique for Maintaining Discovered Association Rules." **In Proceedings of the fifth International Conference on Database Systems for Advanced Applications,** Melbourne, Australia, April 1997.

[7] Thomas, S., Bodagala, S., Alsabti, K., and Ranka, S., "An efficient algorithm for th incremental updation of association rules in large databases." **In Proceedings of the 3 rd International Conference on Knowledge Discovery and Data Mining.**" New Port Beach, California, 1997.

[8] Amornchewin, R., and Kreesuradej, W., "Incremental Association Rules Mining Using Promising Frequent ไอเท็มเซต Algorithm." **In Proceeding 6th International Conference on Information, Communications and Signal Processing,** Singapore, 2007

[9] Papoulis, Pillai, "**Probability, Random Variables, and Stochastic Processes**", 4th Edition.

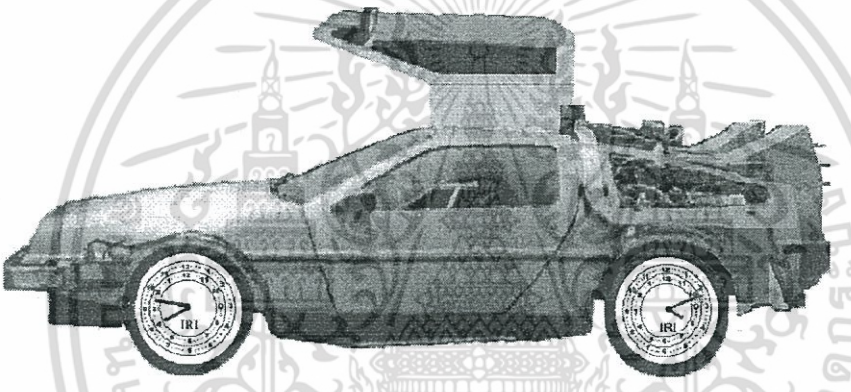
[10] John E. Freund, Benjamin M. Perles, "Modern Elementary Statistics," 12th Edition, Pearson Ed. New Jersey, 2007

ภาคผนวก

Proceedings of the 2013 IEEE 14th International Conference on Information Reuse and Integration

IEEE IRI 2013

Sponsored by:
IEEE Systems, Man and Cybernetics Society (IEEE SMC)
IEEE Computer Society
Society for Information Reuse and Integration (SIRI)



August 14 -16, 2013, San Francisco, California, USA

Editors: Chengcui Zhang, James Joshi, Elisa Bertino, Bhavani Thuraisingham

A Publication of
IEEE Systems, Man, and Cybernetics Society (SMC)
445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331, USA
IEEE Catalog Number: CFP13IRI-ART
ISBN: 978-1-4799-1050-2

Session C11: Reuse in Software Engineering I

Session C12: Machine Learning

Session C13: Workshop on Formal Methods Integration (FMi)

Probability-Based Incremental Association Rule Discovery Using the Normal Approximation

Araya Ariya, Worapoj Kreesuradej
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang, Thailand
araya_aa@hotmail.com, worapoj@it.kmitl.ac.th

Abstract

An incremental association rules mining is one of an association rule mining research topics which finds the relation between set of item in dynamic databases. As data grows up rapidly, the co-occurrence itemset which discovered in the previous mining may be changed and the association rule will be change consequently. Incremental association rule mining research attempts to maintain that rules. Probability-based algorithm, one of an incremental algorithm, applied the principle of Bernoulli trial to predict expected frequent itemsets for reducing collected border itemsets and a number of times to rescan the original database. However, the numerical problem will occur when the algorithm deals with a large database. To manipulate with this problem, the improved probability-based incremental association rule discovery using normal approximation to estimate the probability of occurrence of expected frequent itemset is introduced in this paper. In addition, the confidence interval is applied to ensure that the collecting of expected frequent itemsets is properly kept.

Keywords: Data Mining, Incremental Association Rule Discovery, Normal Approximation

1. Introduction

Association rule mining is well known as one of a core task of data mining in knowledge discovery in databases process. It was first introduced by Agrawal et al. [1] in a pilot study of the market basket analysis which found simultaneous bought itemset. It shows the perspective of information hidden in large databases in the form of if antecedent then consequent ($X \rightarrow Y$) such as customers who buy beers they must also buy diapers. The process of basic rule discovery operation is defined as follows. Firstly, counting co-occurrence itemsets in entire databases, secondly, itemsets which is greater than or equal to minimum support threshold is collected to create an association rule. Finally, rule which is greater than or equal to minimum confidence threshold will be a valid extracted association rule. From that research, the first market basket analysis, the problem statement of an association rule mining is commonly defined as follows.

Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of literal items. DB is a database which contains transactions. Each transaction T is a set of items where $T \subseteq I$. Given X is an item and $X \subseteq I$. Each transaction contains X if and only if $X \subseteq T$. Let X and Y are items where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y \neq \emptyset$. Set of items, itemsets, is called a frequent itemset or large itemset if and only if its support count, calculated from a number of transactions in DB that contain $X \cup Y$, is greater than or equal to support threshold s%. Each frequent itemset can be made the association rule if and only if it is satisfied by confidence threshold c% which calculates from a number of transactions in DB that contain X and also contain Y. Both s% and c% are liberally defined by users.

The number of researchers has studied extensively about association rule mining research area in many issues. Incremental association rule discovery is one of those issues which maintain rules when new transactions are appended to an original database. In fact, data has grown rapidly; thus, when a new set of transactions, called increment database, are inserted into the original database, some rules from the previous mining may be invalid. This problem has been motivating many researchers to propose several rules maintaining algorithms.

A basic and simple method for solving this problem is to rescan entire databases with Apriori algorithm [2] to get new itemsets. However, this method is time-consuming and inefficiency. By reducing a number of times to scan databases, several algorithms are proposed such as Sliding Windows Filtering (SWF) [3], Negative Border (NBd) [4], probability-based incremental association rule discovery [5] and so on. This research also proposes in this direction.

For the probability-based incremental association rule discovery algorithm, it needs only one time to scan the whole original database and works by using the principles

IEEE IRI 2013, August 14-16, 2013, San Francisco, California, USA
978-1-4799-1050-2/13/$31.00 ©2013 IEEE

of Bernoulli trials to predict the expected frequent itemsets, i.e., the infrequent itemset which can possibly be a frequent itemset. However, numerical difficulty in computing probabilities occurs when the algorithm deals with a large database. To manipulate with this problem, the improved probability-based incremental association rule discovery using normal approximation to estimate the probability of occurrence of expected frequent itemset is introduced in this paper. In addition, a statistical confidence interval is applied to ensure that the collecting of expected frequent itemsets is properly kept.

The other parts of this paper are organized as follows. The literatures of an association rule mining and incremental association rule mining are reviewed in section 2. The problem statement of an incremental association rule mining and the probability-based incremental association rule discovery algorithm are detailed in section 3. Probability-based incremental association rule discovery using the normal approximation and the experiment are presented in section 4 and 5 respectively. The summary and conclusion is described briefly in section 6.

2. Related Work

In 1993, association rule discovery was first proposed by Agrawal et al. [1] in a pilot study in market basket analysis which found the relationship between the buying items in a retail transaction database. Next year, Apriori [2], the most popular algorithm of association rule mining, was issued. Apriori is normally divided into 2 major steps: finding frequent itemsets (sometimes called large itemsets) and generating rules. After Apriori was revealed, there are many researchers propose algorithms in this field.

In the real world, databases are dynamic. The database size has been enlarging because the new set of transactions is continuously inserted into the original database. When the new increment database, the new set of transactions, is inserted in to the original database, the old existing rule may be invalid. It is easy to rerun Apriori in whole database (consists of original database and increment database) to get the updated rules if and only if both databases and a number of item are small. In fact, the database is big and possibly bigger in over time, rerunning Apriori is not the suitable way to do because too much time is consumed. Thus, the incremental association rule discovery, one of the association rule discovery issue, is studied extensively to maintain association rules in dynamic databases.

Fast UPdate algorithm (FUP) [6] was proposed to maintain association rules in dynamic databases. It works by using frequent itemsets from previous mining in the original database compares with frequent itemsets in the increment database. For each iteration, a frequent itemset in the increment database which is not a frequent itemset in the original database will be rescanned in the original database and updates its support count. From the FUP experiment result, even though it can save the computational time but it still needs to rescan an original database k times when new frequent k-itemsets are found. This is the disadvantage of FUP.

Sliding Windows Filtering (SWF) [3] was proposed to reduce a number of rescanning times of an original database by dividing both original database and increment database into several partitions, and processing from the first partition to the last partition. There are 2 major procedures: preprocessing procedure and incremental procedure. Two new ideas are proposed in SWF: all 1-tiemsets are assumed to frequent itemsets and candidate $k \geq 3$ itemsets are obtained from $C_{k-1} * C_{k-1}$, these ideas can decrease a number of candidate itemsets. This algorithm is a good work for both deleted and inserted database. In addition, SWF requires only one time to rescan an original database.

Negative Border algorithm (NBd) [4] was proposed to reduce the number of rescanning times of an original database by collecting both frequent itemsets and border itemsets (itemset which is not frequent itemsets but its proper subsets are frequent itemset). This algorithm is successful for reducing the number of rescanning times but a large number of border itemsets have to collect. Thus this Negative Border consumes a large amount of memory. Moreover, in the worst case, Negative Border algorithm needs to rescan an original database several times when new frequent itemsets are discovered in an updated database.

To solve with the collecting of massive border itemsets problem, Tsai et al. [7], Hong et al. [8], and Amornchewin and Kreesuradej [5] presented algorithms which are not only reduce a number of collecting border itemsets but also decrease a rescanning time. These algorithms keep both frequent itemsets and expected frequent itemsets, i.e., itemsets which are able to be frequent itemset. However, each researcher gives the quite different method to select expected frequent itemsets. Tsai et al. determined a new threshold, called tolerance degree. It collaborates with support threshold to select expected frequent itemsets. As Hong et al. assigned the 2 news thresholds, upper support and lower support threshold, to select expect frequent itemsets. Amornchewin and Kreesuradej applied the probability theory (the principle of Bernoulli trial) and defined a new threshold, $prob_{pl}$, to predict expected frequent itemsets. Our proposed algorithm was based on this direction.

As mentioned to the previous study [5], we observed that the binomial probability has a numerical problem when factorial is computed with a large value, i.e., factorial of a large value cannot fit to the size of an integer variable. To solve this problem, the probability of occurrence of expected frequent itemset estimation using normal approximation is introduced in section 4.

3. Incremental Association Rule Mining

In this section, an incremental association rule mining concept is introduced concisely. The problem statement of an incremental association rule mining is described in subsection 3.1 and the probability-based incremental association rule discovery algorithm is reviewed in subsection 3.2.

3.1 Problem Statement of the Incremental Association Rule Mining

Generally, dynamic database is categorized databases into 2 types: an original database and an increment database. The original database is the old database which old transactions are collected. The increment database is the new database which a new group of transactions are inserted into the original database. When a new increment database is merged to the original database, the association rule from the previous mining may have been changed. The problem statement for an incremental association rule discovery is normally defined as follows.

Let DB is an original database which is a collection of old transactions. db is an increment database which is a collection of new transactions. Then, UD is an updated database which is the database after merging DB and db together, i.e., UD = DB ∪ db. The number of transactions in a database is called the size of the database. Thus, the size of DB, db and UD are |DB|, |db| and |UD| = |DB|+|db| respectively. The notation of the incremental association rule mining problem statement is defined in table 1.

Table 1 The notation the incremental association rule mining problem statement

notation	meaning
DB	an original database
db	an increment database
UD	an updated database
F_k^{DB} , F_k^{UD}	frequent k-itemset of DB, and UD respectively
δ_x^{DB} , δ_x^{UD}	a support count of itemset X in DB and UD respectively

Basically, the minimum support threshold, s, is assumed to be a constant number. Before the updating activity has begun, F^{DB} is a frequent itemsets of DB if and only if support count of itemset X, δ^{DB}, is greater than or equal to s×|DB|, i.e., $\delta^{DB} \geq s \times |DB|$. After the updating activity has ended, F^{UD} is called a frequent itemsets of UD if and only if $\delta^{UD} \geq s \times |DB|$.

As mentioned to Tsai et al. [8], when an original database and increment database are merged, an itemset, i.e., X, can possibly become to 4 cases:

Case 1 : X is a frequent itemset in both DB and UD

Case 2 : X is a frequent itemset in DB and an infrequent itemset in UD

Case 3 : X is an infrequent itemset in DB and a frequent itemset in UD

Case 4 : X is an infrequent itemset in both DB and UD

From all cases mentioned above, the first two cases are easily discovered frequent itemsets in an updated database since their support count are exactly known. Accordingly, the updating tasks in these 2 cases are negligible tasks. In the fourth case, it does not necessarily keep attention because it does not need to rescan an original database. The most difficult task of these 4 cases is the third case because it needs to rescan an original database to obtain the support count of itemsets in the updating tasks. The rescanning an original database is the really big problem because a lot of I/O operations are required.

3.2 Probability-based Incremental Association Rule Discovery Algorithm

Probability-based incremental association rule discovery algorithm [5] was proposed by Amornchewin and Kreesuradej in 2009. The main idea of this algorithm is predicting expected frequent itemsets using the principle of Bernoulli trials. The expected frequent itemsets are collected during finding the frequent itemsets of an original database. This predicted expected frequent itemsets can help the algorithm to reduce the number of times to rescan the original database when the new set of transactions is appended to the original database. In this subsection, the necessary concept of probability-based algorithm is briefly described.

For probability-based incremental association rule discovery algorithm, the process of inserting m transactions into an original database of n transactions can be considered as $(n+m)$ Bernoulli trials. Each itemset has its probability of occurrence in a transaction, denoted by p. Then the probability of the occurrence of itemset in $(n+m)$ transactions, denoted by $P(X)$, can be found by the following equation:

$$P(X) = \binom{n+m}{x} . p^x . (1-p)^{n+m-x} \quad (1)$$

where p is the probability of an itemset appearing in a transaction, m is a number of new transactions, and n is a

number of transactions of an original database.

Thus, if k is a minimum support count after inserting new transactions into an original database, the probability of an itemset to be a frequent itemset in an updated database can be obtained as the following equation:

$$P(x \geq k)_{item} = 1 - P(x < k)_{item} \quad (2)$$

According to eq.1, $P(x \geq k)_{item}$ can be found as the following equations:

$$P(x \geq k)_{item} = \sum_{x=0}^{k-1} \binom{n+m}{x} p^x . (1-p)^{n+m-x} \quad (3)$$

Here, an expected frequent itemset is an itemset that is not a frequent itemset but has its probability to be a frequent itemset greater than $Prob_{pl}$. $Prob_{pl}$ is a constant threshold specified by users. $Prob_{pl}$ indicates the minimum confidence level that a promising frequent itemset will be a frequent itemset after inserting new transaction into an original database. The higher $Prob_{pl}$ is set, the lesser expected frequent itemsets are kept. As results, the algorithm may need more a number of rescanning times in the original database when the algorithm performs the discovering new frequent itemset task.

Probability-based incremental association rule discovery algorithm assumes that the statistics of old transactions obtained from previous mining can be utilized for approximating that of new transactions. Therefore, the algorithm uses support count of itemsets obtained from mining the original database to approximate the probability of itemsets when new transactions are inserted into the original database. Thus, the probability of occurrence itemset in (3), i.e., p, can be approximate as the following equation:

$$p = \frac{c(itemset, DB)}{|DB|} \quad (4)$$

where $c(itemset, DB)$ is the support count of the itemsets obtained from the original database.

However, in practice, the probability of itemset to be a frequent itemset in an updated database, $P(x \geq k)_{item}$, which obtained from binomial probability as eq. 3 suffer from a numerical problem when factorial is computed with a large value, i.e., the factorial of a large value cannot fit to the size of an integer variable.

For any positive integers of n, the factorial of n, i.e. $n!$, is the product of all integers from 1 to n. Thus, the value of n! increases swiftly with a large value of n. For a data type in a computer language, a long integer data type, which occupies 64 bits, has 2^{63}-1 range of value. This range can only store 20! However, the binomial probability as eq. 3 usually involves in computing the factorial of n that greater than 20! Therefore, the long integer type does not have enough memory to store the factorial result. Such that problem, some probabilities of itemsets cannot be computed directly as eq. 3. In the previous work, these probabilities of itemsets are approximated by a linear extrapolation in order to avoid the problem. This gives some error of probability values.

To deal with this problem, probability-based incremental association rule discovery using the normal approximation is presented in the next section.

4. Probability-based Incremental Association Rule Discovery Using Normal Approximation

As Amornchewin and Kreesuradej work [5], the process of inserting m transactions into an original database of n transactions can be considered as $(n+m)$ Bernoulli trials. Then, the probability of the occurrence of itemset in $(n+m)$ transactions, denoted by $P(X)$, can also be obtained by eq. 3. Unlike the previous work, the probability of occurrence of itemset, i.e. $P(X)$, is calculated by using normal approximation theory in this work.

4.1 Normal Approximation to the Binomial

According to Bernoulli trial, it is a random experiment that one of two outcomes is occurred: success and failure. Let p is a probability of success and $q = 1- p$ is a probability of failure. The probability mass function (p.m.f.) of X can be written as

$$f(x) = p^x (1-p)^{1-x} \quad , \quad \begin{matrix} x = 0(failure) \\ x = 1(success) \end{matrix} \quad (5)$$

Generally, Bernoulli trial is considered as the special case of binomial distribution because Binomial experiment is an experiment that consists of several repeated independent Bernoulli trials [10].

Definition1: [9]"*Let X is the number of observed success in n Bernoulli trials, then the possible values of X are 0,1,2,...,n. If x successes occur, where x = 0,1,2,...,n, then n-x failures occurs The number of ways of selecting x positions for the x success in the n trials is*"

$$\binom{n}{x} = \frac{n!}{x!(n-x)!} \quad (6)$$

Definition2: [9] *"Since the trials are independent and the probabilities of success and failure on each trial are, respectively, p and q=1-p, the probability of each of these ways is* $p^x(1-p)^{n-x}$. *Thus, f(x), the p.m.f of X, is the sum of probabilities of the* $\binom{n}{x}$ *mutually exclusive events; that is,*

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x} \quad , \quad x=0,1,2,...,n \quad (7)$$

These probabilities are called binomial probabilities and the random variable X is said to have a binomial distribution."

The cumulative probability $P(X<k)$ is calculated by

$$P(X < k) = \sum_{x=0}^{k-1} \binom{n}{x} p^x (1-p)^{n-x} \quad (8)$$

Thus, $P(X \geq k)$ can be calculated by

$$P(X \geq k) = 1 - \sum_{x=0}^{k-1} \binom{n}{x} p^x (1-p)^{n-x} \quad (9)$$

According to probability-based algorithm, an expected frequent itemset is predicted by applying eq. 9. In practice, the binomial probability has a numerical problem when factorial is computed with a large n, i.e., a factorial value of a large n cannot fit to the size of an integer variable.

Definition3:[11] "When n, a number of trials, is large and p, the probability of success, is not too far from ½. Figure 1 shows the histograms of binomial distribution with p=1/2 and n = 2,5,10 and 25, and it can be seen that with increasing n these distribution approach the symmetrical bell-shaped pattern of a normal curve"

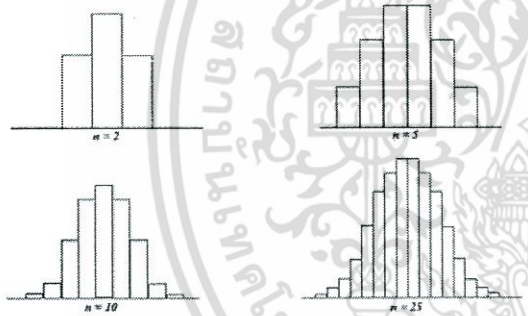


Figure 1. Binomial distribution with p = ½ and n = 2, 5, 10 and 25

When n is large enough, the normal distribution with $\mu = np$ and $\sigma = \sqrt{np(1-p)}$ can be used to approximate to the binomial distribution, when $np>5$ and $n(1-p) >5$ [11]. However, a continuity correction must be applied when the binomial distribution, which is a discrete distribution, is approximated by the normal distribution.

Here, given $\mu = np$, $\sigma = \sqrt{np(1-p)}$ and X be a number of success, the area under the curve to the left of x is defined as

$$P(X \leq k) = \int_{-\infty}^{k} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} . \quad (10)$$

By using the continuity correction, the number of successes, i.e. k, is decreased by 0.5. Accordingly, the probability success k times can be derived as the following equation

$$P(X \geq k) = 1 - \sum_{k=0}^{k-1} \int_{k-0.5}^{k+0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \quad (11)$$

Therefore, eq. 11 can be derived to the following equation

$$\mathrm{Pr}obEF_X = P(X \geq k) = 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx , \quad (12)$$

where $\mu = np$, $\sigma = \sqrt{np(1-p)}$, n is a number of transaction of updated database and p is the probability of occurrence itemsets. For the proposed algorithm, eq. 12 is used to calculate the probability of an expected itemset instead of eq. 3.

Since the estimation of the probability of an expected itemset based on eq. 12 is the point estimation, the actual probability of an expected itemset may vary from the estimation of the probability of an expected itemset. Therefore, the proposed algorithm also introduces a probability tolerance threshold to deal with the difference between the estimated probability and the actual probability. Probability tolerance threshold ensures that an expected frequent itemset is tolerably kept. Probability tolerance threshold is based on a statistical confidence interval.

Typically, the confidence interval of the probability of an occurrence of an itemset, i.e., p, can be defined as following:

$$p = p \pm Z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} . \quad (13)$$

According to eq. 13, there are two threshold values: an upper threshold and a lower threshold. To ensures that an expected frequent itemset is tolerably kept, Probability Tolerance Threshold of an itemset (ptt_X) is defined as

$$ptt_X = p + Z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} \quad (14)$$

Here, ptt_x in eq. 14 is used in eq. 12 instead of p. Then, an itemset is an expected frequent itemset if its probability of an itemset is equal to or greater than $prob_{pl}$. These expected frequent itemsets are kept in order to reduce the number of times to rescan an original database. The proposed algorithm is presented in the next section.

4.2 Probability-based Incremental Association Rule Discovery Algorithm Using the Normal Approximation

The proposed algorithm, called Probability-Based Incremental Association Rule Discovery Algorithm Using The Normal Approximation, is based on the probability-based incremental association rule discovery algorithm. There are 2 main phases: original mining and incremental

mining phase. The notation used in this 2 main phases is defined in table 2.

Table 2 The notation of probability-based incremental association rule discovery using the normal approximation algorithm

notation	meaning
F_k^{DB} , F_k^{db} , F_k^{UD}	frequent k-itemset of DB, db and UD respectively
EF_k^{DB} , EF_k^{UD}	expected frequent k-itemset of DB and UD respectively
δ_x^{DB} , δ_x^{db} , δ_x^{UD}	a support count of itemset X in DB , db and UD respectively
C_1^{DB} , C_1^{UD}	Candidate 1-itemset of DB and UD
s	a minimum support threshold
ρ^{DB} , ρ^{UD}	support count of minimum probability of itemset in DB and UD respectively
Z	confidence interval
$Prob_{pt}$	probability value threshold defined by user

Algorithm1: Original Mining Phase
Input: DB, |db|, $Prob_{pt}$, s, Z
Output: F_k^{DB} , EF_k^{DB} , C_1^{DB} , ρ^{DB}

1 k=1
2 scan DB for all X∈ C_k and obtain δ_x^{DB}
3 F_k^{DB} ={X | $\delta_x^{DB} \geq s\times|DB|$}
4 for {X | $\delta_x^{DB} < s\times|DB|$}
5 calculate ptt_x //using equation (14)
6 calculate probability of expected itemset X ($ProbEF_X$) //using equation (12)
7 $\rho^{DB} = \min(\delta_x^{DB} | ProbEF_x \geq Prob_{pt})$
8 EF_k^{DB} = {X | $s\times|DB| > \delta_x^{DB} \geq \rho^{DB}$ }
9 C_1^{DB} = {X| $\delta_x^{DB} < \rho^{DB}$ }
10 end
11 k=2
12 while | $F_k^{DB} \cup EF_k^{DB}$ | > 1
13 C_k = ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$)*($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$)
14 repeat line 2-8
15 k++
16 end while loop
17 Return F_k^{DB} , EF_k^{DB} , C_1^{DB} , ρ^{DB}

Figure 2. Original Mining Phase Algorithm

For the first iteration (k = 1) of original database phase as shown in figure 2, itemsets are scanned to an original database and obtained their support count. Then, frequent itemsets are found as line 3. The remaining infrequent itemset are computed to obtain two values: the probability tolerance threshold of an itemset (ptt_X) and the probability of an expected itemset ($ProbEF_X$) as line 5 and 6, respectively. After that, the support count of the minimum of $ProbEF_X$, i.e., ρ^{DB}, is obtained from line 7. This ρ^{DB} indicates the possible minimum support count of expected frequent itemset. Next, an infrequent itemsets which its support count is less than s*|DB| and greater than or equal to ρ^{DB} is collected to the set of expected frequent itemset as line 8. In addition, candidate 1-itemsets are obtained from line 9.

For the second iteration and beyond ($k \geq 2$), the apriori_gen concept is used to generate candidate k-itemsets. Unlike apriori_gen, the candidate k-itemset of the proposed algorithm is generated by ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$)* ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$) as line 13. After the candidate k-itemsets are obtained, the other steps is performed as the first iteration as line 14 and $C_{k>=2}$ does not collect. These steps of k≥2 iteration are operated until candidate k-itemset cannot generate. When the original mining phase is ended, all outputs of this phase are F_k^{DB} , EF_k^{DB} , C_1^{DB} , ρ^{DB} as line 17. These out puts are used in the next phase.

Algorithm2: Incremental Mining Phase
Input: DB, db, $Prob_{pt}$, s, F_k^{DB} , EF_k^{DB} , C_1^{DB} , ρ^{DB} ,Z
Output: F_k^{UD} , EF_k^{UD} , C_1^{UD} , ρ^{UD}

1 k=1
2 scan db and updating support count to obtain δ_x^{UD}
3 F_k^{UD} ={X | $\delta_x^{UD} \geq s\times|UD|$}
4 for {X | $\delta_x^{UD} < s\times|UD|$}
5 calculate ptt_x //using equation (14)
6 calculate probability of expected itemset X ($ProbEF_X$) //using equation (12)
7 $\rho^{UD} = \min(\delta_x^{DB} | ProbEF_x \geq Prob_{pt})$
8 EF_k^{UD} = {X | $s\times|UD| > \delta_x^{UD} \geq \rho^{UD}$ }
9 C_1^{UD} = {X| $\delta_x^{UD} < \rho^{UD}$ }
10 end
11 for k = 2
12 while $F_{k-1}^{UD} \neq \emptyset$
13 Generating Candidate itemset ( ) // call algorithm 3
14 repeat line 2-8
15 for X∈ C_k^{new} // C_k^{new} *is obtained from line 12*
16 $Temp\_scanDB$ = {X | (δ_x^{db} +(ρ^{DB} -1)) ≥ $s\times|UD|$}
17 end
18 k++
19 end while loop
20 end
21 if $Temp\_scanDB \neq \emptyset$
22 Rescanning Original Database to obtain new F_k^{UD} *and* EF_k^{UD}
23 endif
24 clear $Temp\_scanDB$
25 Return F_k^{UD} , EF_k^{UD} , C_1^{UD} , ρ^{UD}

Figure 3. Incremental Mining Phase Algorithm

The incremental mining phase, there are 2 main tasks in this phase: k=*1* iteration and $k \geq 2$ iteration.

For the iteration of k=*1*, an increment database, db, is scanned and updated the support count of itemsets as shown in line 2. Then frequent 1-itemsets of updated

database, F_k^{UD}, are found as line 3. For itemsets which are not become F_k^{UD}, they are calculated the Probability tolerance threshold of an itemset (ptt_X) and the probability of an expected itemset ($ProbEF_X$) as eq.14 and eq.12. After ρ^{UD} is calculated as line 7, an expected frequent itemset (EF_k^{UD}) and candidate 1-itemset (C_1^{UD}) of an updated database is obtained as line 8 and 9, respectively.

For the iteration of k ≥ 2, in generally, this task is performed as the first iteration. However, in this iteration, there are 2 added steps: generating candidate itemset and rescanning original database in line 13 and 22, respectively.

The generating candidate itemset is detailed in figure 4. Generating candidate 2-itemset and candidate k≥2 itemset is different. For candidate 2-itemset, it is generated by $F_1^{UD} * F_1^{UD}$ as line 2, while candidate k≥2 itemset is generated by $F_{k-1}^{db} * F_{k-1}^{db}$ as line 8 when F_{k-1}^{db} is obtained from itemset which its support count is greater than or equal to s*|db|. To ensure that no candidate itemsets are lost, itemset which is a member of $(F_{k-1}^{DB} \cup EF_{k-1}^{DB})$ is also brought to consider as line 5 and 6. In addition, the new candidate k-itemset (C_k^{new}) is obtained from line 3 and 9. This C_k^{new} are itemsets which their support count is only known in db but not in DB. These new candidate itemsets are reused in line 15 of figure 3 to extract *Temp_scanDB*.

Algorithm3: Generating Candidate Itemset
Input: C_k^{db}, F_1^{UD}, F_k^{DB}, EF_k^{DB}, s, |db|
Output: C_k^{db}, C_k^{new}
1 if k = 2
2 $C_2^{db} = F_1^{UD} * F_1^{UD}$
3 $C_2^{new} = \{X \in C_2^{db} \mid X \notin (F_2^{DB} \cup EF_2^{DB})\}$
4 else if k ≥ 3
5 $X = F_{k-1}^{DB} \cup EF_{k-1}^{DB} \cup C_{k-1}^{db}$
6 $\delta_x^{UD} = \delta_x^{DB} + \delta_x^{db}$
7 $F_{k-1}^{db} = \{X \mid \delta_x^{UD} \geq s^*|db|\}$
8 $C_k^{db} = F_{k-1}^{db} * F_{k-1}^{db}$
9 $C_k^{new} = \{X \in C_k^{db} \mid X \notin (F_k^{DB} \cup EF_k^{DB})\}$
10 end if
11 Return C_k^{db}, C_k^{new}

Figure 4. Generating Candidate Itemset

The new candidate itemsets (C_k^{new}) will be collected to *Temp_scanDB* if and only if the support count of C_k^{new} plus (ρ^{DB} -1), where ρ^{DB} obtained from the original mining phase, is greater than or equal to s*|UD|. This process is shown in line 16 of figure 3 for pruning C_k^{new} which impossibly be a frequent itemset or an expected frequent itemset of the updated database.

For the final step of an incremental mining phase, *Temp_scanDB* is rescanned to an original database to update support count and find new frequent itemset and expected frequent itemset of an updated database as shown in line 21-23 of figure 3. Both new frequent itemset and new expected frequent itemset are merged to the set of frequent itemset and expected frequent itemset which are found before. This guarantees that there is no remaining frequent itemset and expected frequent itemset to found absolutely.

5. Experiments

The objective of the proposed algorithm in this paper is to solve the numerical problem occurs with binomial distribution. Moreover, the confidence interval is introduced to increase the probability value of an itemset to be a frequent itemset in an updated database. The hypothesis is the proposed algorithm can reduce the number of *Temp_scanDB* itemsets which is rescanned to the original database.

To evaluate the efficiency of our algorithm, probability-based incremental association rule discovery using the normal approximation, this algorithm is implemented and tested on a PC with 2.93 GHz Intel Core i7, and the main memory is 3 GB. The experiment is tested with a synthetic dataset which are generated by synthetic technique proposed by Agrawal [1]. The synthetic dataset is T10I4D100K with the original database of 10,000 transactions. A number of transactions of increment database appended to an original database are 3,000 and 5,000 respectively.

The experiment is conducted with 0.005% minimum support threshold and $Prob_{pl}$ = 0.3. The average of execution time compared with Apriori, FUP and probability-based algorithm are demonstrated in table 3 and 4 respectively.

Table 3. Average of Execution time for adding 3,000 transactions

Algorithm	Apriori	FUP	Probability-based	Prob-based using Normal Approximation
Execution time (sec)	61,452.0120	34,951.3102	16,252.6708	14,656.8360
a number of frequent itemset	1,105	1,105	1,105	1,105
a number of collected expected frequent itemsets	-	-	40	115
maximum frequent itemset (size of k)	F_5	F_5	F_5	F_5
a number of Temp_scanDB	-	-	95	31

Table 4. Average of Execution time for adding 5,000 transactions

Algorithm	Apriori	FUP	Probability-based	Prob-based using Normal Approximation
Execution time (sec)	69,215.6714	42,219.5093	26,826.7789	24,316.6723
a number of frequent itemset	1,097	1,097	1,097	1,097
a number of collected expected frequent itemsets	-	-	50	137
maximum frequent itemset (size of k)	F_4	F_4	F_4	F_4
a number of Temp_scanDB	-	-	85	39

The experiment results from table 3 and 4 display that the execution time of the proposed algorithm, the probability-based using the normal approximation algorithm, is concisely better than Apriori and FUP algorithm because Apriori needs to rerun a whole database (the merging between the original database and increment database) and FUP needs k-times (as size of k) to rescan the original database. While the proposed algorithm needs only one time to rescan the original database.

For comparing between the proposed algorithm and probability-based algorithm, the execution time of the proposed algorithm is slightly better than probability-base algorithm. A number of collected expected frequent itemset the proposed algorithm is greater than the probability-based algorithm. This affects to the number of *Temp_scanDB*, that is the number of *Temp_scanDB* of the proposed algorithm is less the than probability-based algorithm. These experiment results demonstrate that our hypothesis is accepted, i.e., the proposed algorithm can decrease the number of *Temp_scanDB*. As a result, the proposed algorithm can reduce a time-consuming of rescanning the original database.

6. Conclusion

In this paper, we propose the incremental association rule algorithm, probability-based incremental association rule discovery using the normal approximation algorithm, which estimates the probability of occurrence of expected frequent itemset using normal approximation. The proposed algorithm can deal with the numerical problem occurred when large n factorial is computed by binomial distribution. In addition, we introduce the increasing probability value of an itemset to be a frequent itemset in an updated database using confidence interval. The experiment results show the slightly better execution time when compare with the probability-based algorithm and the decreasing of a number of *Temp_scanDB*. However, we encounter with trade off a number of Temp_scanDB against a number of collected expected frequent itemset.

However, the common limitation of the proposed algorithm and probability-based algorithm is the fixed size of the increment database. That is, both algorithms need to know the exact number of the increment database size for computing the probability value of the co-occurrence itemset in the updated database. Practically, the increment database may be continually added to the original database with various sizes. Thus, the future research is looking for the approach to deal with the fixed increment database size problem.

7. References

[1] R Agrawal, T Imielinski, A Swami, "A mining association rules between sets of items in large database," in proceeding of the ACM SIGMOD Int'l Conf. on Management of Data (ACM SIGMOD'93), Washington, USA, May 1993, pp.207-216.

[2] R Agrawal, R Srikant, "Fast algorithm for mining association rules," in Proc. 20th Int. Conf. Very Large Databases (VLDB'94), Santiago, Chile, September 12-15, 1994, pp.487-499.

[3] C. H. Lee, C. R. Lin, and M. S. Chen, "Sliding-window Filtering: An Efficient Algorithm for Incremental Mining," Proceeding of the ACM 10th International Conference on Information and Knowledge Management (CIKM'01), November 2001. pp 263-270.

[4] H Toivonen, " Sampling large database for association rules," In Proceedings of the 22th International Conference on Very Large Database (VLDB'96), September 1996, pp. 134-145.

[5] R Amornchewin, Kreesuradej Worapoj, "Mining Dynamic Databases using Probability-based Incremental association Rule Discovery Algorithm," Journal of Universal Computer Science, Vol.15, No.12, April 2009, pp. 2409-2428.

[6] D W Cheng, J Han, V T Ng, C Y Wong, "Maintenance of Discovered Association Rules in Large Databases: An incremental updating technique," In 12th IEEE International conference on Data Engineering, 1996, pp.106-114.

[7] S M Tsai Paulry, Lee Chin-Chong, L P Chen Arbee, "An Efficient Approach for incremental Association Rule Mining," Proceedings of the third Pacific-Asia Conference on methodologies for Knowledge Discovery and Data Mining, Lecture Notes in Computer Science, Vol. 1574 archive, 1999.

[8] T P Hong, C Y Wang, Y H Tao, "A new incremental data minting algorithm using pre-large itemsets," Journal of Intelligent Data Analysis, Vol.5, No.2, 2001, pp.111-129.

[9] V. H Robert, A.T Elliot, "2.4 Bernoulli trials and the binomial distribution," *Probability and Statistical Inference.* 8th edi, Pearson Education, Inc, New Jersey, 2010, pp. 78-86.

[10] J. K Larry, "4.4 The Binomial Distribution," *Exploring Statistics: A Modern introduction to Data Analysis and Inferernce,* 2nd edi., Integre Technical Publishing company, Inc.; G&S Typesetters, Inc, 1998, pp.265-275.

[11] E. F John, M. P Benjamin, "7.4 The normal approximation to the binomial distribution," *Statistics: A first course,* 8th edi., Pearson Education, Inc, New Jersey, 2004, pp. 256-260.