การเพิ่มขนาดกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ

INCREMENTAL ASSOCIATION RULE USING THE NORMAL DISTRIBUTION APPROXIMATION

อารยา อริยา

ARAYA ARIYA

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2559

KMITL-2016-IT-D-001-002

การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ

INCREMENTAL ASSOCIATION RULE USING THE NORMAL DISTRIBUTION APPROXIMATION

อารยา อริยา

ARAYA ARIYA

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2559

KMITL-2016-IT-D-001-002

INCREMENTAL ASSOCIATION RULE USING THE NORMAL DISTRIBUTION APPROXIMATION

ARAYA ARIYA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2016

KMITL-2016-IT-D-001-002

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ

Incremental Association Rule using the Normal Distribution Approximation

นักศึกษา นางสาวอารยา อริยา

รหัสประจำตัว 51066304

ปริญญา ปรัชญาดุษฎีบัณฑิต

สาขาวิชา เทคโนโลยีสารสนเทศ

อาจารย์ที่ปรึกษาวิทยานิพนธ์ รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช

คณะกรรมการสอบวิทยานิพนธ์	ลายมือชื่อ
ผู้ช่วยศาสตราจารย์ ดร.กันต์พงษ์ วรรัตน์ปัญญา	
ผู้ช่วยศาสตราจารย์ ดร.สุพจน์ นิตย์สุวัฒน์	
รองศาสตราจารย์ ดร.วรพจน์ กรีสุระเดช	
รองศาสตราจารย์ ดร.พรฤดี เนติโสภากุล	
รองศาสตราจารย์ ดร.อาริต ธรรมโน	

วัน / เดือน/ ปี ที่สอบ วันศุกร์ที่ 15 กรกฎาคม 2559 เวลา 10.00 น. เป็นต้นไป

สถานที่สอบ ณ ห้อง M03 คณะเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศรับรองแล้ว

(รองศาสตราจารย์ ดร.พนพร โชติกกำธร)

คณบดีคณะเทคโนโลยีสารสนเทศ

วันที่ 25 **เดือน** กรกฎาคม **พ.ศ.** 2559

หัวข้อวิทยานิพนธ์	การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ
นักศึกษา	นางสาวอารยา อริยา
รหัสประจำตัว	51066304
ปริญญา	ปรัชญาดุษฎีบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
พ.ศ.	2559
อาจารย์ที่ปรึกษา	รศ.ดร.วรพจน์ กรีสุระเดช

บทคัดย่อ

งานวิจัยฉบับนี้นำเสนอขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ มีวัตถุประสงค์เพื่อนำหลักการประมาณค่าด้วยการแจกแจงปกติมาแก้ปัญหาการคำนวณค่าแฟคทอเรียลในการคำนวณค่าความน่าจะเป็นแบบทวินาม นอกจากนี้งานวิจัยได้พัฒนาขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักการประมาณค่าแบบการแจกแจงปกติในการประมาณค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยได้โดยไม่จำเป็นต้องทราบขนาดฐานข้อมูลใหม่ และเพื่อหลีกเลี่ยงการคำนวณค่าความน่าจะเป็นของไอเท็มเซตทุกตัว งานวิจัยนี้ได้เสนอการคำนวณค่าที่ใช้ทดสอบเพื่อหาไอเท็มเซตที่คาดว่าจะเกิดบ่อย ด้วยฟังก์ชันความน่าจะเป็นอย่างต่อเนื่องของไอเท็มเซต ในการทดสอบประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ได้ทดสอบความถูกต้องโดยนำไอเท็มเซตที่เกิดบ่อยที่ค้นพบโดยขั้นตอนวิธีที่นำเสนอเปรียบเทียบกับไอเท็มเซตที่เกิดบ่อยที่ค้นพบโดยขั้นตอนวิธีอะพริโอริ ผลการทดลองพบว่าขั้นตอนวิธีที่นำเสนอมีการค้นหาไอเท็มเซตที่เกิดบ่อยได้ถูกต้องและครบถ้วน การวัดประสิทธิภาพทางด้านเวลาในการประมวลผล งานวิจัยนี้ได้ทดสอบโดยเปรียบเทียบเวลาที่ใช้ในการประมวลผลกับขั้นตอนวิธีด้านการเพิ่มขยายกฎความสัมพันธ์ที่มีการนำเสนอก่อนหน้า ได้แก่ขั้นตอนวิธีเอฟยูพี เนกาทีฟบอร์เดอร์ พรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น ผลการทดลองพบว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ใช้เวลาในการประมวลผลที่รวดเร็วกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ที่ทำการเปรียบเทียบ

Thesis Title	Incremental Association Rule Using The Normal Distribution Approximation
Student	Miss Araya Ariya
Student ID.	51066304
Degree	Doctor of Philosophy
Program	Information Technology
Year	2016
Thesis Advisor	Assoc.Prof.Dr.Worapoj Kreesuradej

ABSTRACT

This research aims to propose the algorithm of an incremental association rule using the normal distribution approximation. Normal distribution approximation is applied to this proposed algorithm for dealing with the factorial computation problem of binomial distribution. To operate with different sizes of increment database, normal distribution approximation is used to estimate the expected frequent itemset without knowing the increment database size. Moreover, to avoid computing the probability of occurrence for all itemsets, the probability density function of itemset is used to find a threshold for selecting expected frequent itemsets. The accuracy of the proposed algorithm is compared with Apriori. The results show that all frequent itemsets discovered by both Apriori and the proposed algorithm are the same. That means the proposed algorithm can discover frequent itemsets correctly. The performance of the proposed algorithm is compared with the previous incremental association rule algorithms such as FUP, Negative Border, Pre-Large and Probability-based algorithm. The experimental results show that the execution time of the proposed algorithm is better than the compared algorithms.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ.ดร.วรพจน์ กรีสุระเดช ที่ให้ความช่วยเหลือ ให้คำชี้แนะช่วยแก้ปัญหาตลอดจนให้ความรู้และประสบการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบพระคุณ ผศ.ดร.สุพจน์ นิตย์สุวัฒน์ และท่านคณะกรรมการสอบหัวข้อและโครงร่างวิทยานิพนธ์ทุกท่านที่ได้กรุณาให้คำแนะนำตลอดจนข้อชี้แนะ เพื่อให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

ขอขอบพระคุณสำนักงานคณะกรรมการการอุดมศึกษาแห่งชาติและมหาวิทยาลัยราชภัฏลำปางที่ได้ให้ทุนสนับสนุนและค่าใช้จ่ายในการเรียน และขอขอบพระคุณบิดามารดาที่เป็นกำลังใจที่ดีตลอดมา

สำหรับคุณงามความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดาตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีแก่ข้าพเจ้า

อารยา อริยา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญตาราง (ต่อ)

ตารางที่ **หน้า**

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ความก้าวหน้าทางเทคโนโลยีการจัดเก็บข้อมูลในปัจจุบันทำให้องค์กรสามารถจัดเก็บข้อมูลได้มากขึ้นด้วยต้นทุนที่ไม่สูงมากนักเมื่อเทียบกับอดีต โดยเฉพาะองค์กรทางธุรกิจที่มีการจัดเก็บข้อมูลมากมาย โดยเฉพาะข้อมูลรายการธุรกรรม (Transaction data) ที่เกิดขึ้นทุกวัน วันละหลายรายการ อาทิ ข้อมูลการซื้อขายสินค้า ข้อมูลธุรกรรมทางการเงิน เช่น การฝากเงิน การถอนเงิน การโอนเงิน ข้อมูลการผลิตของโรงงาน ข้อมูลการส่งออก-นำเข้าสินค้า เป็นต้น ในสมัยอดีตการใช้ประโยชน์จากข้อมูลที่ถูกจัดเก็บไว้ จะอยู่ในรูปของการเรียงลำดับข้อมูล การจัดทำสถิติ การดึงข้อมูลจากฐานข้อมูลมาเพื่อทำรายงานสรุปแก่ผู้บริหาร เหล่านี้ล้วนแต่เป็นการจัดการข้อมูลที่ผู้ใช้มีความต้องการและทราบอยู่แล้วว่าจะนำข้อมูลมาใช้ได้อย่างไร อย่างไรก็ตาม ในฐานข้อมูลที่เก็บข้อมูลจำนวนมหาศาลนี้ ยังมีสารสนเทศที่ซ่อนเร้นอยู่ที่ผู้ใช้ไม่ทราบมาก่อน โดยจะต้องใช้กระบวนการและเทคนิคทางด้านคอมพิวเตอร์และสถิติมาเพื่อสกัดสารสนเทศเหล่านี้ ซึ่งสารสนเทศดังกล่าวล้วนเป็นประโยชน์ต่อองค์กรอย่างมาก

การทำเหมืองข้อมูล (Data mining) หรือบางครั้งเรียกว่า กระบวนการค้นหาความรู้ในฐานข้อมูล (Knowledge Discovery in Database: KDD) เป็นกระบวนการที่ใช้ในการค้นหารูปแบบหรือความสัมพันธ์ที่ซ่อนอยู่ในข้อมูลจำนวนมหาศาลที่ถูกจัดเก็บไว้โดยอัตโนมัติ ปัจจุบันมีหลายองค์กรพยายามนำเอาหลักการของการทำเหมืองข้อมูลไปใช้ในระบบสารสนเทศของตนเพื่อสร้างความได้เปรียบให้แก่องค์กร เช่น การใช้เทคนิคการทำเหมืองข้อมูลในการจัดกลุ่มลูกค้าเงินกู้ชั้นดีของธนาคาร การจำแนกกลุ่มลูกค้าที่มีความสามารถในการซื้อสินค้าราคาสูงจำพวกบ้านและรถยนต์ การตรวจจับความผิดปกติในการทำธุรกรรมทางการเงิน เป็นต้น

การค้นหากฎความสัมพันธ์ (Association rule discovery) เป็นหนึ่งในเทคนิคที่สำคัญและได้รับความนิยมในกระบวนการทำเหมืองข้อมูล เพื่อค้นหารูปแบบความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูล ผลที่ได้จากการประมวลผลจะเป็นการสกัดรูปแบบข้อมูลที่น่าสนใจให้ออกมาในรูปแบบของกฎความสัมพันธ์ ถ้า....แล้ว.... (IF X THEN Y) โดยมีหลักการทำงาน 2 ขั้นตอนหลัก ได้แก่ 1) การค้นหาไอเท็มเซตที่เรียกว่า ไอเท็มเซตที่เกิดบ่อย (Frequent itemset) ซึ่งเป็นไอเท็มที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ (Minimum support) ที่ผู้ใช้กำหนด 2) นำไอเท็มเซตที่เกิดบ่อยที่ได้จากขั้นตอนแรกมาสร้างกฎความสัมพันธ์ โดยกฎความสัมพันธ์ที่น่าสนใจ จะเป็นกฎความสัมพันธ์ที่มีค่าความเชื่อมั่นมากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำ (Minimum confidence) ซึ่งถูกกำหนดโดยผู้ใช้

สำหรับงานวิจัยทางด้านการค้นหากฎความสัมพันธ์ ขั้นตอนวิธีที่ได้รับการยอมรับและเป็นที่นิยมสำหรับการหากฎความสัมพันธ์ คือ ขั้นตอนวิธีอะพริโอริ (Apriori) [1] ที่ถูกเสนอโดย Agrawal และคณะ ขั้นตอนวิธีนี้ใช้สำหรับค้นหาไอเท็มเซตที่เกิดบ่อย ประกอบด้วย 2 ขั้นตอนหลัก ได้แก่ การสร้างไอเท็มเซตคู่แข่ง (Candidate itemset) ด้วยกระบวนการเชื่อมไอเท็มเซต (Join) และการตัด (Prune) ไอเท็มเซตคู่แข่งที่ไม่สามารถเป็นไอเท็มเซตที่เกิดบ่อยทิ้งไป

อย่างไรก็ตาม เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิม การค้นหากฎความสัมพันธ์ด้วยวิธีการของอะพริโอริ จะต้องทำการประมวลผลข้อมูลในฐานข้อมูลทั้งหมดอีกครั้ง เพื่อสร้างไอเท็มเซตคู่แข่งใหม่ และสแกนฐานข้อมูลเพื่อหาค่าสนับสนุนของไอเท็มเซตคู่แข่งใหม่ในทุกๆ รอบ ส่งผลต่อเวลาที่ใช้ในการประมวลผลถูกใช้มากเกินความจำเป็นและไม่เกิดประสิทธิภาพในการทำงาน ยกตัวอย่างเช่น สมมติความยาวของไอเท็มเซตที่เกิดบ่อยมีค่าเท่ากับ 3 ($L_{k=3}$) หากต้องการปรับปรุงไอเท็มเซตที่เกิดบ่อยให้เป็นปัจจุบัน ขั้นตอนวิธีอะพริโอริ จะต้องประมวลผลฐานข้อมูลทั้งเก่าและใหม่ตั้งแต่ต้นรวม 3 รอบ

ด้วยข้อด้อยดังกล่าวของขั้นตอนวิธีอะพริโอริ จึงได้มีผู้นำเสนอการปรับปรุงกฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิม เช่น Cheng และคณะ [2] เสนอขั้นตอนวิธีเอฟยูพี (Fast update algorithm: FUP) ซึ่งเป็นขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ (Incremental association rule discovery) โดยอาศัยองค์ความรู้เดิมที่ได้จากการประมวลผลในฐานข้อมูลเดิมมาช่วยเพิ่มประสิทธิภาพในการค้นหากฎความสัมพันธ์ นั่นคือ ขั้นตอนวิธีเอฟยูพี จะใช้ไอเท็มเซตที่เกิดบ่อยที่ได้จากการประมวลผลในฐานข้อมูลเดิมมาช่วยลดจำนวนไอเท็มเซตคู่แข่งที่จะถูกนำไปสแกนในฐานข้อมูลเดิม ด้วยวิธีการดังกล่าว ทำให้เอฟยูพี ใช้เวลาที่ใช้ในการประมวลผลน้อยกว่า อะพริโอริ

อย่างไรก็ตามในแต่ละรอบ k เมื่อมีการค้นพบไอเท็มเซตคู่แข่งที่ไม่ได้เป็นสมาชิกของไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม ไอเท็มเซตคู่แข่งนั้นๆ จะถูกนำไปสแกนในฐานข้อมูลเดิมทุกรอบ k นั่นหมายความว่า หากขนาดของไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุงมีความยาวเท่ากับ 5 ($L_{k=5}$) ขั้นตอนวิธีเอฟยูพี จะต้องนำไอเท็มเซตคู่แข่งไปสแกนในฐานข้อมูลเดิมรวมทั้งสิ้น 5 รอบ เช่นเดียวกับขั้นตอนวิธีอะพริโอริ ต่างกันตรงที่จำนวนไอเท็มเซตคู่แข่งที่ถูกนำไปสแกนในฐานข้อมูลเดิมของเอฟยูพี จะมีจำนวนน้อยกว่าอะพริโอริ เท่านั้น

เพื่อลดจำนวนการสแกนฐานข้อมูลเดิมให้เหลือจำนวนรอบการสแกนที่น้อยที่สุด ได้มีนักวิจัยนำเสนอขั้นตอนวิธีเพื่อลดปัญหาการนำไอเท็มเซตกลับไปสแกนฐานข้อมูลเดิม อาทิ ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ (Negative Border) [3,4] ขั้นตอนวิธีพรีลาร์จ (Pre-Large) [5] และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น (Probability-based) [6]

ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ ลดจำนวนรอบการสแกนฐานข้อมูลด้วยการเก็บทั้งไอเท็มเซตที่เกิดบ่อยและไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่เรียกว่าบอร์เดอร์ไอเท็มเซต (Border itemset) ทำให้สามารถลดจำนวนรอบการสแกนฐานข้อมูลเดิมได้อย่างมากสุดเพียง 1 ครั้ง อย่างไรก็ดี ด้วยการเก็บบอร์เดอร์ไอเท็มเซตจำนวนมหาศาล ทำให้สิ้นเปลืองพื้นที่ในการจัดเก็บและต้องใช้เวลาในการเชื่อมไอเท็มเซต มากขึ้นกว่าเดิม ขั้นตอนวิธี พรีลาร์จ และ ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น จึงถูกเสนอแนวคิดเพื่อลดจำนวนบอร์เดอร์ไอเท็มเซต โดย พรีลาร์จ จะกำหนดให้มีพารามิเตอร์เพิ่มอีก 2 ตัว คือ ค่าสนับสนุนที่เป็นขอบเขตบน (Upper support) และค่าสนับสนุนที่เป็นขอบเขตล่าง (Lower support) เพื่อกรองไอเท็มเซตที่ไม่ผ่านค่าพารามิเตอร์นี้ออกไป ทำให้จำนวนบอร์เดอร์เซตลดลง ส่วนขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นได้นำเสนอเทคนิคการประมาณค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยสำหรับเก็บไว้เพื่อนำไปประมวลผลในรอบการปรับปรุงไอเท็มเซตและสามารถลดจำนวนครั้งของการสแกนฐานข้อมูลเดิมให้เหลือเพียงครั้งเดียว

อย่างไรก็ตามโดยพื้นฐานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นซึ่งใช้หลักการความน่าจะเป็นของเบอร์นูลลี่ (Bernoulli trials) มาทำนายไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย ทำให้มีปัญหาในการหาความน่าจะเป็นในกรณีที่ต้องคำนวณค่าแฟคทอเรียลของจำนวนจริงที่มีค่ามาก งานวิจัยดังกล่าวจึงอาศัยหลักการประมาณค่าความน่าจะเป็นด้วยการเทียบค่า ทำให้ความน่าจะเป็นมีความคาดเคลื่อนไปจากความเป็นจริง อีกทั้งขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น จำเป็นต้องมีการคำนวณค่าความน่าจะเป็นของไอเท็มเซตทุกตัว ซึ่งหากมีไอเท็มจำนวนมาก จะทำให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นสูญเสียเวลาในการคำนวณค่าความน่าจะเป็นดังกล่าว นอกจากนั้นขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นยังมีข้อจำกัดในด้านขนาดของชุดข้อมูลใหม่ที่จะเพิ่มเข้ามาที่จะต้องทราบแน่นอน จึงจะสามารถคำนวณค่าความน่าจะเป็นของไอเท็มเซตได้ ซึ่งในความเป็นจริงชุดข้อมูลใหม่ที่ถูกเพิ่มเข้ามาอาจจะมีขนาดไม่เท่ากันทุกครั้ง ดังนั้นหากมีการใช้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นเพื่อการประมวลผล จะต้องมีการประมวลผลในรอบฐานข้อมูลเดิมก่อนทุกครั้งเสมอ ซึ่งจะเป็นการลดประสิทธิภาพของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

ด้วยปัญหาดังกล่าวข้างต้นของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นงานวิจัยฉบับนี้จึงดำเนินการศึกษาค้นคว้าเพื่อปรับปรุงขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจกปกติ เพื่อนำหลักการประมาณค่าด้วยการแจกแจงแบบปกติมาประยุกต์ใช้เพื่อให้ขั้นตอนวิธีที่นำเสนอในงานวิจัยฉบับนี้ สามารถแก้ปัญหาการ

คำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัว ปัญหาการคำนวณค่าแฟคทอเรียล และปัญหาการจำกัดขนาดของข้อมูลชุดใหม่ที่จะถูกเพิ่มเข้ามาในฐานข้อมูลเดิม

1.2 วัตถุประสงค์ของการศึกษา

งานวิจัยเรื่อง การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติ มีวัตถุประสงค์ของการศึกษา ดังนี้

1. เพื่อศึกษาและพัฒนาขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ให้มีการทำงานที่รวดเร็วขึ้นเมื่อข้อมูลชุดใหม่ถูกเพิ่มเข้าไปในฐานข้อมูลเดิม โดยแก้ไขปัญหาการคำนวณแฟคทอเรียล การลดขั้นตอนการคำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัว และการไม่จำกัดขนาดของการเพิ่มชุดข้อมูลใหม่

2. เพื่อทดสอบประสิทธิภาพของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติ

1.3 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

ทฤษฎีและแนวคิดต่างๆ ที่นำมาประยุกต์ใช้และเป็นแนวทางในการวิจัยเรื่องการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ประกอบด้วย

1.3.1 การค้นหากฎความสัมพันธ์ (Association rule discovery)

1.3.2 การเพิ่มขยายกฎความสัมพันธ์ (Incremental association rule discovery)

1.3.3 ทฤษฎีการหาความน่าจะเป็นแบบเบอร์นูลลี (Bernoulli theory)

1.3.4 ทฤษฎีการประมาณค่าทวินามด้วยการแจกแจงปกติ (Normal approximation to binomial theory)

1.4 ขอบเขตของการวิจัย

งานวิจัยเรื่องการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ มีขอบเขตการวิจัยดังนี้

1.4.1 งานวิจัยนี้เป็นงานวิจัยเกี่ยวกับการเพิ่มขยายกฎความสัมพันธ์โดยใช้หลักการทำงานของขั้นตอนวิธีอะพริโอริเป็นฐานในการค้นหาไอเท็มเซตที่เกิดบ่อย

1.4.2 งานวิจัยนี้จะศึกษาเฉพาะกรณีที่มีข้อมูลชุดใหม่ถูกเพิ่มเข้ามาในฐานข้อมูลเท่านั้น

1.4.3 งานวิจัยนี้จะศึกษาภายใต้ข้อกำหนดว่าค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนดจะไม่มีการเปลี่ยนแปลง

1.5 ขั้นตอนของการศึกษา

งานวิจัยเรื่อง การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติ มีขั้นตอนของการศึกษาดังนี้

1.5.1 ศึกษาแนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้องกับงานวิจัยจากตำรา และเอกสารบทความต่างๆ

1.5.2 กำหนดหัวข้อ วัตถุประสงค์ และขอบเขตการทำงานของงานวิจัย

1.5.3 วิเคราะห์ข้อดีและข้อเสียของขั้นตอนวิธีที่เกี่ยวข้อง

1.5.4 ออกแบบขั้นตอนวิธีใหม่ที่พัฒนาประสิทธิภาพการทำงานการเพิ่มขยายกฎความสัมพันธ์

1.5.5 พัฒนาโปรแกรมการทำงานของขั้นตอนวิธีด้วยซอฟต์แวร์แมทแลป (MATLAB) รวมถึงการทดสอบการทำงานและแก้ไขข้อผิดพลาดของโปรแกรม

1.5.6 ศึกษาเทคนิคการสังเคราะห์ข้อมูล (Synthesis dataset) ที่ถูกออกแบบโดย Agrawal และคณะ [7] ซึ่งอาศัยหลักการทางสถิติที่มีการเลียนแบบพฤติกรรมการซื้อสินค้าของผู้บริโภค ทั้งนี้แนวคิดดังกล่าวถูกใช้อย่างแพร่หลายในงานวิจัยด้านการค้นหากฎความสัมพันธ์

1.5.7 พัฒนาโปรแกรมการสังเคราะห์ข้อมูลตามหลักการของ Agrawal และคณะ เพื่อสร้างชุดข้อมูลสำหรับทดสอบการทำงานของขั้นตอนวิธี

1.5.8 ทดลองการทำงานของขั้นตอนวิธีด้วยชุดข้อมูลสังเคราะห์ที่สร้างขึ้น เพื่อวัดประสิทธิภาพการทำงานของขั้นตอนวิธี

1.5.9 เรียบเรียงและจัดทำเล่มวิทยานิพนธ์

1.6 นิยามศัพท์

ในงานวิจัยฉบับนี้ มีการใช้คำศัพท์เฉพาะในการศึกษาขั้นตอนวิธีด้านการเพิ่มขยายกฎความสัมพันธ์ เพื่อให้เข้าใจตรงกัน ผู้วิจัยได้นิยามศัพท์ที่สำคัญและนิยมใช้ในงานวิจัย ดังนี้

1.6.1 ฐานข้อมูลเดิม (Original database) หมายถึง ฐานข้อมูลที่ยังไม่มีการเพิ่มข้อมูลชุดใหม่ที่เป็นข้อมูลปัจจุบันเข้าไปในฐานข้อมูล

1.6.2 ฐานข้อมูลใหม่ (Increment database) หมายถึง ฐานข้อมูลที่ประกอบด้วยข้อมูลชุดใหม่ที่จะถูกเพิ่มเข้าไปในฐานข้อมูลเดิม

1.6.3 ฐานข้อมูลปรับปรุง (Updated database) หมายถึง ฐานข้อมูลที่ได้รับการปรับปรุงแล้ว นั่นคือ ฐานข้อมูลเดิมที่มีการเพิ่มข้อมูลชุดใหม่เรียบร้อยแล้ว

1.6.4 ค่าสนับสนุนที่เป็นค่านับ (Support count) หมายถึง ค่าความถี่หรือจำนวนของรายการธุรกรรมในฐานข้อมูลที่ปรากฏไอเท็มเซตใดๆ

1.6.5 ค่าสนับสนุนขั้นต่ำ (Minimum support) หมายถึง ค่าที่ใช้ทดสอบว่าไอเท็มเซตชุดใดจะถูกกำหนดให้เป็นไอเท็มเซตที่เกิดบ่อย ค่าสนับสนุนขั้นต่ำนี้ ผู้ใช้จะเป็นผู้กำหนดตามความเหมาะสม โดยค่าสนับสนุนขั้นต่ำจะกำหนดเป็นร้อยละของจำนวนรายการธุรกรรมในฐานข้อมูลที่ปรากฏไอเท็ม A และ B $(A \cup B)$ ในอีกความหมายหนึ่ง ค่าสนับสนุนขั้นต่ำจะหมายถึงค่าความน่าจะเป็นที่ไอเท็ม A และ B จะปรากฏในฐานข้อมูลพร้อมกัน สัญลักษณ์ของค่าสนับสนุนขั้นต่ำที่ใช้โดยทั่วไปคือ $s\%$

1.6.6 ไอเท็มเซตคู่แข่ง (Candidate itemset) หมายถึงเซตของไอเท็มที่ถูกนำไปคำนวณหาค่าสนับสนุน เพื่อนำมาทดสอบว่า ไอเท็มเซตตัวใดจะถูกจัดอยู่ในกลุ่มของไอเท็มเซตที่เกิดบ่อย หรือไอเท็มเซตที่ไม่ได้เกิดบ่อย การสร้างไอเท็มเซตคู่แข่ง ตั้งแต่ $k \geq 2$ ขึ้นไป จะได้จากกระบวนการสร้างไอเท็มเซตคู่แข่งที่เรียกว่าการเชื่อมไอเท็มเซต

1.6.7 ไอเท็มเซตที่เกิดบ่อย (Frequent itemset) หมายถึง เซ็ตของไอเท็มที่น่าสนใจ โดยเซ็ตของไอเท็มใดๆ จะถูกเรียกว่าไอเท็มเซตที่เกิดบ่อยก็ต่อเมื่อ ค่าสนับสนุนของไอเท็มเซตนั้นๆ มีค่ามากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ

1.6.8 ไอเท็มเซตที่ไม่ได้เกิดบ่อย (Infrequent itemset) หมายถึง เซ็ตของไอเท็มที่ไม่น่าสนใจ โดยเซ็ตของไอเท็มใดๆ จะถูกเรียกว่าไอเท็มเซตที่ไม่ได้เกิดบ่อยก็ต่อเมื่อ ค่าสนับสนุนของไอเท็มเซตนั้นๆ มีค่าน้อยกว่าค่าสนับสนุนขั้นต่ำ

1.6.9 ค่าความเชื่อมั่นขั้นต่ำ (Minimum Confidence) หมายถึง ค่าที่ใช้ทดสอบว่ากฎความสัมพันธ์คู่ใดจะถูกกำหนดให้เป็นกฎที่น่าสนใจ (Interesting rule) หรือบางงานวิจัยจะเรียกว่ากฎที่เข้มแข็ง (Strong rule) ค่าความเชื่อมั่นขั้นต่ำนี้ ผู้ใช้จะเป็นผู้กำหนดตามความเหมาะสมและนิยมกำหนดเป็นร้อยละของจำนวนรายการธุรกรรมในฐานข้อมูลที่เมื่อปรากฏไอเท็ม A แล้วจะต้องปรากฏไอเท็ม B ในรายการธุรกรรมเดียวกันด้วย ซึ่งจะเป็นไปตามลักษณะของความน่าจะเป็นแบบมีเงื่อนไข $P(B|A)$ สัญลักษณ์ของค่าความเชื่อมั่นขั้นต่ำที่ใช้โดยทั่วไปคือ $c\%$

1.6.10 กฎความสัมพันธ์ (Association rule) หมายถึง กฎความสัมพันธ์ที่ได้จากการผ่านกระบวนการค้นหารูปแบบ (Pattern) ที่น่าสนใจในฐานข้อมูล แล้วแสดงออกมาในรูปของกฎความสัมพันธ์ ถ้า...แล้ว... ($if-then$)

1.6.11 $k-$ไอเท็มเซต ($k-itemset$) หมายถึง เซ็ตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน k ตัว โดย $k \geq 1$ เสมอ ตัวอย่าง เช่น

$k=1$ จะเรียกว่า $1-itemset$ หมายถึง เซ็ตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน 1 ตัว นิยมเขียนอยู่ในรูปของ $\{I_1, I_2, I_3, ..., I_n\}$ เมื่อ $I_1, I_2, I_3, ..., I_n$ คือไอเท็ม

k=2 จะเรียกว่า $2-itemset$ หมายถึง เซ็ตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน 2 ตัว นิยมเขียนอยู่ในรูปของ $\{\{I_1I_2\},\{I_1I_3\},\{I_1I_4\},\{I_2I_3\},\{I_2I_4\},\{I_3I_4\}\}$

k=3 จะเรียกว่า $3-itemset$ หมายถึง เซ็ตของไอเท็มที่ประกอบด้วยไอเท็มจำนวน 3 ตัว นิยมเขียนอยู่ในรูปของ $\{\{I_1I_2I_3\},\{I_1I_2I_4\},\{I_1I_3I_4\},\{I_2I_3I_4\}\}$

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในงานวิจัยและงานวิจัยที่เกี่ยวข้อง

การค้นหากฎความสัมพันธ์ เป็นเทคนิคสำคัญของการทำเหมืองข้อมูล โดยจะค้นหาสารสนเทศที่น่าสนใจของข้อมูลที่ถูกจัดเก็บไว้ในฐานข้อมูล สารสนเทศที่ได้จะอยู่ในลักษณะของรูปแบบความสัมพันธ์ระหว่างข้อมูลโดยจะแสดงออกมาในรูปของกฎความสัมพันธ์ ซึ่งจะช่วยในการตัดสินใจ การวางแผนและการบริหารได้

ในบทนี้จะกล่าวถึงแนวคิดและทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับงานวิจัยการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ประกอบด้วย

1. การค้นหากฎความสัมพันธ์
2. การเพิ่มขยายกฎความสัมพันธ์
3. ทฤษฎีการหาค่าความน่าจะเป็นแบบเบอร์นูลลี
4. ทฤษฎีการประมาณค่าทวินามด้วยการแจกแจงปกติ

โดยในแต่ละแนวคิดและทฤษฎีต่างๆ มีรายละเอียดดังนี้

2.1 การค้นหากฎความสัมพันธ์ (Association Rule Discovery)

แนวคิดเกี่ยวกับการค้นหากฎความสัมพันธ์ ถูกนำเสนอขึ้นมาครั้งแรกในปี ค.ศ. 1993 โดย Agrawal และคณะ [7] เพื่อใช้ในการค้นหารูปแบบความสัมพันธ์ระหว่าง ไอเท็ม (Item) ในชุดข้อมูลที่เป็นลักษณะชุดข้อมูลรายการธุรกรรม (Transactional dataset) ทั้งนี้ ตัวแบบทางคณิตศาสตร์ที่ได้ถูกนำเสนอเพื่อใช้ในกระบวนการค้นหากฎความสัมพันธ์ เป็นดังนี้

กำหนดให้

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

T หมายถึง รายการธุรกรรม (Transaction) หรือ รายการข้อมูล ในแต่ละรายการธุรกรรม ประกอบด้วยเซ็ตของไอเท็ม นั่นคือ $T \subseteq I$ และ รายการธุรกรรม T แต่ละรายการจะสัมพันธ์กับตัวระบุ (Identifier) ที่เรียกว่า *TID* (Transaction identifier) นอกจากนี้ แต่ละไอเท็มจะต้องมีคุณลักษณะเป็นตัวแปรทวิ (Binary variable) นั่นคือมีได้ 2 ค่า ได้แก่ ซื้อและไม่ซื้อ หรือปรากฏและไม่ปรากฏไอเท็มเซตในฐานข้อมูล เป็นต้น ทั้งนี้จำนวนรายการซื้อของแต่ละไอเท็มจะไม่ถูกนำมาพิจารณาเช่น กำหนดให้ไอเท็ม I_1 แทน ขนมปัง ในแต่ละรายการที่มีการซื้อขนมปัง จะนับว่ามีการซื้อขนมปังจำนวน 1 ครั้ง โดยไม่สนใจว่าในรายการนั้นจะมีการซื้อขนมปังจำนวนมากกว่า 1 ถุงหรือไม่

D หมายถึง เซ็ตของรายการธุรกรรมในฐานข้อมูล

สมมติกำหนดให้ไอเท็ม X เป็นไอเท็มที่เกิดขึ้นในรายการธุรกรรมหนึ่งๆ นั่นคือ $X \subseteq T$ ลักษณะของกฎความสัมพันธ์จะเป็นกฎความสัมพันธ์แบบ *IF...THEN...* ซึ่งจะแสดงในรูปแบบของ $X \Rightarrow Y$ โดย $X \subset I, Y \subset I$ และ $X \cap Y = \phi$

ไอเท็มเซตใดๆ ที่มีโอกาสถูกนำไปใช้สร้างกฎความสัมพันธ์ $X \Rightarrow Y$ จะต้องมีค่าสนับสนุน (Support) มากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ (Minimum support) หรือ min_sup ที่ผู้ใช้กำหนด

ส่วนกฎความสัมพันธ์ $X \Rightarrow Y$ ใดๆ จะถูกเรียกว่าเป็นกฎที่น่าสนใจ จะต้องมีค่าความเชื่อมั่น (Confidence) มากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำ (Minimum confidence) หรือ min_*conf* ที่ผู้ใช้กำหนด

ทั้งนี้แนวคิดเกี่ยวกับค่าสนับสนุนขั้นต่ำและค่าความเชื่อมั่นขั้นต่ำสำหรับการค้นหากฎความสัมพันธ์ สามารถแสดงให้อยู่ในรูปแบบของความน่าจะเป็นได้ดังนี้

$$Support(X \Rightarrow Y) = P(X \cup Y) \tag{2.1}$$

$$Confidence(X \Rightarrow Y) = P(X \mid Y) \tag{2.2}$$

ตัวอย่าง สมมติให้ฐานข้อมูลหนึ่งมีรายการธุรกรรมจำนวน 10 รายการธุรกรรม พบว่ามีไอเท็ม A และ B ปรากฏขึ้นพร้อมกันในรายการธุรกรรมเดียวกันจำนวน 6 รายการธุรกรรม ซึ่งในงานวิจัยจะเรียกว่า ไอเท็มเซต $\{A, B\}$ มีค่าสนับสนุนเท่ากับ 6 และสมมติให้ค่าสนับสนุนขั้นต่ำ $\text{min\_sup} = 40\%$ ซึ่งหมายถึงร้อยละ 40 ของรายการธุรกรรมทั้งหมดในฐานข้อมูลที่จะปรากฏไอเท็มเซตนั้นๆ คู่กัน

จากตัวอย่าง ฐานข้อมูลมีจำนวน 10 รายการธุรกรรม ดังนั้นค่า min_sup จะเท่ากับ $40\% \times 10 = 4$ รายการธุรกรรม เมื่อไอเท็ม A และ B ปรากฏขึ้นในรายการธุรกรรมเดียวกันจำนวน 6 รายการธุรกรรม ซึ่งมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ นั่นคือ $6 > \text{min\_sup}$ ดังนั้น ไอเท็ม $\{A, B\}$ จึงเรียกว่าไอเท็มเซตที่เกิดบ่อย ซึ่งจะถูกนำไปสร้างเป็นกฎความสัมพันธ์ต่อไป

จากที่ได้กล่าวข้างต้น การหากฎความสัมพันธ์จะประกอบด้วย 2 ขั้นตอนหลัก คือการหาไอเท็มเซตที่เกิดบ่อยและการสร้างกฎความสัมพันธ์ ซึ่งกระบวนการสร้างกฎความสัมพันธ์นั้นจะสามารถสร้างกฎความสัมพันธ์ได้ก็ต่อเมื่อกระบวนการคำนวณหาไอเท็มเซตที่เกิดบ่อยกระทำเสร็จสิ้นแล้ว สำหรับงานวิจัยทางด้านการค้นหากฎความสัมพันธ์ ขั้นตอนวิธีที่ได้รับความนิยมในการนำมาใช้ในการหาไอเท็มเซตที่เกิดบ่อยได้แก่ ขั้นตอนวิธีอะพริโอริ ซึ่งจะอธิบายในหัวข้อถัดไป

2.1.1 ขั้นตอนวิธีอะพริโอริ (Apriori Algorithm) [1]

ขั้นตอนวิธีอะพริโอริ เป็นขั้นตอนวิธีที่ค่อนข้างมีบทบาทและเป็นที่นิยมในงานวิจัยด้านการค้นหากฎความสัมพันธ์ ขั้นตอนวิธีนี้จะมีการคำนวณหาไอเท็มเซตที่เกิดบ่อย แสดงดังรูปที่ 2.1

```
1)   L_1 = {L arg e1 − itemset};
2)   for ( k = 2; L_{k-1} ≠ φ; 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;
```

รูปที่ 2.1 ขั้นตอนการหาไอเท็มเซตที่เกิดบ่อยของขั้นตอนวิธีอะพริโอริ

การทำงานของอะพริโอริ จะเริ่มต้นจากการสแกนแต่ละรายการธุรกรรมในฐานข้อมูล โดยมีการสแกนวนซ้ำหลายครั้ง ไอเท็มเซตที่เกิดบ่อย $k-$ไอเท็มเซต (L_k) ที่คำนวณได้แต่ละรอบจะคำนวณจาก ไอเท็มเซตคู่แข่ง $k-1$ ไอเท็มเซต (C_{k-1}) ซึ่งเป็นลักษณะการทำงานของ Level-wise-step ในการคำนวณหาไอเท็มเซตที่เกิดบ่อยของขั้นตอนวิธีนี้จะแบ่งการทำงานออกเป็น 2 ขั้นตอนหลัก ได้แก่

1. ขั้นตอนการเชื่อมไอเท็มเซต (Join step)

เป็นขั้นตอนการนำ L_{k-1} เมื่อ $k \geq 2$ มาทำการเชื่อมไอเท็มเซต เพื่อสร้างไอเท็มเซตคู่แข่ง C_k ซึ่งจะใช้ในการคำนวณหา L_k ในรอบถัดไป เช่น C_2 ได้มาจากการเชื่อมไอเท็มเซตระหว่าง $L_1 * L_1$ จากนั้นจึงนำ C_2 ที่ได้ไปคำนวณหา L_2 โดยการนำค่าสนับสนุนของ C_2 ไปเปรียบเทียบกับค่า min_sup เมื่อได้ L_2 ก็จะทำการเชื่อมไอเท็มเซตระหว่าง $L_2 * L_2$ ให้ได้ C_3 เพื่อนำไปคำนวณหา L_3 เป็นลำดับถัดไป และจะทำเช่นนี้ไปเรื่อยๆ จนกระทั่งไม่มี C_k ที่จะนำไปหา L_k แล้ว นั่นคือ $C_k = \phi$ กระบวนการเชื่อมไอเท็มเซต แสดงดังรูปที่ 2.2

```
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 ขั้นตอนการเชื่อมไอเท็มเซต (Join step) ของขั้นตอนวิธีอะพริโอริ

2. ขั้นตอนการตัดไอเท็มเซต (Prune step)

เป็นขั้นตอนสำหรับการตัดไอเท็มเซต C_k ที่ไม่มีโอกาสเป็น L_k ได้ โดยพิจารณาเปรียบเทียบระหว่างค่าสนับสนุนของไอเท็มเซตและค่า min_sup นั่นคือ C_k ใดๆ ที่ $X.\text{sup}port < \text{min\_sup}$ จะถูกตัดออกไปและไอเท็มเซตที่เหลือ ซึ่ง $X.\text{sup}port \geq \text{min\_sup}$ จะเรียกว่าเป็น L_k ขั้นตอนของการตัดไอเท็มเซต แสดงดังรูปที่ 2.3 นอกจากนี้ คุณสมบัติของไอเท็มเซตได้ถูกกำหนดไว้ดังนี้ "ถ้าสับเซตของ $k-1$ ของไอเท็มเซตใดๆ ใน C_k ที่ไม่ได้เป็นสมาชิกของ L_{k-1} แล้ว ไอเท็มเซตนั้นๆ จะไม่สามารถเป็น L_k ได้ ดังนั้นจึงสามารถตัดไอเท็มเซตนั้นๆ ออกจาก C_k ได้"

```
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 ขั้นตอนการตัดไอเท็มเซต (Prune step) ของขั้นตอนวิธีอะพริโอริ

เมื่อเสร็จสิ้นกระบวนการหาไอเท็มเซตที่เกิดบ่อยแล้ว ขั้นตอนถัดมาคือการหากฎความสัมพันธ์ที่เป็นไปตามเงื่อนไขของค่าสนับสนุนขั้นต่ำ min_sup และค่าความเชื่อมั่นขั้นต่ำ min_*conf* ที่ผู้ใช้กำหนดไว้ ซึ่งค่าทั้งสองดังกล่าวจะมีช่วงระหว่าง 0 – 100% หากกฎความสัมพันธ์ใดๆ ที่เป็นไปตามค่า min_sup และ min_*conf* ดังกล่าว จะจัดว่ากฎความสัมพันธ์นั้นเป็นกฎที่น่าสนใจหรือกฎที่เข้มแข็ง (Strong rule) ในทางกลับกัน หากกฎความสัมพันธ์ใดที่ไม่เป็นตามค่า min_sup และ min_*conf* ดังกล่าว จะจัดว่ากฎนั้นเป็นกฎที่ไม่น่าสนใจหรือเป็นกฎที่อ่อนแอ (Weak rule)

ข้อดีของขั้นตอนวิธีอะพริโอริ

มีการออกแบบเทคนิคในการลดจำนวนไอเท็มเซตคู่แข่งที่ได้จากกระบวนการเชื่อมไอเท็มเซต ด้วยหลักการตัดไอเท็มเซตที่ไม่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยออกไป โดยใช้ความรู้เดิมในรอบก่อนหน้า $(k-1)$ เข้ามาช่วย

ข้อเสียของขั้นตอนวิธีอะพริโอริ

1. จำเป็นต้องมีการสแกนฐานข้อมูลหลายครั้ง และมีการสร้างไอเท็มเซตคู่แข่งจำนวนมากในการคำนวณแต่ละรอบ ทำให้ใช้เวลาในการประมวลผลค่อนข้างนาน

2. หากมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูล จะต้องหาไอเท็มเซตที่เกิดบ่อยใหม่ด้วยการสแกนฐานข้อมูลทั้งหมด ซึ่งทำให้สูญเสียเวลาในการประมวลผลโดยไม่จำเป็น

2.2 การเพิ่มขยายกฎความสัมพันธ์ (Incremental Association Rule Discovery)

ธรรมชาติของข้อมูลนั้น จะมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิมตลอดเวลา ซึ่งลักษณะของฐานข้อมูลดังกล่าวจะเรียกว่าฐานข้อมูลเชิงพลวัต (Dynamic database) จึงทำให้เกิดกฎความสัมพันธ์ใหม่ที่น่าสนใจเกิดขึ้น ขณะเดียวกันกฎความสัมพันธ์เดิมอาจจะกลายเป็นกฎที่ไม่น่าสนใจอีกต่อไป Tsai และคณะ [8] ได้อธิบายเหตุการณ์ที่มีโอกาสจะเกิดขึ้นเมื่อมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิม ประกอบด้วย 4 เหตุการณ์ ดังนี้

1. ไอเท็มเซต ที่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมจะยังคงเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่เช่นเดิม

2. ไอเท็มเซต ที่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมไม่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่

3. ไอเท็มเซต ที่ไม่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมเปลี่ยนเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่

4. ไอเท็มเซต ที่ไม่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมไม่สามารถเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่เช่นเดิม

การรักษาไว้ซึ่งความถูกต้องของกฎความสัมพันธ์ (Maintenance of association rules) จึงเป็นปัญหาสำคัญที่มีผู้วิจัยหลายรายพยายามศึกษาหาแนวทางแก้ไขปัญหาดังกล่าว ทั้งนี้ รัชดาภรณ์ และวรพจน์ [6] ได้มีการจำแนกงานวิจัยเกี่ยวกับการเพิ่มขยายการค้นหากฎความสัมพันธ์ตามสมมติฐานไว้ 2 ประเด็น ดังนี้

1. กฎความสัมพันธ์ที่หาได้จะมีการเปลี่ยนแปลงเมื่อเวลาเปลี่ยนไป (Association rules are not stable over time)

งานวิจัยในกลุ่มนี้มีข้อสมมติฐานว่าข้อมูลเก่า (Old dataset) และข้อมูลใหม่ (New dataset) มีน้ำหนักความสำคัญไม่เท่ากัน นั่นคือ จะกำหนดให้น้ำหนักข้อมูลชุดเก่าน้อยกว่าข้อมูลชุดใหม่ เนื่องจากในบางครั้ง ข้อมูลใหม่ๆ จะแสดงให้เห็นแนวโน้มความเป็นไปในอนาคตที่น่าสนใจมากกว่าข้อมูลชุดเก่า ตัวอย่างงานวิจัยในกลุ่มนี้ได้แก่ เทคนิคการถ่วงน้ำหนัก (Weighting technique) [9] และ ฟังก์ชันอิทธิพลของเวลา (Time Influence function) [10]

2. กฎความสัมพันธ์ที่หาได้จะไม่มีการเปลี่ยนแปลงเมื่อเวลาเปลี่ยนไป (Association rules are stable over time)

งานวิจัยในกลุ่มนี้มีข้อสมมติฐานว่าข้อมูลเก่า (Old dataset) และข้อมูลใหม่ (New dataset) มีน้ำหนักความสำคัญเท่ากัน ผลลัพธ์ของกฎความสัมพันธ์ที่ได้จะเหมือนการประมวลผลด้วยขั้นตอนวิธี อะพริโอริ ซึ่งจะเป็นการประมวลผลข้อมูลทั้งเก่าและใหม่รวมกันทั้งหมด ทั้งนี้ รัชดาภรณ์และวรพจน์ [5] ได้จำแนกเทคนิคที่ใช้ในงานวิจัยกลุ่มนี้ออกเป็น 3 กลุ่ม ดังนี้

2.1 กลุ่มที่ใช้หลักการของอะพริโอริเป็นฐาน (Apriori-based)

งานวิจัยกลุ่มนี้ เป็นงานวิจัยที่ใช้หลักการของขั้นตอนวิธีอะพริโอริ เป็นฐาน เช่น ขั้นตอนวิธีเอฟยูพี [2], เอฟยูพี2 [11], เนกาทีฟบอร์เดอร์ [3,4], พรีลาร์จ [5], ขั้นตอนวิธีการประมาณค่าไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย [12] และ ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น [6]

ขั้นตอนวิธีเอฟยูพี อาศัยหลักการของอะพริโอริในการทำงาน และเพิ่มการใช้ประโยชน์จากผลการประมวลผลในรอบก่อนหน้าเพื่อลดจำนวนไอเท็มเซตคู่แข่งที่จะใช้สแกนในฐานข้อมูลเดิม แต่ เอฟยูพี จะใช้ได้ในกรณีที่มีการเพิ่มฐานข้อมูลใหม่เข้าไปเท่านั้น หลังจากนั้นได้มีการพัฒนา เอฟยูพี2 เพื่อปรับปรุงประสิทธิภาพของ เอฟยูพี ให้สามารถหาไอเท็มเซตที่เกิดบ่อยได้ทั้งกรณีที่มีการเพิ่ม ลบ และการแก้ไขข้อมูล

อย่างไรก็ตาม ทั้ง เอฟยูพี และ เอฟยูพี2 ยังมีความจำเป็นต้องสแกนฐานข้อมูลเดิมในทุกๆ รอบ ต่อมา เนกาทีฟบอร์เดอร์ ได้ถูกเสนอเพื่อลดจำนวนรอบของการสแกนฐานข้อมูล กระบวนการทำงานของ เนกาทีฟบอร์เดอร์ มีการทำงานเช่นเดียวกับ อะพริโอริ และมีการเก็บไอเท็มเซตที่เกิดบ่อยเช่นเดียวกับ เอฟยูพี และ เอฟยูพี2 เพื่อลดจำนวนรอบของการสแกนฐานข้อมูล เนกาทีฟบอร์เดอร์ ได้มีการจัดเก็บไอเท็มเซตที่ไม่ได้เกิดบ่อย ซึ่งเรียกว่า บอร์เดอร์เซต (Border set) เพื่อให้มีข้อมูลเพียงพอที่ใช้ในการประมวลผลเมื่อมีการเพิ่มข้อมูลใหม่เข้ามา อย่างไรก็ตามแม้ว่าขั้นตอนวิธีเนกาทีฟบอร์เดอร์ จะสามารถลดจำนวนรอบของการสแกนฐานข้อมูลเดิมได้เป็นอย่างดี แต่ต้องจัดเก็บบอร์เดอร์เซตจำนวนมหาศาล และมีความเสี่ยงที่จะหาไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรุงได้ไม่ครบกรณีที่มีไอเท็มตัวใหม่เกิดขึ้น เนื่องจาก เนกาทีฟบอร์เดอร์ ไม่ได้ออกแบบขั้นตอนวิธีให้รองรับการเกิดไอเท็มเซตตัวใหม่

เพื่อแก้ไขปัญหาการเก็บบอร์เดอร์เซตที่มากเกินไป ขั้นตอนวิธี พรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจึงเสนอแนวคิดเพื่อลดจำนวนบอร์เดอร์ไอเท็มเซต โดย พรีลาร์จ จะกำหนดให้มีพารามิเตอร์เพิ่มอีก 2 ตัว คือ ค่าสนับสนุนที่เป็นขอบเขตบน (Upper support) และค่าสนับสนุนที่เป็นขอบเขตล่าง (Lower support) เพื่อกรองไอเท็มที่เซ็ตที่ไม่ผ่านค่าพารามิเตอร์นี้ออกไป ทำให้จำนวนบอร์เดอร์เซตลดลง ส่วนขั้นตอนวิธีการ

ประมาณค่าไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยและ ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น ได้นำเสนอเทคนิคการประมาณค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยสำหรับเก็บไว้เพื่อนำไปประมวลผลและสามารถลดจำนวนครั้งของการสแกนฐานข้อมูลเดิมให้เหลือเพียงครั้งเดียว โดยได้นิยามไอเท็มเซตที่เป็นบอร์เดอร์เซตใหม่ว่าไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย (Promising frequent itemset) [12] หรือ ไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง (Expected frequent itemset) [6] ซึ่งหมายถึง ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่มีโอกาสที่จะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งคำนวณได้จากความน่าจะเป็น

2.2 กลุ่มที่ใช้หลักการแบ่งข้อมูลออกเป็นส่วนๆ (Partition-based)

งานวิจัยกลุ่มนี้ เป็นกลุ่มที่นำเสนอเทคนิคการเพิ่มขยายกฎความสัมพันธ์โดยการแบ่งข้อมูลที่มีอยู่ในฐานข้อมูลออกเป็นส่วนๆ ที่เรียกว่าพาร์ทิชั่น (Partition) เนื่องจากงานวิจัยกลุ่มอะพริโอริ ประสบกับปัญหาหลัก 2 ประเด็น คือ 1) ปัญหาการสร้างไอเท็มเซตคู่แข่งจำนวนมาก และ 2) ปัญหาการสแกนฐานข้อมูลเดิมหลายๆ รอบ เอสดับบลิวเอฟ (Sliding windows filtering) [13] ซึ่งเป็นขั้นตอนวิธีที่ได้รับความนิยมในกลุ่มที่ใช้หลักการแบ่งข้อมูลออกเป็นส่วนๆ ได้เสนอแนวคิดเพื่อแก้ไขปัญหาดังกล่าวด้วยการแบ่งฐานข้อมูลออกเป็นส่วนๆ แล้วทำการประมวลผลแต่ละส่วน ซึ่ง เอสดับบลิวเอฟ จะมีขั้นตอนการทำงาน 2 ส่วน ได้แก่ 1) การแบ่งส่วนข้อมูลออกเป็นส่วน และ 2) การประมวลผลในแต่ละส่วน โดยแบ่งการประมวลผลเป็นการประมวลผลให้แก่ ส่วนของข้อมูลเก่า และส่วนของข้อมูลใหม่ การทำงานทั้งสองส่วนนี้ จะเก็บไอเท็มเซตคู่แข่งไว้เพื่อคำนวณหาไอเท็มเซตที่เกิดบ่อยเป็นลำดับถัดไป จุดเด่นของ เอสดับบลิวเอฟ คือ จะเริ่มคำนวณหาไอเท็มเซตคู่แข่งที่ $(k \geq 2)$ โดยอนุมานให้ 1 – ไอเท็มเซต ทุกตัวเป็นไอเท็มเซตที่เกิดบ่อย โดยไม่สนใจว่าค่าสนับสนุนของไอเท็มเซตแต่ละตัวนั้นมีค่ามากกว่าค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนดหรือไม่ งานวิจัยถัดมาคือ เอฟไอเอสดับบลิวเอฟ และ ซีไอเอสดับบลิวเอฟ [14] ได้ทำการวิจัยต่อเนื่องมาโดยอาศัยหลักการจาก เอสดับบลิวเอฟ แต่จะมีการเก็บทั้งค่าไอเท็มเซตคู่แข่งและไอเท็มเซตที่เกิดบ่อยเพื่อลดจำนวนไอเท็มเซตคู่แข่งที่ถูกสร้างขึ้นและเพิ่มประสิทธิภาพให้แก่ เอสดับบลิวเอฟ

2.3 กลุ่มที่ใช้หลักการสร้างรูปแบบการเจริญเติบโต (Pattern-Growth)

งานวิจัยกลุ่มนี้ เป็นกลุ่มที่นำเสนอเทคนิคการเพิ่มขยายการค้นหากฎความสัมพันธ์โดยใช้หลักการจาก เอฟพีทรี (FP-tree) เพื่อลดปัญหาการสร้างไอเท็มเซตคู่แข่งจำนวนมาก ซึ่งเป็นปัญหาที่งานวิจัยกลุ่มที่ใช้หลักการของอะพริโอริเป็นฐานและกลุ่มที่ใช้หลักการแบ่งข้อมูลออกเป็นส่วนประสบ แต่เนื่องจากเอฟพีทรี ไม่สามารถประยุกต์ใช้งานได้โดยตรงกับ

ปัญหาด้านการเพิ่มขยายกฎความสัมพันธ์ จึงมีผู้นำเสนองานวิจัยที่สนับสนุนการแก้ปัญหาดังกล่าว เช่น ดีบีทรี และ พีโอทีเอฟพีทรี [15]

กระบวนการทำงานของขั้นตอนวิธีดังกล่าวข้างต้น รายละเอียดดังนี้

2.2.1 ขั้นตอนวิธี เอฟยูพี [2]

ขั้นตอนวิธี เอฟยูพี (Fast UPdated algorithm) เป็นงานวิจัยแรกที่นำเสนอเทคนิคการเพิ่มขยายกฎความสัมพันธ์เพื่อรักษากฎความสัมพันธ์ให้เป็นปัจจุบันเมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิม การทำงานของ เอฟยูพี อาศัยหลักการทำงานเช่นเดียวกับ อะพริโอริ ซึ่งจะมีการทำงานหลายรอบ โดยรอบแรกจะเริ่มตั้งแต่ 1 – ไอเท็มเซต ไปจนถึง k – ไอเท็มเซต ทั้งนี้ ขั้นตอนวิธีนี้จะทำงานภายใต้การอนุมานว่า ค่า min_sup และค่า min_*conf* คงที่

ความหมายและสัญลักษณะต่างๆ ที่ใช้ในขั้นตอนวิธี เอฟยูพี มีดังนี้

DB หมายถึง ฐานข้อมูลเดิม

db หมายถึง ฐานข้อมูลใหม่ที่ถูกเพิ่มเข้ามา

D หมายถึง จำนวนรายการธุรกรรมที่มีอยู่ในฐานข้อมูลเดิม

d หมายถึง จำนวนรายการธุรกรรมที่มีอยู่ในฐานข้อมูลใหม่

s หมายถึง ค่าสนับสนุนขั้นต่ำ (min_sup)

L_k หมายถึง ไอเท็มเซตที่เกิดบ่อยขนาด k – ไอเท็มเซต ในฐานข้อมูลเดิม เมื่อ $k = 1,2,3,...$

L'_k หมายถึง ไอเท็มเซตที่เกิดบ่อยขนาด k – ไอเท็มเซต ในฐานข้อมูลปรับปรุง $(DB \cup db)$ เมื่อ $k = 1,2,3,...$

$X.\text{sup}\,port_D$ หมายถึง ค่าสนับสนุนของไอเท็ม X ในฐานข้อมูลเดิม

$X.\text{sup}\,port_d$ หมายถึง ค่าสนับสนุนของไอเท็ม X ในฐานข้อมูลใหม่

$X.\text{sup}\,port_{UD}$ หมายถึง ค่าสนับสนุนของไอเท็ม X ในฐานข้อมูลปรับปรุง

ขั้นตอนวิธี เอฟยูพี จะแบ่งการทำงานออกเป็น 2 ส่วนหลัก คือแรกและที่สองเป็นต้นไป รายละเอียดขั้นตอนการทำงาน เอฟยูพี ทั้งสองส่วน อธิบายดังนี้

2.2.1.1 กระบวนการทำงานรอบแรก

การทำงานในส่วนนี้ จะเป็นการตัด (Prune) ไอเท็มที่เป็นไอเท็มที่แพ้ หรือไอเท็มเซตที่ไม่สามารถเป็นไอเท็มเซตที่เกิดบ่อย และหาไอเท็มที่เป็นไอเท็มที่ชนะ คือไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อย

■ สแกนฐานข้อมูลใหม่ *db* เพื่อทำการปรับปรุงค่าสนับสนุน (Support) ของไอเท็ม และตัดไอเท็มที่เป็นแพ้ ออกไป รวมถึงหาไอเท็มที่ชนะ มีหลักการพิจารณาดังนี้

◆ กรณี $X \in L_1$ ให้นำค่าสนับสนุนของไอเท็ม X ใน DB และ db มารวมกัน จะได้ $X.\sup port_{UD} = X.\sup port_D + X.\sup port_d$ จากนั้นทำการตรวจสอบค่าสนับสนุนที่ได้ โดย

- ถ้า $X.\sup port_{UD} \geq s \times (D+d)$ แสดงว่าไอเท็ม X นั้นๆ เป็นไอเท็มที่ชนะและให้ $X \cup L_1'$

- ถ้า $X.\sup port_{UD} < s \times (D+d)$ แสดงว่าไอเท็ม X นั้นๆ เป็นไอเท็มที่แพ้และให้ตัดไอเท็ม X นั้นทิ้งไป

◆ กรณี $X \notin L_1$ จะมีการพิจารณา 2 ประเด็น ดังนี้

- ตัดไอเท็มที่ไม่มีโอกาสเป็น L_1' ได้ โดยพิจารณาจาก $X \notin L_1$ และ $X.\sup port_{UD} < s \times (D+d)$ แสดงว่าไอเท็ม X นั้นเป็นไอเท็มที่แพ้ จะทำการตัดไอเท็ม X นั้นทิ้งไป ซึ่งจะช่วยลดการสแกนไอเท็มใน DB

- ตรวจสอบไอเท็มที่มีโอกาสเป็น L_1' ได้ โดยพิจารณาจาก $X \notin L_1$ และ $X.\sup port_{UD} \geq s \times (D+d)$ ให้นำไอเท็ม X ดังกล่าว ไปสแกนใน DB จากนั้นจึงนำค่า $X.\sup port_{UD}$ ที่ได้ มาตรวจสอบว่าเป็นไอเท็มที่แพ้หรือไอเท็มที่ชนะ โดย

✡ ถ้า $X.\sup port_{UD} \geq s \times (D+d)$ แสดงว่าไอเท็ม X นั้นๆ เป็น ไอเท็มที่ชนะและให้ $X \in L_1'$

✡ ถ้า $X.\sup port_{UD} < s \times (D+d)$ แสดงว่าไอเท็ม X นั้นๆ เป็นไอเท็มที่แพ้ จะทำการตัดไอเท็ม X นั้นทิ้งไป

เมื่อเสร็จสิ้นกระบวนการทำงานในรอบนี้จะได้ไอเท็มเซตที่เกิดบ่อยขนาด $1-$ ไอเท็มเซต (L_1') ในฐานข้อมูลที่ปรับปรุงแล้ว (UD) ทั้งนี้กระบวนการทำงานในรอบแรกเพื่อหา L_1' ของ เอฟยูพี แสดงดังรูปที่ 2.4 และขั้นตอนวิธีแสดงการทำงานของรอบนี้ แสดงดังรูปที่ 2.5

2.2.1.2 กระบวนการทำงานรอบที่สองเป็นต้นไป

การทำงานของขั้นตอนวิธี เอฟยูพี ในส่วนนี้ จะเป็นหา L_k' เมื่อ $k \geq 2$ ซึ่งจะมีการทำงานบางส่วนที่มีหลักการทำงานคล้ายการทำงานในรอบแรกนั่นคือ การหาไอเท็มที่แพ้ที่ไม่สามารถเป็น L_k' เพื่อลดการสแกน C_k เมื่อ $k \geq 2$ ใน db โดยอาศัยแนวคิดที่ว่า "ถ้าไอเท็ม X ใดๆ ที่เป็นไอเท็มที่แพ้ในการประมวลผลรอบที่ $k-1$ เมื่อ $k \geq 2$ แล้ว ไอเท็มเซตใดๆ ของ L_k ในฐานข้อมูลเดิมที่มีไอเท็ม X ดังกล่าวเป็นสับเซตอยู่ จะไม่สามารถเป็นไอเท็มที่ชนะ ในรอบที่ k" จากแนวคิดดังกล่าว ในการทำงานรอบที่สองเป็นต้นไปแสดงดังรูปที่ 2.6 มีการทำงานดังนี้

(1) หาไอเท็มเซตที่เป็นสมาชิกของ L_2 และ $L_2^{'}$ นั่นคือ หาไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อย 2 – ไอเท็มเซตทั้งในฐานข้อมูลเดิมและฐานข้อมูลที่ปรับปรุงแล้ว

ขั้นตอนนี้จะทำการตัดไอเท็มที่แพ้ ซึ่งเป็นไอเท็มที่ไม่สามารถเป็น $L_2^{'}$ เพื่อลดจำนวนไอเท็มที่จะสแกนใน db โดยพิจารณาจาก $Y = L_1 - L_1^{'}$ เมื่อ Y คือไอเท็มใดๆ ที่เป็นสมาชิกของ L_1 แต่ไม่เป็นสมาชิกของ $L_1^{'}$ นั่นคือ เมื่อ $X \in L_2$ ที่มีไอเท็ม Y เป็นสับเซต จะไม่สามารถเป็น $L_2^{'}$ ได้ ดังนั้น ไอเท็ม X ดังกล่าวจะถูกตัดทิ้งไป

ตัวอย่าง

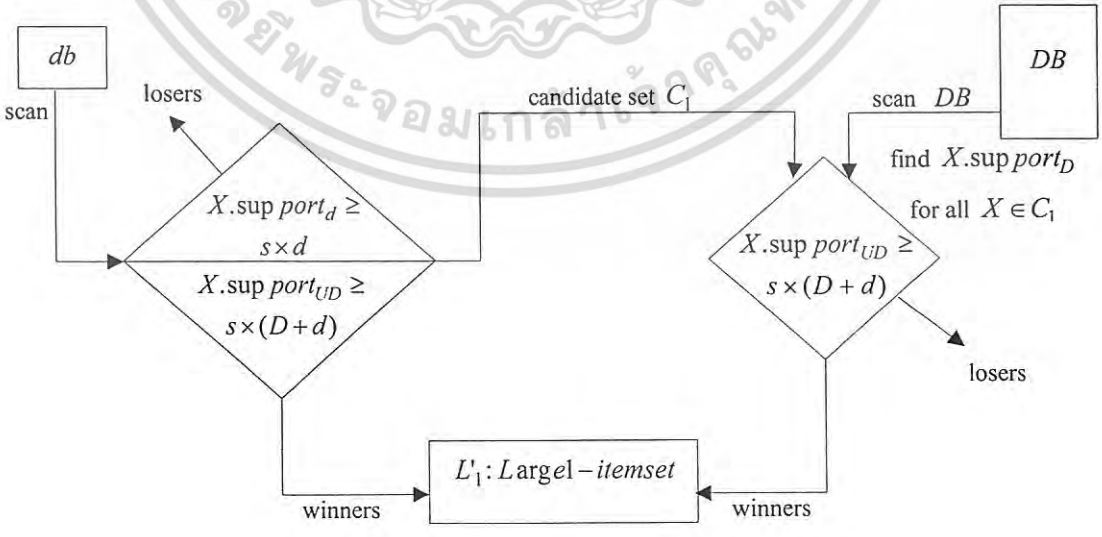
$L_1 = \{A, B, C\}$ $L_1^{'} = \{A, C, D\}$ $L_2 = \{AB, AC\}$

ดังนั้น จะได้ $L_1 - L_1^{'} = \{B\}$

จากตัวอย่างจะเห็นได้ว่า $\{B\}$ เป็นสมาชิกของ L_1 แต่ไม่เป็นสมาชิกใน $L_1^{'}$ ดังนั้นไอเท็มเซตใดๆ ใน L_2 ที่มี $\{B\}$ เป็นสมาชิก ไอเท็มเซตนั้นๆ จะถูกเรียกว่าเป็นไอเท็มที่แพ้ และจะถูกตัดออกไป โดยไม่ต้องนำไปสแกนใน db ซึ่งจากตัวอย่าง ไอเท็มเซตใน L_2 ที่มี $\{B\}$ เป็นสับเซตคือ $\{AB\}$ ดังนั้น $\{AB\}$ จะถูกตัดออกจาก L_2 และ L_2 จะเหลือสมาชิกอยู่คือ $\{AC\}$ ซึ่ง $\{AC\}$ จะถูกนำไปสแกนใน db เพื่อปรับปรุงค่าสนับสนุน

ในการสแกนสมาชิกตัวที่เหลือใน L_2 และปรับปรุงค่าสนับสนุนเรียบร้อยแล้ว หากพบว่า

- $X.\sup port_{UD} \geq s \times (D + d)$ แล้ว ไอเท็ม X นั้นๆ จะเรียกว่าไอเท็มที่ชนะ แล้วจะถูกนำไปเก็บไว้ใน $L_2^{'}$
- $X.\sup port_{UD} < s \times (D + d)$ แล้ว ไอเท็ม X นั้นๆ จะเรียกว่าไอเท็มที่แพ้ และจะถูกตัดทิ้งไป



ภาพที่ 2.4 ผังงานการทำงานรอบแรกของขั้นตอนวิธี เอฟยูพี

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 $k-itemset$ in DB, where $k = 1,...,r$;
(3) db : an increment database (with its size equal to d);
(4) s : the minimum support factor.

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 $DB \cup db$ */

$W = L_1$; $C = \phi$; $L'_1 = \phi$; $P = \phi$;

/* 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** $X.\text{sup}\, port_d ++$;

else {

if $X \notin C$

then { $C = C \cup \{X\}$; $X.\text{sup}\, port_d = 0$; }

/* initial the support count and add X into C */

$X.\text{sup}\, port_d ++$; }

};

for all $X \in W$ **do** /* put winners into L'_1 */

if $X.\text{sup}\, port_{UD} \geq s \times (D+d)$

then $L'_1 = L'_1 \cup \{X\}$;

for all $X \in C$ **do** /* prune candidate sets in C */

if $X.\text{sup}\, port_d < s \times d$

then $\{C = C - \{X\}; P = P - \{X\};\}$

/* 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** $X.\text{sup}\, port_D ++$;

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 $X.\text{sup}\, port_{UD} \geq s \times (D+d)$

then $L'_1 = L'_1 \cup \{X\}$;

return L'_1 /* end of the 1st iteration */

ภาพที่ 2.5 ขั้นตอนการปรับปรุงไอเท็มเซตในรอบแรกของขั้นตอนวิธี เอฟยูพี

The $k-th$ iteration: /* for $k=2$ or larger, repeat this program fragment to find L'_1, the set of all large $k-itemsets$ in the updated database, until either L'_k returned is empty or $db=\phi$ */

$W = L_k$; $L'_k = \phi$; /* 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 $(k-1)-itemset$ $Y \subseteq L_{k-1}-L'_{k-1}$ **do**
    if $Y \in X$ then $\{W = W - \{X\}; break;\}$
for all $T \in db$ **do** { /* scan db */
  for all $X \in subset(W,T)$ **do** $X.\text{sup}\ port_d++$;
  /* $subset(W,T)$ returns all the sets in W contained in T[2] */
  for all $X \in subset(C,T)$ **do** $X.\text{sup}\ port_d++$;
  /* 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 $X.\text{sup}\ port_{UD} \geq s \times (D+d)$
  then $L'_k = L'_k \cup \{X\}$;
for all $X \in C$ **do** /* prune candidate sets in C */
  if $X.\text{sup}\ port_d \geq s \times d$ **then** $C = C - \{X\}$;
for all $T \in DB$ **do** { /* scan DB */
  for all $X \in subset(C,T)$ **do** $X.\text{sup}\ port_D++$;
  Reduce_ $DB(T)$;}
  /* Some items in transactions in DB can be removed, discuss in next section */
for all $X \in C$ **do** /* put the winners from C into L'_k */
  if $X.\text{sup}\ port_{UD} \geq s \times (D+d)$
  then $L'_k = L'_k \cup \{X\}$;
return L'_k /* end of the $k-th$ iteration */

ภาพที่ 2.6 ขั้นตอนการปรับปรุงไอเท็มเซต ในรอบที่สองเป็นต้นไปของขั้นตอนวิธี เอฟยูพี

(2) หาไอเท็มเซตที่เกิดบ่อย 2 – ไอเท็มเซต ใหม่ที่เกิดขึ้นในฐานข้อมูลปรับปรุง (L_2')

ในขั้นตอนนี้จะเริ่มคำนวณ C_2 ด้วยการเชื่อมไอเท็มเซต ระหว่าง $L_1' * L_1'$ เพื่อนำไปสแกนใน db และหา L_2' โดยพิจารณาดังนี้

◆ กรณี $X \in C_2$ และ $X \in L_2'$

ไม่ต้องนำไอเท็ม X ดังกล่าวไปสแกนใน db เนื่องจาก X เป็นสมาชิกของ L_2' แล้ว ซึ่งคำนวณได้จากขั้นตอนที่ (1)

◆ กรณี $X \in C_2$ และ $X \notin L_2'$

ให้นำไอเท็ม X ดังกล่าวไปสแกนใน db แล้วตรวจสอบ

- ถ้า $X.\sup port_{UD} \geq s \times d$ ให้นำไอเท็ม X ดังกล่าวไปสแกนใน DB แล้วปรับปรุงค่าสนับสนุน จากนั้นหากตรวจสอบแล้วพบว่า $X.\sup port_{UD} \geq s \times (D + d)$ ให้เพิ่มไอเท็ม X นั้นเป็นสมาชิกของ L_2'

- ถ้า $X.\sup port_{UD} < s \times d$ ให้ตัดไอเท็ม X ออกจาก C_2 และไม่ต้องนำไอเท็ม X ไปสแกนใน DB

เมื่อเสร็จขั้นตอนนี้แล้ว ผลลัพธ์ที่ได้คือ L_2'

(3) ทำการวนซ้ำเช่นเดียวกับข้อ (1) เพื่อหา L_k' เมื่อ $k \geq 3$ ในรอบถัดไปจนกว่าจะไม่สามารถหา L_k' ได้

ข้อดีของขั้นตอนวิธี เอฟยูพี

1. มีการนำผลลัพธ์ที่ได้จากการประมวลผลในฐานข้อมูลเดิมมาใช้ร่วมกับการประมวลผลในฐานข้อมูลมูลใหม่ที่ถูกเพิ่มเข้ามา

2. สามารถลดจำนวนไอเท็มเซตคู่แข่งที่จะนำไปสแกนในฐานข้อมูลเดิม

ข้อเสียของขั้นตอนวิธี เอฟยูพี

1. สามารถใช้งานได้ในกรณีที่มีการเพิ่มข้อมูลชุดใหม่เข้าไปเท่านั้น ไม่สามารถใช้ได้กับกรณีการลบ และการปรับปรุง (Modification) ในฐานข้อมูลได้

2. จำเป็นต้องมีการสแกนฐานข้อมูลเดิม เพื่อหาไอเท็มเซตที่เกิดบ่อยใหม่ ทุกรอบ k

2.2.2 ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ [3,4]

ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ เป็นขั้นตอนวิธีที่ถูกออกแบบมาเพื่อลดปัญหาของ อะพริโอริและเอฟยูพี ในด้านความจำเป็นที่จะต้องกลับไปสแกนฐานข้อมูลเดิมจำนวนหลายรอบ ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ สามารถประมวลผลได้ 2 กรณีคือ การเพิ่มข้อมูลใหม่เข้าไปและการลบข้อมูลเก่าออกไปจากฐานข้อมูลเดิม ทั้งนี้ เนกาทีฟบอร์เดอร์ จะทำงานภายใต้ข้อสมมติฐานว่า ค่าสนับสนุนขั้นต่ำ min_sup และค่าความเชื่อมั่นขั้นต่ำ min_*conf* ไม่มีการเปลี่ยนแปลง

หลักการของเนกาทีฟบอร์เดอร์ คือจะมีการเก็บค่า C_k ทั้ง $C_k \in L_k$ และ $C_k \notin L_k$ โดย $C_k \notin L_k$ จะเรียกว่าบอร์เดอร์เซต หรือ Border set (NBd) ตัวอย่างเช่น

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

จะได้ว่า Border set ของ L_1 หรือ $NBd(L_1) = \{C, D\}$

จากตัวอย่างข้างต้น สามารถเขียนให้อยู่ในรูปของสมการได้ ดังนี้

$$NBd(L_k) = C_k - L_k \quad \text{หรือ} \quad C_k = L_k \cup NBd(L_k) \tag{2.3}$$

ความหมายและสัญลักษณ์ที่ใช้ในขั้นตอนวิธีเนกาทีฟบอร์เดอร์ มีรายละเอียดดังนี้

DB	หมายถึง	ฐานข้อมูลเดิม
db	หมายถึง	ฐานข้อมูลใหม่
DB^+	หมายถึง	ฐานข้อมูลปรับปรุง
L^{DB}	หมายถึง	ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม
L^{db}	หมายถึง	ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลใหม่
L^{DB^+}	หมายถึง	ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง
$NBd(L^{DB})$	หมายถึง	ไอเท็มที่เป็นบอร์เดอร์เซตของไอเท็มเซตที่เกิดบ่อยที่อยู่ในฐานข้อมูลเดิม
$NBd(L^{db})$	หมายถึง	ไอเท็มที่เป็นบอร์เดอร์เซตของไอเท็มเซตที่เกิดบ่อยที่อยู่ในฐานข้อมูลใหม่
$NBd(L^{DB^+})$	หมายถึง	ไอเท็มที่เป็นบอร์เดอร์เซตของไอเท็มเซตที่เกิดบ่อยที่อยู่ในฐานข้อมูลปรับปรุง
s	หมายถึง	ไอเท็มเซต s ใดๆ ในฐานข้อมูลทั้ง DB, db และ DB^+
$s.count$	หมายถึง	ค่าสนับสนุนของไอเท็ม s
$t_{DB}(s)$	หมายถึง	จำนวนรายการธุรกรรมใน DB ที่มี s เป็นสมาชิก

$t_{db}(s)$ หมายถึง จำนวนรายการธุรกรรมใน db ที่มี s เป็นสมาชิก

ขั้นตอนการทำงานของขั้นตอนวิธีเนกาทีฟบอร์เดอร์ จะแบ่งการทำงานออกเป็น 2 ส่วน คือ ส่วนของการเพิ่มข้อมูลใหม่ (Addition of new transaction) และการทำงานในส่วนของการลบข้อมูลเดิมออกจากฐานข้อมูล (Deletion of existing transaction) ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ แสดงดังรูปที่ 2.7 โดยในแต่ละส่วนมีรายละเอียดการทำงานดังนี้

Function **Update-Large-Itemset** $(L^{DB}, NBd(L^{DB}), db)$

// DB and db denote the number of transaction in the original database and the increment database respectively

Compute L^{db}

for each itemset $s \in L^{DB} \cup NBd(L^{DB})$ **do**

$t_{db}(s) =$ number of transactions in db containing s

$L^{DB+} = \phi$

for each itemset $s \in L^{DB}$ **do**

if $\left(t_{DB}(s) + t_{db}(s) \geq \min\sup^{*}(DB + db)\right)$ **then**

$L^{DB+} = L^{DB+} \cup s$

for each itemset $s \in L^{db}$ **do**

if $s \notin L^{DB}$ and $s \in NBd(L^{DB})$ and $\left(t_{DB}(s) + t_{db}(s) \geq \min\sup^{*}(DB + db)\right)$ **then**

$L^{DB+} = L^{DB+} \cup s$

if $L^{DB+} \neq L^{DB+}$ **then**

$NBd(L^{DB+}) = negativeborder - gen(L^{DB+})$

else $NBd(L^{DB+}) = NBd(L^{DB})$

if $L^{DB} \cup NBd(L^{DB}) \neq L^{DB+} \cup NBd(L^{DB+})$ **then**

$S = L^{DB+}$

repeat

compute $S = S \cup NBd(S)$

until S does not grow

$L^{DB+} = \{x \in S \mid \sup port(x) \geq \min\sup\}$

// $\sup port(x)$ is the support count of x in $DB \cup db$

$NBd(L^{DB+}) = negativeborder - gen(L^{DB+})$

ภาพที่ 2.7 ขั้นตอนการทำงานของขั้นตอนวิธีเนกาทีฟบอร์เดอร์

2.2.2.1 การเพิ่มข้อมูลใหม่ (Addition of new transactions)

กระบวนการทำงานการเพิ่มขยายกฎความสัมพันธ์เมื่อมีฐานข้อมูลใหม่เพิ่มเข้าไปในฐานข้อมูลเดิมของขั้นตอนวิธี เนกาทีฟบอร์เดอร์ มีรายละเอียดการทำงานดังนี้

(1) ปรับปรุงค่าสนับสนุนให้กับไอเท็มเซตที่เป็นสมาชิกของ L_k และ $NBd(L_k)$ ในฐานข้อมูลเดิม โดยเริ่มจากการสแกนรายการธุรกรรมใหม่ที่ถูกเพิ่มเข้ามาในฐานข้อมูล เพื่อคำนวณหาไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่ L_k^{db} ในขณะเดียวกันให้ปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เป็นทั้ง L_k และ $NBd(L_k)$ ในฐานข้อมูลเดิมด้วย จากนั้นให้พิจารณาดังนี้

■ กรณี $s \in L^{DB}$

- ถ้า $t_{DB}(s)+t_{db}(s) < \min\_\sup\times(t_{DB}+t_{db})$ แล้ว ให้ตัด ไอเท็ม s นั้นออกจาก L^{DB}

- ถ้า $t_{DB}(s)+t_{db}(s) \geq \min\_\sup\times(t_{DB}+t_{db})$ แล้ว ให้เพิ่มไอเท็ม s นั้นเข้าเป็นสมาชิกของ L^{DB}

■ กรณี $s \in L^{db}$ และ $s \notin L^{DB}$ และ $s \in NBd(L^{DB})$

- ถ้า $t_{DB}(s)+t_{db}(s) \geq \min\_\sup\times(t_{DB}+t_{db})$ แล้ว ให้เพิ่มไอเท็ม s นั้นเข้าเป็นสมาชิกของ L^{DB^+}

(2) หลังจากปรับปรุงค่าสนับสนุนให้แก่ไอเท็มเซตที่เป็นสมาชิกของ L^{DB} และ $NBd(L^{DB})$ ในฐานข้อมูลเดิมเรียบร้อยแล้ว ผลลัพธ์ที่ได้คือไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง หรือ L^{DB^+} จากนั้นจะทำการเปรียบเทียบความแตกต่างของไอเท็มเซตใน L^{DB} และ L^{DB^+} ดังนี้

■ กรณี $L^{DB} = L^{DB^+}$ หมายถึง ไม่มีการเปลี่ยนแปลงของไอเท็มเซตใน DB และ DB^+ กล่าวอีกนัยหนึ่งคือ ไอเท็มที่เซตที่เป็นสมาชิกของ NBd ไม่มีการเปลี่ยนแปลง นั่นคือ $NBd(L^{DB}) = NBd(L^{DB^+})$

■ กรณี $L^{DB} \neq L^{DB^+}$ หมายถึง มีการเปลี่ยนแปลงของไอเท็มเซตใน ใน DB และ DB^+ กล่าวอีกนัยหนึ่งคือไอเท็มเซตที่เป็นสมาชิกของ NBd มีการเปลี่ยนแปลง นั่นคือ $NBd(L^{DB}) \neq NBd(L^{DB^+})$ ทำให้ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ ต้องคำนวณค่า L^{DB^+} ใหม่โดยใช้ฟังก์ชัน $Negativeborder-gen(L)$ แสดงดังรูปที่ 2.8 โดยการนำเอา L^{DB^+} ไปคำนวณหา $NBd(L^{DB^+})$ ในแต่ละรอบ k ตามหลักการของ Level-wise-step

Function $negativeborder-gen(L)$

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

ภาพที่ 2.8 ขั้นตอนการทำงานของฟังก์ชัน $negativeborder-gen(L)$

(3) จากขั้นตอนที่ (2) เมื่อทำการคำนวณหา $NBd(L^{DB^+})$ เรียบร้อยแล้ว จะนำ $L^{DB} \cup NBd(L^{DB})$ ของฐานข้อมูลเดิม มาเปรียบเทียบกับ $L^{DB^+} \cup NBd(L^{DB^+})$ ของฐานข้อมูลใหม่ เพื่อพิจารณาความเปลี่ยนแปลง หากพบว่า $L^{DB} \cup NBd(L^{DB}) \neq L^{DB^+} \cup NBd(L^{DB^+})$ ขั้นตอนวิธีจะทำการหาค่า $Negativeborder-closure$ ของ L^{DB^+} และจะทำการสแกนฐานข้อมูลเดิมอีกครั้งเพื่อปรับปรุงค่า L^{DB} และ $NBd(L^{DB})$

2.2.2.2 การลบข้อมูลเดิม (Deletion of existing transactions)

กรณีที่มีการลบข้อมูลออกจากฐานข้อมูลเดิม เนกาทีฟบอร์เดอร์ จะคำนวณค่า L^{DB^-} และ $NBd(L^{DB^-})$ ใหม่ ด้วยการนำค่า $s.count$ ใน db มาลบออกจาก L^{DB} และ $NBd(L^{DB})$ แล้วตรวจสอบค่า $s.count_{DB-db}$ กับค่า $\min\_sup \times (DB-db)$ เช่นเดียวกับขั้นตอนที่ (1) – (3) ในขั้นตอนการเพิ่มข้อมูลใหม่ (Addition of new transactions)

ข้อดีของขั้นตอนวิธี เนกาทีฟบอร์เดอร์

1. มีการใช้ไอเท็มเซตที่เรียกว่า บอร์เดอร์เซต $NBd(L)$ เป็นตัวตัดสินใจในการสแกนฐานข้อมูลเดิม

2. มีการนำไอเท็มเซตกลับไปสแกนในฐานข้อมูลเดิมเพียงครั้งเดียวเท่านั้น ในกรณี $L^{DB} \cup NBd(L^{DB}) \neq L^{DB^+} \cup NBd(L^{DB^+})$

ข้อเสียของขั้นตอนวิธี เนกาทีฟบอร์เดอร์

1. ใช้ได้เฉพาะในกรณีไม่มีไอเท็มใหม่ (new item) ใหม่เกิดขึ้น

2. การหา $Negativeborder-closure$ ใช้เวลานานในการหาไอเท็มเซตที่เกิดบ่อย เนื่องจากต้องมีการวนซ้ำหลายรอบเพื่อให้ได้ $L^{DB} \cup NBd(L^{DB}) = L^{DB^+} \cup NBd(L^{DB^+})$ ทั้งหมด

3. ต้องใช้พื้นที่เก็บข้อมูลเพิ่มขึ้นเพื่อใช้เก็บ L^{DB} และ $NBd(L^{DB})$ ทั้งหมด

2.2.3 ขั้นตอนวิธีพรีลาร์จ [5]

ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ เก็บไอเท็มเซตทั้งไอเท็มเซตที่เกิดบ่อย L^{DB} และไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย $NBd(L^{DB})$ เพื่อหลีกเลี่ยงการสแกนฐานข้อมูลทั้งหมดในขั้นตอนการปรับปรุงค่าสนับสนุน การเก็บ $NBd(L^{DB})$ ทั้งหมด ทำให้สิ้นเปลืองพื้นที่ในการจัดเก็บ ด้วยปัญหาดังกล่าว ขั้นตอนวิธี พรีลาร์จ ได้เสนอแนวคิดให้จัดเก็บ $NBd(L^{DB})$ เฉพาะไอเท็มที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง L^{UD} ทั้งนี้ ได้นิยามไอเท็มที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงว่าพรีลาร์จไอเท็มเซต (Pre-Large itemset) ด้วยแนวคิดดังกล่าวช่วยให้ใช้พื้นที่ในการจัดเก็บไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยน้อยลง

ขั้นตอนวิธีพรีลาร์จ ได้เสนอแนวคิดในการพิจารณาว่าจะสแกนฐานข้อมูลเดิมหรือไม่ด้วยการกำหนดค่าทดสอบ 2 ค่าเพื่อใช้ในการทดสอบ ได้แก่ ค่าสนับสนุนที่เป็นขอบเขตล่าง (Lower support factor) และค่าสนับสนุนที่เป็นขอบเขตบน (Upper support factor) โดยค่าสนับสนุนที่เป็นขอบเขตบนก็คือค่าสนับสนุนขั้นต่ำ min_sup ที่ผู้ใช้เป็นผู้กำหนด ส่วนค่า lower support คือ ค่าใหม่อีกหนึ่งค่าที่ผู้ใช้ต้องกำหนดเพื่อตรวจสอบว่าไอเท็มใดบ้างที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งทั้งค่า ค่าสนับสนุนที่เป็นขอบเขตล่างและค่าสนับสนุนที่เป็นขอบเขตบนจะมีค่าอยู่ระหว่าง 0 – 100 %

นอกจากค่าค่าสนับสนุนที่เป็นขอบเขตล่างและค่าสนับสนุนที่เป็นขอบเขตบน จะใช้ในการพิจารณาการเป็นไอเท็มเซตที่เกิดบ่อย และ พรีลาร์จไอเท็มเซต แล้ว ทั้ง 2 ค่า จะถูกใช้เพื่อคำนวณ หาจำนวนที่ปลอดภัย (Safety number) แทนด้วยสัญลักษณ์ f เพื่อทดสอบว่าในการเพิ่มข้อมูลใหม่แต่ละครั้งแล้วทำการประมวลผล จำเป็นที่จะต้องกลับไปสแกนฐานข้อมูลเดิมหรือไม่ ซึ่งจะช่วยลดการสแกนฐานข้อมูล โดยคำนวณดังนี้

$$f = \frac{(S_u - S_l)d}{1 - S_u} \tag{2.4}$$

เมื่อ f คือ จำนวนที่ปลอดภัยหรือจำนวนรายการธุรกรรมใหม่ที่ทำให้การประมวลผลไม่จำเป็นต้องกลับไปสแกนฐานข้อมูลเดิม

S_u คือ ค่าสนับสนุนที่เป็นขอบเขตบน ที่ผู้ใช้กำหนด (ค่า min_sup)

S_l คือ ค่าสนับสนุนที่เป็นขอบเขตล่างที่ผู้ใช้กำหนด

d คือ จำนวนรายการธุรกรรมของฐานข้อมูลเดิม

ตัวอย่าง กำหนดให้ฐานข้อมูลเดิมมีจำนวนรายการธุรกรรม $d=100$ และผู้ใช้กำหนดค่า $S_l=50\%$ และ $S_u=60\%$ การคำนวณจำนวนที่ปลอดภัยหรือจำนวนรายการธุรกรรมใหม่ที่ทำให้การประมวลผลไม่จำเป็นต้องกลับไปสแกนฐานข้อมูลเดิม แสดงดังนี้

$$f=\frac{(0.6-0.5)100}{1-0.6}=25$$

จากตัวอย่างการคำนวณ หมายความว่า ถ้าจำนวนรายการธุรกรรมใหม่มีค่าน้อยกว่าหรือเท่ากับ 25 รายการธุรกรรม การทำงานของขั้นตอนวิธีก็ไม่จำเป็นที่จะต้องกลับไปสแกนฐานข้อมูลเดิม

กระบวนการทำงานของขั้นตอนวิธี พรีลาร์จ มีรายละเอียดดังนี้

1. ข้อมูลนำเข้าก่อนการประมวลผล ประกอบด้วย $L_k^D, P_k^D, S_l, S_u, d$ เมื่อ กำหนดให้

L_k^D คือ ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม

P_k^D คือ พรีลาร์จไอเท็มเซต ของฐานข้อมูลเดิม

S_l คือ ค่าสนับสนุนที่เป็นขอบเขตล่าง

S_u คือ ค่าสนับสนุนที่เป็นขอบเขตบน

d คือ จำนวนรายการธุรกรรมในฐานข้อมูลเดิม

2. ในการเพิ่มข้อมูลครั้งแรกจะกำหนดตัวแปร $c=0$ เมื่อตัวแปร c เป็นตัวแปรที่ใช้เก็บจำนวนรายการธุรกรรมใหม่ที่ถูกเพิ่มเข้ามา โดยจะสะสมค่าไปเรื่อยๆ ในแต่ละรอบการเพิ่มรายการธุรกรรมใหม่

3. คำนวณค่าจำนวนที่ปลอดภัย (f) ด้วยสูตรคำนวณ ตามสมการที่ 2.4

4. สแกนฐานข้อมูลใหม่เพื่อหาค่าสนับสนุนของ C_k^T เมื่อ T คือฐานข้อมูลใหม่

5. แบ่งประเภทของ C_k^T ออกเป็น 3 ประเภท ได้แก่ $C_k^T \in L_k^{DB}, C_k^T \in P_k^D$ และ $C_k^T \notin (L_k^D \cup P_k^D)$

6. ให้พิจารณา C_k^T แต่ละตัวดังนี้

6.1 กรณี $C_k^T \in L_k^D$ และ $C_k^T \in P_k^D$

■ ให้ปรับปรุงค่าสนับสนุนโดย $S^U(I)=S^T(I)+S^D(I)$ เมื่อ $S^U(I)$, $S^T(I)$ และ $S^D(I)$ คือค่าสนับสนุนของไอเท็ม I ในฐานข้อมูลปรับปรุง ฐานข้อมูลใหม่ และฐานข้อมูลเดิมตามลำดับ

■ พิจารณาไอเท็มที่ปรับปรุงค่าสนับสนุนแล้ว ถ้าหาก $\frac{S^U(I)}{d+t+c} \geq S_u$ แล้ว จะกำหนดให้ไอเท็ม I นั้นเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง L_k^U

■ พิจารณาไอเท็มที่ปรับปรุงค่าสนับสนุนแล้ว ถ้าหาก $\frac{S^U(I)}{d+t+c} \geq S_l$ แล้ว จะกำหนดให้ไอเท็ม I นั้นเป็น พรีลาร์จไอเท็มเซตของฐานข้อมูลปรับปรุง P_k^U

■ พิจารณาไอเท็มที่ปรับปรุงค่าสนับสนุนแล้ว ถ้าหาก $\frac{S^U(I)}{d+t+c} < S_l$ แล้ว ขั้นตอนวิธีจะไม่เก็บไอเท็ม I ดังกล่าว เพราะไม่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อย

6.2 กรณี $C_k^T \notin (L_k^D \cup P_k^D)$

■ พิจารณาค่าสนับสนุนของ C_k^T เฉพาะค่าสนับสนุนที่คำนวณได้จากฐานข้อมูลใหม่ ถ้า ไอเท็ม I เป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลใหม่ L_k^T หรือเป็น พรีลาร์จของฐานข้อมูลใหม่ P_k^T ให้เก็บไอเท็ม I ดังกล่าวไว้ในตัวแปร R เพื่อนำกลับไปสแกนในฐานข้อมูลเดิมในขั้นตอนที่ 7 (กรณีที่จำเป็น)

7. พิจารณาการนำไอเท็มเซตกลับไปสแกนในฐานข้อมูลเดิม ดังนี้

7.1 ถ้า $t+c \leq f$ หรือ $R=\phi$ ไม่จำเป็นต้องกลับไปสแกนฐานข้อมูลเดิม

7.2 ถ้า $t+c > f$ หรือ $R \neq \phi$ ให้นำไอเท็มที่อยู่ในตัวแปร R ไปสแกนในฐานข้อมูลเดิมแล้วปรับปรุงค่าสนับสนุน จากนั้นให้พิจารณาว่าเป็น L_k^U หรือ P_k^U หรือไม่

8. ทำซ้ำตั้งแต่ข้อ 4 – 7 จนกว่าจะไม่สามารถหา L_k^U และ P_k^U ได้ต่อไป

9. ถ้า $t+c > f$ ให้ปรับปรุงค่า d โดย $d=d+t+c$ โดยให้ค่า $c=0$ แต่ถ้าหากคำนวณแล้วปรากฏว่า $t+c \leq f$ ให้ปรับปรุงค่า c ใหม่เป็น $c=t+c$

ข้อดีของขั้นตอนวิธี พรีลาร์จ

1. มีการเก็บไอเท็มที่เรียกว่า พรีลาร์จ เพื่อลดจำนวนการเก็บ $NBd(L^{DB})$ ที่ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ นำเสนอ

2. มีการคำนวณจำนวนที่ปลอดภัย เพื่อพิจารณาการนำไอเท็มกลับไปสแกนฐานข้อมูลเดิม ซึ่งหากรายการธุรกรรมมีจำนวนไม่เกินค่าที่กำหนดก็ไม่จำเป็นต้องกลับไปสแกนฐานข้อมูลเดิม

ข้อเสียของขั้นตอนวิธี พรีลาร์จ

ในกรณีที่มีจำนวนรายการธุรกรรมใหม่ที่ถูกเพิ่มเข้ามามากกว่าจำนวนที่ปลอดภัยที่คำนวณไว้ ขั้นตอนวิธี พรีลาร์จ ก็จำเป็นต้องกลับไปสแกนฐานข้อมูลเดิม

2.2.4 ขั้นตอนวิธีการประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย [12]

การประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย เป็นขั้นตอนวิธีที่ศึกษาด้านการเพิ่มขยายกฎความสัมพันธ์ โดยพยายามหลีกเลี่ยงการสแกนฐานข้อมูลเดิม หลักการของขั้นตอน

วิธีนี้คือ การใช้ค่าสนับสนุนสูงสุด (Maximum support count) ของ C_1 ที่ได้จากการคำนวณในฐานข้อมูลเดิม มาทำการพยากรณ์หาไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่มีโอกาสที่จะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งในขั้นตอนวิธีนี้จะเรียกไอเท็มดังกล่าวว่าไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย (Promising frequent itemset : PL)

ในการหา L_k และ PL_k ของขั้นตอนวิธีนี้ จะใช้หลักการเช่นเดียวกับ อะพริโอริ แต่จะมีความแตกต่างกันในส่วนของกระบวนการเชื่อมไอเท็มเซตเพื่อสร้าง C_k ซึ่งขั้นตอนวิธี อะพริโอริ จะมีการเชื่อมไอเท็มเซตเพื่อสร้างแคนดิดเดตไอเท็มเซต โดย $C_k = L_{k-1} * L_{k-1}$ ในขณะที่ ขั้นตอนวิธีการประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย จะเชื่อมไอเท็มเซตเพื่อสร้างไอเท็มเซตคู่แข่ง โดย $C_k = (L_{k-1} \cup PL_{k-1}) * (L_{k-1} \cup PL_{k-1})$

ความหมายของสัญลักษณ์ที่ใช้ในขั้นตอนวิธีการประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย มีรายละเอียดดังนี้

$\min_{PL}$ หมายถึง ค่าที่ใช้ตรวจสอบว่าไอเท็มเซตใดที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่มีโอกาสที่จะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง คำนวณได้จากสมการ 2.5

$$\min\_\sup_{DB} - \left[\frac{\max \sup}{totalsize} \times inc\_size \right] \leq \min_{PL} < \min\_\sup_{DB} \qquad (2.5)$$

เมื่อกำหนดให้

$\min\_\sup_{DB}$ หมายถึง ค่าของฐานข้อมูลเดิม

$\max \sup$ หมายถึง ค่าสนับสนุนของ C_1 ที่มีค่ามากที่สุด

$totalsize$ หมายถึง จำนวนรายการธุรกรรมที่มีอยู่ในฐานข้อมูลเดิม

inc_size หมายถึง จำนวนรายการธุรกรรมที่มีอยู่ในฐานข้อมูลใหม่ที่คาดว่าจะถูกเพิ่มเข้ามาในฐานข้อมูลเดิม

สัญลักษณ์ที่ใช้ในขั้นตอนวิธีการประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย มีดังนี้

L_{DB}^k หมายถึง frequent $k-itemset$ ในฐานข้อมูลเดิม เมื่อ $k \geq 1$

PL_{DB}^k หมายถึง promising frequent $k-itemset$ ในฐานข้อมูลเดิม เมื่อ $k \geq 1$

$L_{(DB \cup db)}^k$ หมายถึง frequent $k-itemset$ ในฐานข้อมูลปรับปรุง เมื่อ $k \geq 1$

$PL_{(DB \cup db)}^k$ หมายถึง promising frequent $k-itemset$ ในฐานข้อมูลปรับปรุง เมื่อ $k \geq 1$

C_{DB}^{k} หมายถึง candidate $k-itemset$ ในฐานข้อมูลเดิม เมื่อ $k \geq 1$

C_{db}^{k} หมายถึง candidate $k-itemset$ ในฐานข้อมูลใหม่ เมื่อ $k \geq 1$

$X.\text{sup}\,port$ หมายถึง ค่าสนับสนุนของไอเท็ม X ใดๆ

ขั้นตอนวิธีนี้ แบ่งขั้นตอนการทำงานออกเป็น 2 ส่วน คือ 1) ส่วนของการประมวลผลฐานข้อมูลเดิม (Original database discovery) และ 2) ส่วนของการประมวลผลเพื่อปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย (Updating frequent and promising frequent itemset) รายละเอียดการทำงานทั้ง 2 ส่วน ดังนี้

1. การทำงานในส่วนของการประมวลผลฐานข้อมูลเดิม

การทำงานในส่วนนี้ มีวัตถุประสงค์เพื่อคำนวณหาไอเท็มเซตที่เกิดบ่อยและ ไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย ขั้นตอนการทำงานของขั้นตอนวิธีนี้แสดงดังรูปที่ 2.9 รายละเอียดการทำงาน ดังนี้

1.1 สแกนฐานข้อมูลเดิมเพื่อหาค่าสนับสนุนของ C_1 และคำนวณหาค่า max sup เพื่อใช้ในการคำนวณหา $\min_{PL}$ ด้วยสมการ (2.5) เพื่อใช้ในการหา promising frequent itemset (PL_k)

1.2 หา L_{DB}^{1} และ PL_{DB}^{1} ด้วยวิธีการเช่นเดียวกับขั้นตอนวิธี อะพริโอริ โดยค่า min_sup ที่กำหนดโดยผู้ใช้ จะใช้ทดสอบหา L_{DB}^{1} ส่วนค่า $\min_{PL}$ ที่คำนวณได้จากขั้นตอนที่ 1.1 จะใช้ทดสอบหา PL_{DB}^{1} เมื่อเสร็จสิ้นขั้นตอนนี้ ผลัพธ์ที่ได้คือ $X \in L_{DB}^{1}$ และ $X \in PL_{DB}^{1}$ เมื่อ X คือไอเท็มเซต

1.3 สร้าง C_2 ด้วยวิธีการเชื่อมไอเท็มเซต โดย $C_2 = (L_1 \cup PL_1) * (L_1 \cup PL_1)$

1.4 ทำซ้ำข้อ 1.1 – 1.3 เพื่อหา L_{DB}^{k} และ PL_{DB}^{k} เมื่อ $k \geq 2$ จนกระทั่งไม่สามารถหา L_{DB}^{k} และ PL_{DB}^{k} ได้

ผลลัพธ์ที่ได้จากการทำงานในส่วนการประมวลผลฐานข้อมูลเดิม คือ L_{DB}^{k} และ PL_{DB}^{k} ของฐานข้อมูลเดิม

2. การทำงานในส่วนของการปรับปรุงไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย

การทำงานในส่วนนี้ มีวัตถุประสงค์เพื่อปรับปรุงค่าสนับสนุนของ L_{DB}^{k} และ PL_{DB}^{k} ของฐานข้อมูลเดิม ให้เป็น L_{DB}^{k} และ PL_{DB}^{k} ของฐานข้อมูลปรับปรุง ผลลัพธ์ที่ได้จากการประมวลส่วนนี้ อาจจะแสดงให้เห็นถึงการเปลี่ยนแปลงของไอเท็มเซต เช่น PL_{DB}^{k} อาจจะเปลี่ยนเป็น $L_{(DB \cup db)}^{k}$ เป็นต้น

การทำงานในส่วนนี้ จะแบ่งการทำงานออกเป็น 2 ส่วนย่อยได้แก่ 2.1) ส่วนการปรับปรุงค่าสนับสนุนของ L^1_{DB} และ PL^1_{DB} และ 2.2) ส่วนการปรับปรุงค่าสนับสนุนของ $L^{k\geq 2}_{DB}$ และ $PL^{k\geq 2}_{DB}$ รายละเอียดการทำงาน เป็นดังนี้

2.1 การทำงานส่วนปรับปรุงไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด 1–ไอเท็มเซต

การทำงานในส่วนของการปรับปรุงค่าสนับสนุนของ ไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด 1– ไอเท็มเซตแสดงดังรูปที่ 2.9 มีรายละเอียดการทำงานดังนี้

2.1.1 สแกน db เพื่อหาค่าสนับสนุนของไอเท็ม X ที่เป็นสมาชิกของ C^1_{db} แล้วปรับปรุงค่าสนับสนุน โดย $X.\sup port_{DB\cup db} = X.\sup port_{DB} + X.\sup port_{db}$ จากนั้นจะพิจารณาเพื่อหาไอเท็มเซตที่เป็นทั้ง $L^1_{DB\cup db}$ และ $PL^1_{DB\cup db}$ ดังนี้

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

- ถ้า $X.\sup port_{DB\cup db} \geq \min\_\sup\times(DB+db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $L^1_{DB\cup db}$ และ $Temp_1$

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

- ถ้า $X.\sup port_{DB\cup db} \geq \min\_\sup\times(DB+db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $L^1_{DB\cup db}$

- ถ้า $\min_{PL_{DB\cup db}} \leq X.\sup port_{DB\cup db} < \min\_\sup\times(DB+db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $PL^1_{DB\cup db}$ ทั้งนี้ $\min_{PL_{DB\cup db}}$ คำนวณได้จากสมการ 2.6

$$\min_{PL_{DB\cup db}} = \min\_\sup_{DB\cup db} - \left[\frac{\max\sup}{totalsize}\times inc\_size\right] \qquad (2.6)$$

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

- ถ้า $X.\sup port_{DB\cup db} \geq \min\_\sup\times(DB+db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $L^1_{DB\cup db}$ และ $Temp_1$

- ถ้า $\min_{PL_{DB\cup db}} \leq X.\sup port_{DB\cup db} < \min\_\sup\times(DB+db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $PL^1_{DB\cup db}$

สำหรับทั้ง 3 กรณี (1) (2) และ (3) จะเป็นการตรวจสอบไอเท็มที่ปรากฏอยู่ในฐานข้อมูลเดิม ส่วนในกรณีถัดไป (4) จะเป็นกรณีที่เป็นไอเท็มใหม่ที่เพิ่งปรากฏขึ้นมาในฐานข้อมูลใหม่ที่ถูกเพิ่มเข้ามา

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 $1-itemset$ in original database.

(3) C^1_{DB} : candidate $1-itemset$ of original database.

(4) C^1_{db} : candidate $1-itemset$ of increment database.

Output:

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

(2) $PL^1_{(DB\cup db)}$: promising frequent $1-itemset$ in updated database.

(3) new frequent itemset : new frequent itemset in updated database.

(4) new promising frequent itemset : new promising frequent itemset in updated database.

(5) $Temp\_newC_k$: new candidate $2-itemset$ in updated database.

1 C^1_{db} = all $1-itemset$ in db with support > 0

2 $k=1$

3 While $C^1_{db} > 0$ do

4 for each $X \in C^1_{DB}$ do $X.\sup port_{(DB\cup db)} = X.\sup port_{DB} + X\sup port_{db}$

5 if ($X \notin L^1_{DB}$ or $X \notin PL^1_{DB}$) and $X.\sup port_{(DB\cup db)} \geq \min\_\sup_{(DB\cup db)}$ then

6 Add X to $L^1_{(DB\cup db)}$ and Add X to $Temp_1$

7 for each $X \in L^1_{DB}$ do

8 if $X.\sup port_{(DB\cup db)} \geq \min\_\sup_{(DB\cup db)}$ then

9 Add X to $L^1_{(DB\cup db)}$

10 else if $X.\sup port_{(DB\cup db)} \geq \min\_PL_{(DB\cup db)}$ then

11 Add X to $PL^1_{(DB\cup db)}$

12 for each $X \in PL^1_{DB}$ do

13 if $X.\sup port_{(DB\cup db)} \geq \min\_\sup_{(DB\cup db)}$ then

14 Add X to $L^1_{(DB\cup db)}$ and Add X to $Temp_1$

15 else if $X.\sup port_{(DB\cup db)} \geq \min\_PL_{(DB\cup db)}$ then

16 Add X to $PL^1_{(DB\cup db)}$

17 for each $X \notin C^1_{DB}$ do Add X to $C^1_{(DB\cup db)}$

18 if $X.\sup port_{db} \geq \min\_\sup_{(DB\cup db)}$ (new item in db) then

19 Add X to $L^1_{(DB\cup db)}$ and Add X to $Temp_1$

20 else if $X.\sup port_{db} \geq \min\_PL_{(DB\cup db)}$ then

21 Add X to $PL^1_{(DB\cup db)}$ and Add X to $Temp_1$

ภาพที่ 2.9 ขั้นตอนวิธี การปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย กรณี 1-ไอเท็มเซต

22 if $Temp_1 \neq \phi$ then
23 $Y \in Temp_1$
24 $C^2_{(DB \cup db)}(new) = gen\_newcandidate(Y)$
25 clear $Temp_1$ and Add $C^2_{(DB \cup db)}(new)$ to $Temp\_newC_k$
26 $k = k + 1$

ภาพที่ 2.9 (ต่อ) ขั้นตอนวิธี การปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย กรณี 1-ไอเท็มเซต

(4) กรณี $X \notin C^1_{DB}$ (มีไอเท็มใหม่เกิดขึ้น)

- ถ้า $X.\sup port_{db} \geq \min\_\sup \times (DB + db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $L^1_{DB \cup db}$ และ $Temp_1$

- ถ้า $\min_{PL_{DB \cup db}} \leq X.\sup port_{db} < \min\_\sup \times (DB + db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $PL^1_{DB \cup db}$ และ $Temp_1$

2.1.2 $Temp_1$ จะเป็นตัวแปรที่ใช้เก็บไอเท็มที่เป็นไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยตัวใหม่ในฐานข้อมูลปรับปรุง เพื่อทำการสร้าง $C^2_{DB \cup db}$ ตัวใหม่จากฟังก์ชัน *gen_newcandidate* แสดงดังรูปที่ 2.10 ซึ่งจะใช้หลักการเชื่อมไอเท็มเซต และการตัด เช่นเดียวกับ อะพริโอริ จากนั้นจะทำการเก็บค่า $C^2_{DB \cup db}$ ไว้ใน $Temp\_newC_k$ เพื่อใช้คำนวณในส่วนการปรับปรุงไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซต เมื่อ $k \geq 2$ ซึ่งรายละเอียดขั้นตอนจะอยู่ในหัวข้อที่ 2.2 เป็นลำดับถัดไป

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) $Temp_1$: new frequent $k-itemset$ in updated database.
Output:
(1) $newC^{k+1}$: new candidate $k+1-itemset$ in updated database.

1 if $k \leq (length(L) + length(PL))$
2 For each $Y \in Temp_1$
3 $C^{k+1}_{new} = Y * (L^k_{DB \cup db} \cup PL^k_{DB \cup db})$
4 For $c \in C^{k+1}_{new}$
5 Delete c from $C^{k+1}_{DB_{new}}$ if all subset of c is in L^k or PL^k

รูปที่ 2.10 ขั้นตอนการทำงานของฟังก์ชัน *Gen_newcandidate*

2.2 การทำงานของขั้นตอนวิธีในส่วนการปรับปรุงไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k \geq 2$ ไอเท็มเซต

การทำงานในส่วนนี้ เป็นการปรับปรุงค่าสนับสนุนของ ไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k \geq 2$ ไอเท็มเซต แสดงดังรูปที่ 2.11 ซึ่งมีรายละเอียดการทำงานแต่ละรอบ k ดังนี้

2.2.1 นำไอเท็มเซตทุกตัวที่เป็นสมาชิกของ L^2_{DB} , PL^2_{DB} และ $Temp\_newC_k$ มาทำการสแกนใน db เพื่อปรับปรุงค่าสนับสนุนไอเท็มเซตของฐานข้อมูลปรับปรุง ($X.\sup port_{DB \cup db}$)

2.2.2 จากนั้นจะพิจารณาหาไอเท็มเซตที่เป็นสมาชิกของ $L^2_{DB \cup db}$ และ $PL^2_{DB \cup db}$ ดังนี้

■ กรณี $X \in L^2_{DB}$

- ถ้า $X.\sup port_{DB \cup db} \geq \min\_\sup \times (DB + db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $L^2_{DB \cup db}$

- ถ้า $\min_{PL_{DB \cup db}} \leq X.\sup port_{DB \cup db} < \min\_\sup \times (DB + db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $PL^2_{DB \cup db}$ และ $Temp_1$

■ กรณี $X \in PL^2_{DB}$

- ถ้า $X.\sup port_{DB \cup db} \geq \min\_\sup \times (DB + db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $L^2_{DB \cup db}$ และ $Temp_1$

- ถ้า $\min_{PL_{DB \cup db}} \leq X.\sup port_{DB \cup db} < \min\_\sup \times (DB + db)$ ให้เพิ่มไอเท็ม X เข้าไปใน $PL^2_{DB \cup db}$

■ กรณี $X \in Temp\_newC_k$

- ถ้า $Y.\sup port_{db} \geq \min\_\sup \times db$ ให้เพิ่มไอเท็ม Y เข้าไปใน $Temp\_scanDB(L^2_{DB \cup db})$ และ $Temp_1(L^2_{DB \cup db})$

- ถ้า $Y.\sup port_{db} \geq \min_{PL_{DB \cup db}}$ หรือ $Y.\sup port_{db} \geq \min_{PL_{DB}}$ ให้เพิ่ม ไอเท็ม Y เข้าไปใน $Temp\_scanDB(PL^2_{DB \cup db})$ และ $Temp_1(PL^2_{DB \cup db})$

ทั้งนี้ไอเท็มเซตทุกตัวที่ถูกเก็บไว้ใน $Temp\_scanDB$ จะถูกนำไปสแกนในฐานข้อมูลเดิมในฟังก์ชัน $Find\_\sup portcountDB$ แสดงดังรูปที่ 2.12 เพื่อปรับปรุงค่าสนับสนุนและ $L^k_{DB \cup db}$ และ $PL^k_{DB \cup db}$ ที่เกิดขึ้นใหม่ เมื่อ $k \geq 2$

2.2.3 นำไอเท็มเซตทุกตัวที่เก็บไว้ใน $Temp_1$ มาสร้าง $C^3_{DB \cup db}$ ตัวใหม่จากฟังก์ชัน *Gen_newcandidate* เพื่อใช้คำนวณในรอบที่ k ถัดไป โดยวนซ้ำตั้งแต่ขั้นตอนที่ 2.2.1 – 2.2.3

ข้อดีของขั้นตอนวิธีการประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย

1. สามารถลดจำนวนครั้งในการสแกนฐานข้อมูลได้
2. มีการใช้พื้นที่ในการเก็บ L_k และ PL_k น้อยกว่าเดิม เมื่อเทียบกับขั้นตอนวิธี เนกาทีฟบอร์เดอร์

ข้อเสียของขั้นตอนวิธีการประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย

1. ใช้ได้เฉพาะในกรณีที่มีการเพิ่มข้อมูลเข้าไปใหม่เท่านั้น
2. ในการคำนวณ $\min_{PL}$ จะต้องมีการคาดคะเนขนาดของฐานข้อมูลใหม่ไว้ล่วงหน้า ซึ่งในสถานการณ์จริงขนาดของฐานข้อมูลใหม่ที่คาดคะเนไว้ล่วงหน้าอาจจะไม่เท่ากับขนาดที่แท้จริงของฐานข้อมูลใหม่ที่ถูกเพิ่มเข้ามา ซึ่งทำให้ขั้นตอนวิธีการประมาณไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย มีโอกาสคำนวณ ไอเท็มเซตที่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อย ผิดพลาด

Algorithm Updating frequent and promising frequent for $k \geq 2 - itemset$

Input:

(1) L_{DB}^{k} : frequent $k-itemset$ in original database.

(2) PL_{DB}^{k} : promising frequent $k-itemset$ in original database.

(3) $Temp\_newC_k$: new candidate $k-itemset$ in updated database.

Output:

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

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

(3) $Temp\_scanDB$:

(4) $Temp_1$: new estimated frequent $k-itemset$ and estimated promising frequent $k-itemset$ in updated database

(5) $Temp\_newC_k$: new candidate $k+1-itemset$ in updated database.

1 $k = 2$

2 While $k \leq (length(L^k) + length(PL^k))$ do

3 scan db for $\forall(L^k), \forall(PL^k)$ and $\forall(item) \in Temp\_newC_k$

4 $X.\sup port_{(DB \cup db)} = X.\sup port_{DB} + X \sup port_{db}$

5 For each $X \in L_{DB}^{k}$ do

6 if $X.\sup port_{(DB \cup db)} \geq \min\_\sup_{(DB \cup db)}$ Then

7 Add X to $L_{(DB \cup db)}^{k}$

8 else if $X.\sup port_{(DB \cup db)} \geq \min\_PL_{(DB \cup db)}$ Then

9 Add X to $PL_{(DB \cup db)}^{k}$

10 For each $X \in PL_{DB}^{k}$ do

11 if $X.\sup port_{(DB \cup db)} \geq \min\_\sup_{(DB \cup db)}$ Then

12 Add X to $L_{(DB \cup db)}^{k}$

13 Add X to $Temp_1$

14 else if $X.\sup port_{(DB \cup db)} \geq \min\_PL_{(DB \cup db)}$ then

15 Add X to $PL_{(DB \cup db)}^{k}$

16 For each $Y \in Temp\_newC_k$ do

17 if $Y.\sup port_{(DB \cup db)} \geq \min\_\sup_{db}$ Then

18 Add Y to $Temp\_scanDB(L_{DB \cup db}^{k})$

19 Add Y to $Temp_1(L_{DB \cup db}^{k})$

รูปที่ 2.11 ขั้นตอนวิธีปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย กรณี $k \geq 2$

```
20      else if Y.sup port_(DB∪db) ≥ min_PL_DB∪db or Y.sup port_(DB∪db) ≥ min_PL_DB Then
21              Add Y to Temp_scanDB(PL^k_DB∪db)
22              Add Y to Temp_1(PL^k_DB∪db)
23      clear Temp_newC_k
24      if Temp_1 ≠ φ Then
25          Y ∈ Temp_1
26          C^{k+1}_DB∪db new = gen_newcandidate(Y)
27          clear Temp_1
28          Add C^{k+1}_DB∪db new to Temp_newC_k
29  k = k +
30  if Temp_scanDB ≠ φ Then Find_SupportcountDB
```

รูปที่ 2.11(ต่อ) ขั้นตอนวิธีปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย กรณี $k \geq 2$

Algorithm *Find_SupportcountDB*
Input:
(1) $L^k_{DB\cup db} \in Temp\_scanDB$: Estimated frequent $k-itemset$
(2) $PL^k_{DB\cup db} \in Temp\_scanDB$: 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.sup port_DB∪db = W.sup port_DB + W.sup port_db
4           if W.sup port_DB∪db ≥ min_sup_DB∪db Then
5               Add W to L^k_DB∪db
6           else
7               if W.sup port_DB∪db ≥ min_PL_DB∪db Then
8                   Add W to PL^k_DB∪db
9       Clear Temp_scanDB
```

รูปที่ 2.12 การทำงานของฟังก์ชัน *Find_sup portcountDB*

2.2.5 ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น (Probability-based incremental association rule discovery) [6]

การเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น เป็นขั้นตอนวิธีที่ถูกพัฒนาขึ้นมาโดยอาศัยหลักการหาความน่าจะเป็นด้วยทฤษฎีเบอร์นูลลีในการทำนายไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง เรียกว่า expected frequent itemset ซึ่งไอเท็มดังกล่าวจะช่วยในการค้นหาไอเท็มเซตที่เกิดบ่อยเมื่อมีฐานข้อมูลใหม่เพิ่มเข้ามา โดยจะนำเอาไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ไปช่วยในกระบวนการปรับปรุงค่าสนับสนุนและการสร้างไอเท็มเซตคู่แข่งของฐานข้อมูลปรับปรุง ระหว่างการทำงานแต่ละรอบ k หากมีไอเท็มใดที่มีโอกาสที่จะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง จะถูกเก็บไว้ในตัวแปรชื่อ *Temp_scanDB* และเมื่อประมวลผลเสร็จครบทุกรอบแล้ว ไอเท็มเซตที่อยู่ในตัวแปร *Temp_scanDB* จะถูกนำไปสแกนในฐานข้อมูลเดิม เพื่อปรับปรุงค่าสนับสนุน แล้วนำมาตรวจสอบว่าเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงหรือไม่ เป็นลำดับต่อไป

ด้วยหลักการดังกล่าวของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นซึ่งมีการสแกนฐานข้อมูลเดิมเพียงครั้งเดียว เป็นข้อดีที่ช่วยลดจำนวนการสแกนฐานข้อมูลเดิมเมื่อเปรียบเทียบกับขั้นตอนวิธี เอฟยูพี และมีการเก็บไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงน้อยลงเมื่อเทียบกับ ขั้นตอนวิธี เนกาทีฟบอร์เดอร์

การทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นซึ่งอาศัยหลักความน่าจะเป็นของเบอร์นูลลีมาประยุกต์ใช้ในการทำนายไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงแบ่งกระบวนการทำงานออกเป็น 2 กระบวนการหลัก ได้แก่ 1) กระบวนการหาไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงในฐานข้อมูลเดิม 2) กระบวนการปรับปรุงค่าสนับสนุนและค้นหาไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง โดยทั้งสองกระบวนการนี้ มีขั้นตอนหลักที่ต้องดำเนินการเหมือนกันคือการคำนวณค่าความน่าจะเป็นของไอเท็มที่คาดว่ามีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งมีการคำนวณดังนี้

$$P(X \geq k) = 1 - \sum_{x=0}^{k-1} \binom{n+m}{x} . p^x (1-p)^{m+n-x} \qquad (2.7)$$

เมื่อ

$P(X \geq k)$ หมายถึง ความน่าจะเป็นที่ไอเท็มเซตจะมีค่าสนับสนุน X ที่มากกว่าหรือเท่ากับค่าสนับสนุน k

k หมายถึง ค่าสนับสนุนที่น้อยที่สุดที่จะทำให้ไอเท็มเซตนั้นกลายเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

n หมายถึง ขนาดหรือจำนวนรายการธุรกรรมของฐานข้อมูลเดิม

m หมายถึง ขนาดหรือจำนวนรายการธุรกรรมของฐานข้อมูลใหม่

p หมายถึง ความน่าจะเป็นของการเกิดไอเท็มเซตที่พิจารณาในฐานข้อมูลซึ่งคำนวณจากจำนวนรายการธุรกรรมทั้งหมดของฐานข้อมูลเดิมที่ปรากฏไอเท็มเซตที่พิจารณาหารด้วยจำนวนรายการธุรกรรมทั้งหมดของฐานข้อมูลเดิม

ตัวอย่าง กำหนดให้มีจำนวนฐานข้อมูลเดิม $n=10$ รายการธุรกรรม และจำนวนข้อมูลใหม่ $n=5$ รายการธุรกรรม กำหนดให้มีค่าสนับสนุนขั้นต่ำ 40% และในฐานข้อมูลเก่ามีไอเท็ม A ปรากฏอยู่จำนวน 2 รายการธุรกรรม

จากตัวอย่าง ค่า $n=5$ คือค่าสนับสนุนที่น้อยที่สุดที่ไอเท็มเซตนั้นจะกลายเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง คำนวณจาก $k=(n+m)\times 40\%=6$ ส่วนค่า p คำนวณจากจำนวนรายการธุรกรรมทั้งหมดของฐานข้อมูลเดิมซึ่งมีค่าเท่ากับ 10 ดังนั้น $p=2\div 10=0.2$ จากนั้นจึงนำค่า k และ p ที่ได้มาคำนวณหาความน่าจะเป็นที่ไอเท็ม A จะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง โดยใช้สมการที่ 2.7 คำนวณได้ดังนี้

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

จากนั้นจะนำค่าความน่าจะเป็นที่คำนวณได้มาเปรียบเทียบกับค่า $\mathrm{Pr}ob_{PL}$ ซึ่งเป็นค่าทดสอบอีกค่าหนึ่งที่ผู้ใช้กำหนด เช่น กำหนดให้ค่า $\mathrm{Pr}ob_{PL}=0.1$ จะเห็นได้ว่า ค่าความน่าจะเป็นของไอเท็ม A ซึ่งมีค่าเท่ากับ 0.06 ซึ่งมีค่าน้อยกว่า $\mathrm{Pr}ob_{PL}$ ซึ่งมีค่าเท่ากับ 0.1 ดังนั้น ไอเท็ม A จะไม่จัดว่าเป็นไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงและจะถูกตัดทิ้งไป ไม่ถูกนำไปใช้ประมวลผลในรอบถัดไป

รายละเอียดขั้นตอนกระบวนการทำงานเพื่อค้นหาไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมและกระบวนการปรับปรุงและค้นหาไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่ มีรายละเอียดดังนี้

1. การค้นหาไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม

1.1 สแกนฐานข้อมูลเดิมเพื่อหาค่าสนับสนุนของไอเท็มเซตทุกตัว

1.2 คำนวณหาค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยให้แก่ไอเท็มเซตทุกตัว ด้วยการคำนวณจากสมการที่ 2.7

1.3 หาไอเท็มเซตที่เกิดบ่อย โดยพิจารณาจากค่าสนับสนุนของไอเท็มเซตที่จะต้องมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ min_sup ที่ผู้ใช้กำหนด

1.4 ไอเท็มเซตตัวใดที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำ min_sup และมีค่าความน่าจะเป็นที่คาดว่าไอเท็มนั้นจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงมากกว่าหรือเท่ากับค่า $\Pr ob_{PL}$ จะถูกเรียกว่า ไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย (Expected frequent itemset)

1.5 คำนวณหาค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม แทนค่าด้วยสัญลักษณ์ ρ^{DB} ด้วยการหาค่าสนับสนุนที่มีค่าน้อยที่สุดจากไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยทั้งหมด

1.6 ตั้งแต่รอบที่ $k \geq 2$ ขึ้นไป ให้ดำเนินการสร้างไอเท็มเซตคู่แข่งด้วยการเชื่อมไอเท็มเซตระหว่าง $(F_{k-1}^{DB} \cup EF_{k-1}^{DB}) * (F_{k-1}^{DB} \cup EF_{k-1}^{DB})$

1.7 จากนั้นทำตามขั้นตอนที่ 1.1 – 1.6 จนกว่าจะไม่สามารถสร้างไอเท็มเซตคู่แข่งได้อีก

2. การปรับปรุงและค้นหาไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่

2.1 สแกนฐานข้อมูลใหม่เพื่อหาค่าสนับสนุนของไอเท็มเซตทุกตัว

2.2 ปรับปรุง 1 – ไอเท็มเซต ที่เป็นไอเท็มเซตที่เกิดบ่อย F_1^{DB} และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย 1 – ไอเท็มเซต ของฐานข้อมูลเดิม EF_1^{DB}

2.3 คำนวณหาค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยให้แก่ไอเท็มทุกตัว ด้วยการคำนวณจากสมการที่ 2.7

2.4 หาไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง F_1^{UD} โดยพิจารณาจากค่าสนับสนุนของไอเท็มเซตที่จะต้องมีค่ามากกว่าค่าสนับสนุนขั้นต่ำ min_sup ที่ผู้ใช้กำหนด

2.5 ไอเท็มเซตตัวใดที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำ min_sup และมีค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยมากกว่าค่า $\Pr ob_{PL}$ จะถูกเรียกว่า ไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง EF_1^{UD}

2.6 คำนวณหาค่าคาดหวังที่น้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง แทนค่าด้วยสัญลักษณ์ ρ^{UD} ด้วยการหาค่าสนับสนุนที่มีค่าน้อยที่สุดจากไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยทั้งหมด

2.7 ตั้งแต่รอบที่ $k \geq 2$ ขึ้นไป ให้ดำเนินการสร้างไอเท็มเซตคู่แข่งด้วยการเชื่อมไอเท็มเซตโดย

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

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

2.8 นำไอเท็มเซตคู่แข่งที่ได้มาสแกนในฐานข้อมูลใหม่เพื่อคำนวณหาค่าสนับสนุน

2.9 ปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เป็นสมาชิกของ F_k^{DB} และ EF_k^{DB} แล้วคำนวณหาค่าความน่าจะเป็นของไอเท็มเซต เช่นเดียวกับข้อ 2.3

2.10 หาไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง F_k^{UD} โดยพิจารณาจากค่าสนับสนุนของไอเท็มเซตซึ่งจะต้องมีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำ min_sup ที่ผู้ใช้กำหนด

2.11 ไอเท็มเซตตัวใดที่มีค่าสนับสนุนน้อยกว่าค่าสนับสนุนขั้นต่ำ min_sup และมีค่าความน่าจะเป็นของไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยมากกว่าค่า $\mathrm{Pr}ob_{PL}$ จะถูกเรียกว่า ไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง EF_k^{UD}

2.12 ไอเท็มเซตคู่แข่งตัวใดที่ไม่ได้เป็นสมาชิกของ F_k^{DB} และ EF_k^{DB} ให้นำค่าสนับสนุนของไอเท็มเซตคู่แข่งนั้นบวกด้วยค่า $\rho^{DB} - 1$ แล้วนำไปทดสอบกับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด หากมีค่ามากกว่า หมายความว่าไอเท็มเซตนั้นมีโอกาสที่จะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง ดังนั้นไอเท็มเซตนี้จะถูกเก็บไว้ในตัวแปร *Temp_scanDB* เพื่อนำไปใช้ในการสแกนฐานข้อมูลเดิมอีกครั้งหนึ่ง หลังจากที่ได้ไอเท็มเซตที่เกิดบ่อยและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยครบทุกตัวแล้ว

2.13 ทำซ้ำตั้งแต่ข้อ 2.7 – 2.12 จนกว่าจะไม่สามารถสร้างไอเท็มเซตคู่แข่งได้อีก

2.14 นำไอเท็มเซตที่ถูกเก็บไว้ในตัวแปร *Temp_scanDB* ไปสแกนในฐานข้อมูลเดิมเพื่อปรับปรุงด้วยค่าสนับสนุน จากนั้นจึงหาไอเท็มเซตที่เกิดบ่อยและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย เช่นเดียวกับข้อ 2.10 และ 2.11

ข้อดีของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

1. สามารถลดจำนวนครั้งในการสแกนฐานข้อมูลได้ โดยขั้นตอนวิธีนี้จะมีการสแกนฐานข้อมูลเดิมเพียงครั้งเดียว

2. มีการใช้หลักความน่าจะเป็นในการทำนายไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย ทำให้มีจำนวนไอเท็มเซตที่ถูกเก็บไว้ก่อนการประมวลผลรอบถัดไปน้อยกว่าเมื่อเทียบกับขั้นตอนวิธี เนกาทีฟบอร์เดอร์

ข้อเสียของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

1. ในการคำนวณหาความน่าจะเป็นจะมีปัญหาในการคำนวณค่าแฟคทอเรียล ในกรณีที่จำนวนเต็มมีค่ามาก

2. มีการคำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัว ซึ่งไอเท็มเซตหลายตัวไม่จำเป็นที่จะต้องคำนวณค่าความน่าจะเป็นเนื่องจากเป็นไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อยอยู่แล้ว หรือไอเท็มเซตบางตัวมีค่าสนับสนุนน้อยมากจนไม่สามารถเป็นไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงได้

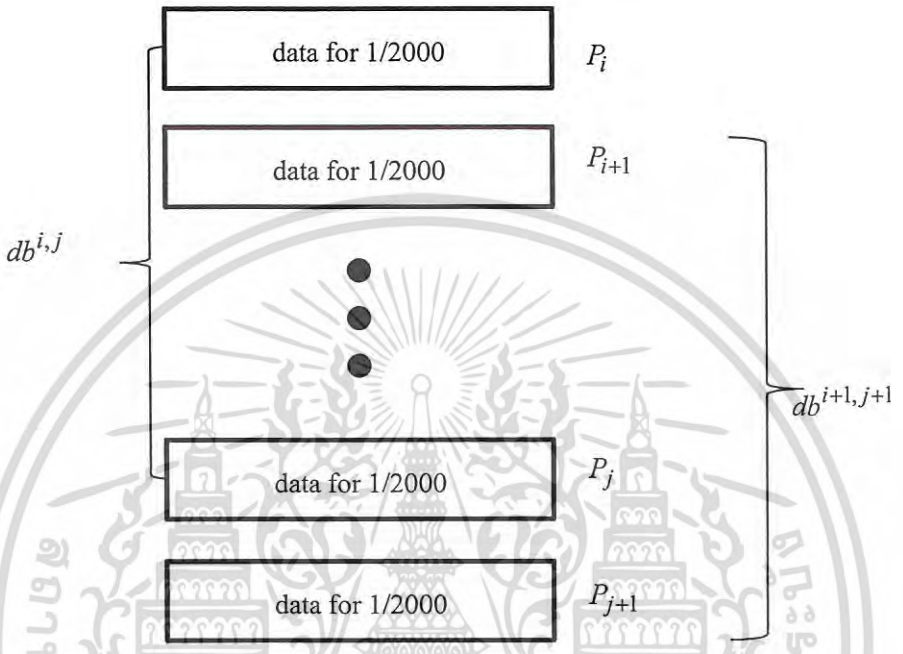
3. ในการคำนวณหาความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยจำเป็นต้องทราบขนาดที่แน่นอนของฐานข้อมูลใหม่หรือจำนวนรายการธุรกรรมในฐานข้อมูลใหม่ที่จะถูกเพิ่มเข้ามาในฐานข้อมูลเดิม ซึ่งเป็นไปได้ว่าบางครั้ง จำนวนรายการธุรกรรมใหม่ที่ถูกเพิ่มเข้ามาในแต่ละครั้งอาจจะมีจำนวนไม่เท่ากัน หรือบางครั้งอาจจะไม่ทราบจำนวนรายการธุรกรรมของฐานข้อมูลใหม่ ซึ่งในกรณีหลังจะทำให้ไม่สามารถคำนวณค่าความน่าจะเป็นได้

2.2.6 ขั้นตอนวิธี เอสดับบลิวเอฟ [13]

โดยธรรมชาติของข้อมูล จะมีข้อมูลใหม่เพิ่มเข้ามาเรื่อยๆ ในขณะเดียวกันก็จะมีข้อมูลเก่าที่ผู้ใช้เห็นว่าไม่จำเป็นหรือไม่สำคัญถูกลบทิ้งไป และการจัดเก็บข้อมูลส่วนใหญ่จะมีเวลาเป็นตัวกำกับข้อมูลเพื่อระบุว่าเป็นข้อมูลในช่วงเวลาใด

เอสดับบลิวเอฟ (Sliding Window Filtering) เป็นขั้นตอนวิธีที่เสนอแนวคิดการเพิ่มขยายกฎความสัมพันธ์ โดยมีหลักการคือแบ่งรายการธุรกรรมในฐานข้อมูลออกเป็นส่วนๆ ที่เรียกว่า พาร์ทิชันหรือ *partition* (P_i) มีการจัดเก็บข้อมูลตามช่วงเวลา เช่น 1/2000 หมายถึง ข้อมูลรายการธุรกรรมที่ถูกจัดเก็บในเดือนมกราคม ปี 2000 หรือ 2/2000 หมายถึง ข้อมูลรายการธุรกรรมที่ถูกจัดเก็บในเดือนกุมภาพันธ์ ปี 2000 เป็นต้น แสดงดังรูปที่ 2.13 จากนั้น เอสดับบลิวเอฟ จะทำการประมวลผลในทีละส่วนข้อมูลที่ถูกแบ่งที่เรียกว่าการประมวลผลเป็นเฟส (Phase of processing) โดยจะมีค่าสนับสนุนขั้นต่ำของแต่ละส่วนข้อมูลที่ถูกแบ่งที่เรียกว่าตัวกรอง (Filtering threshold)

เป็นค่าที่ใช้ทดสอบเพื่อคำนวณหา C_k ในแต่ละส่วนข้อมูลที่ถูกแบ่ง และเก็บค่าที่ได้ไว้ในส่วนถัดไป จากนั้น เอสดับบลิวเอฟ ได้นำหลักการของเทคนิคการลดการสแกน (Scan reduction technique) เข้ามาช่วยในการลดการทำงานของซีพียู



รูปที่ 2.13 ตัวอย่างการเพิ่มขยายกฎความสัมพันธ์ที่แบ่งการจัดเก็บข้อมูลเป็นเวลา

ขั้นตอนวิธี เอสดับบลิวเอฟ ได้อาศัยแนวคิดในการสร้างไอเท็มเซตคู่แข่งที่แตกต่างจากขั้นตอนวิธี อะพริโอริ เพื่อลดจำนวนรอบของการสแกนฐานข้อมูลเดิม แนวคิดดังกล่าวคือจะสร้าง C_k เมื่อ $k \geq 3$ ด้วยการเชื่อมไอเท็มเซตระหว่าง $C'_{k-1} * C'_{k-1}$ และจะทำการเชื่อมไอเท็มเซตไปเรื่อยๆ จนกว่าจะไม่สามารถสร้าง C_k ได้ $(C_k = \phi)$ แล้วนำไปสแกนในฐานข้อมูลเพียงครั้งเดียว เพื่อหาไอเท็มเซตที่เกิดบ่อย ซึ่งแตกต่างจากขั้นตอนวิธี อะพริโอริ ที่คำนวณไอเท็มเซตคู่แข่งจาก $C_k = L_{k-1} * L_{k-1}$ แล้วนำไอเท็มเซตที่ได้กลับไปสแกนฐานข้อมูลหลายๆ รอบ

นอกจากนั้น ขั้นตอนวิธี เอสดับบลิวเอฟ ยังใช้ประโยชน์จากค่าสนับสนุนภายใน (Local minimum support) หรือ ค่า สนับสนุนขั้นต่ำของแต่ละส่วนข้อมูลที่ถูกแบ่งโดยนำค่าดังกล่าวมาช่วยตัดไอเท็มเซตคู่แข่งที่ไม่จำเป็นทิ้งไป เพื่อลดจำนวน C_k ซึ่งการกระทำดังกล่าว ทำให้ขั้นตอนวิธี เอสดับบลิวเอฟ สร้าง C_k ที่มีค่าใกล้เคียงกับ L_k ทั้งนี้ เอสดับบลิวเอฟ ได้แบ่งชนิดของ C_k ออกเป็น 2 ประเภท ดังนี้

1. $\alpha-candidate$ หมายถึงไอเท็มเซตที่เป็น C_k ทั้งในเฟสก่อนหน้าและเฟสปัจจุบัน ซึ่งไอเท็มใดที่เป็นคุณสมบัติเป็น $\alpha-candidate$ จะตรวจสอบความสามารถที่เป็น CF ได้จากค่า threshold หรือ $s\times\sum_{m=I.start,k}|P_m|$

2. $\beta-candidate$ หมายถึง ไอเท็มเซตที่เป็น C_k ในเฟสปัจจุบัน แต่ไม่เป็น C_k ในเฟสก่อนหน้า ซึ่งไอเท็มใดที่มีคุณสมบัติเป็น $\beta-candidate$ จะถูกตรวจสอบว่าเป็น CF หรือไม่ จากค่าตัวกรองหรือ $s\times|P_m|$

ความหมายและสัญลักษณ์ที่ใช้ในขั้นตอนวิธี เอสดับบลิวเอฟ มีรายละเอียดดังนี้

Δ^- หมายถึง รายการธุรกรรมที่ถูกลบออกจากฐานข้อมูลเดิม

Δ^+ หมายถึง รายการธุรกรรมใหม่ที่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิม

D หมายถึง ฐานข้อมูลเดิม

D' หมายถึง ฐานข้อมูลปรับปรุง นั่นคือ $D'=(D-\Delta^-)\cup\Delta^+$

D^- หมายถึง ข้อมูลของฐานข้อมูลเดิมที่ไม่มีการเปลี่ยนแปลง (หมายถึงส่วนนที่ประกอบด้วยข้อมูลเดิมที่ไม่ได้ถูกลบและเพิ่มเข้าใหม่) นั่นคือ $D'=(D-\Delta^-)$

P_k หมายถึง รายการธุรกรรมในฐานข้อมูลซึ่งถูกแบ่งส่วนออกเป็น *partition* ตั้งแต่ $P_1,P_2,...,P_k$

$|P_k|$ หมายถึง จำนวนรายการธุรกรรมใน P_k

$db^{i,j}$ หมายถึง รายการธุรกรรมในฐานข้อมูลเดิมที่เป็นสมาชิกของ P_i ถึง P_j

s หมายถึง ค่าสนับสนุนขั้นต่ำ min_sup ที่ผู้ใช้กำหนด

$|\Delta^-|$ หมายถึง จำนวนรายการธุรกรรมที่ถูกลบออกจากฐานข้อมูลเดิม

$|\Delta^+|$ หมายถึง จำนวนรายการธุรกรรมที่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิม

I หมายถึง ไอเท็ม I ใดๆ

$I.start$ หมายถึง ลำดับ *partition* ของไอเท็ม I ที่คำนวณได้

$I.count$ หมายถึง ค่าสนับสนุนของไอเท็ม I

CF หมายถึง ตัวแปร Cumulative filter ที่ใช้เก็บค่าไอเท็มเซตคู่แข่งที่มีคุณสมบัติเป็นไอเท็มเซตที่เกิดบ่อยของแต่ละ *partition*

ขั้นตอนวิธี เอสดับบลิวเอฟ จะแบ่งขั้นตอนการทำงานออกเป็น 2 ส่วนหลักได้แก่ 1) ส่วนการเตรียมข้อมูล (Processing procedure) เป็นส่วนที่ใช้ประมวลผลในฐานข้อมูลเดิม ซึ่งจะมีการสแกนฐานข้อมูลเดิมจำนวน 2 ครั้ง และ 2) ส่วนการปรับปรุงข้อมูล (Incremental procedure) เป็น

ส่วนที่ใช้ประมวลผลข้อมูลในฐานข้อมูลใหม่ เพื่อปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อย โดยการทำงานแต่ละส่วนมีรายละเอียดดังนี้

1. การทำงานในส่วนของการเตรียมข้อมูล

การทำงานในส่วนการเตรียมข้อมูลเป็นการประมวลผลข้อมูลในฐานข้อมูลเดิม มีรายละเอียดการทำงานดังนี้

1.1 แบ่งรายการธุรกรรมในฐานข้อมูลเดิมออกเป็นส่วนๆ เรียกว่าพาร์ทิชัน จำนวน n ส่วน

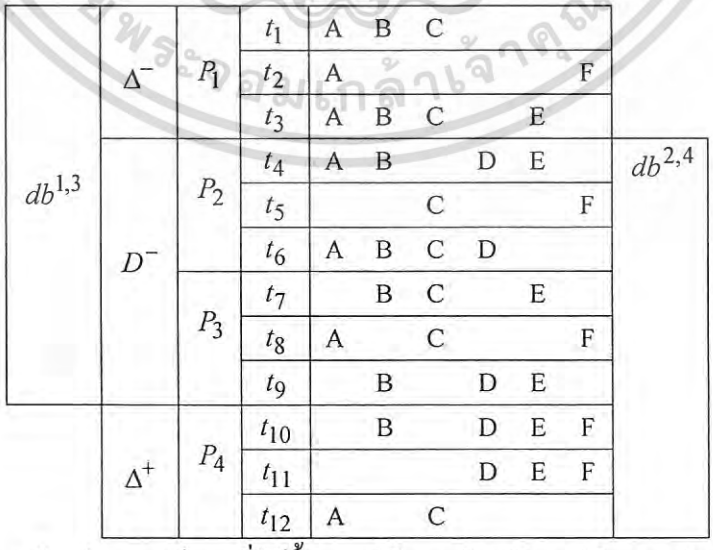
จากตัวอย่างในรูปที่ 2.14 มีข้อมูลจำนวน 12 รายการธุรกรรม โดย รายการธุรกรรมที่ $t_1 - t_9$ เป็นรายการธุรกรรมที่อยู่ในฐานข้อมูลเดิม เริ่มแรกจะทำการแบ่งข้อมูล $t_1 - t_9$ ออกเป็นพาร์ทิชัน ซึ่งจากตัวอย่างจะแบ่งออกเป็น 3 พาร์ทิชัน คือ P_1, P_2 และ P_3 ประกอบด้วยข้อมูล $t_1 - t_3$, $t_4 - t_6$ และ $t_7 - t_9$ ตามลำดับ และให้ $db^{1,3}$ คือรายการธุรกรรมที่อยู่ใน $P_1 - P_3$

1.2 ประมวลผลในแต่ละพาร์ทิชัน โดยเริ่มตั้งแต่พาร์ทิชันแรกจนถึงพาร์ทิชันสุดท้าย และมีการเก็บค่า *I.count* และ *I.start* ของแต่ละไอเท็มเซตในแต่ละพาร์ทิชัน โดยเริ่มต้นขั้นตอนวิธี เอสดับบลิวเอฟ จะคำนวณหา C_2 ก่อนเป็นอันดับแรก ทั้งนี้ขั้นตอนวิธี เอสดับบลิวเอฟ จะกำหนดให้ C_1 ทุกตัวเป็นสมาชิกของ L_1 ซึ่งจากตัวอย่างในรูปที่ 2.14 ค่า C_2 ที่คำนวณได้ใน P_1 ดังนี้

C_2 ที่หาได้จาก t_1 ได้แก่ $\{AB, AC, BC\}$

C_2 ที่หาได้จาก t_2 ได้แก่ $\{AF\}$

C_2 ที่หาได้จาก t_3 ได้แก่ $\{AB, AC, AE, BC, BE, CE\}$



รูปที่ 2.14 ตัวอย่างฐานข้อมูลที่ใช้ขั้นตอนวิธี เอสดับบลิวเอฟ ในการประมวลผล

ดังนั้น เมื่อนำ 2 – ไอเท็มเซต ที่คำนวณได้มารวมกันจะได้ C_2 ที่เป็นสมาชิกของ P_1 ได้แก่ $\{AB, AC, AE, AF, BC, BE, CE\}$ จากนั้นจะคำนวณหาค่า *I.count* เพื่อนำมาทดสอบว่า ไอเท็มเซตใดควรจะถูกจัดเก็บไว้ในตัวแปร *CF* โดยพิจารณาดังนี้

1.2.1 กรณี $I \in CF$

- ถ้า $I.count \geq s \times \sum_{m=I.start,k} |P_m|$ จะกำหนดให้ไอเท็ม I เป็นสมาชิกของตัวแปร *CF* คงเดิม

- ถ้า $I.count < s \times \sum_{m=I.start,k} |P_m|$ ให้ลบ ไอเท็ม I ออกจากตัวแปร *CF* ในกรณีนี้ รอบที่ $P_{k=1}$ จะไม่มีการตรวจสอบเนื่องจากแรกเริ่มจะกำหนดให้ $CF = \phi$ เสมอ และจะเริ่มตรวจสอบเมื่อ $P_{k\geq 2}$

1.2.2 กรณี $I \notin CF$ ให้เก็บหมายเลข *partition* ไว้ในตัวแปร *I.start* หรือ $I.start = k$ ถ้า $I.count \geq s \times |P_k|$ จะกำหนดให้ไอเท็ม I เป็นสมาชิกของ *CF*

เมื่อเสร็จสิ้นขั้นตอนที่ 1.2 ผลลัพธ์ที่ได้คือ C_2 ของทุก *partition* ที่เป็นสมาชิกของตัวแปร *CF* แสดงดังรูปที่ 2.15 โดยสัญลักษณ์ ○ จะหมายถึง C_2 ที่เป็นสมาชิกของ *CF*

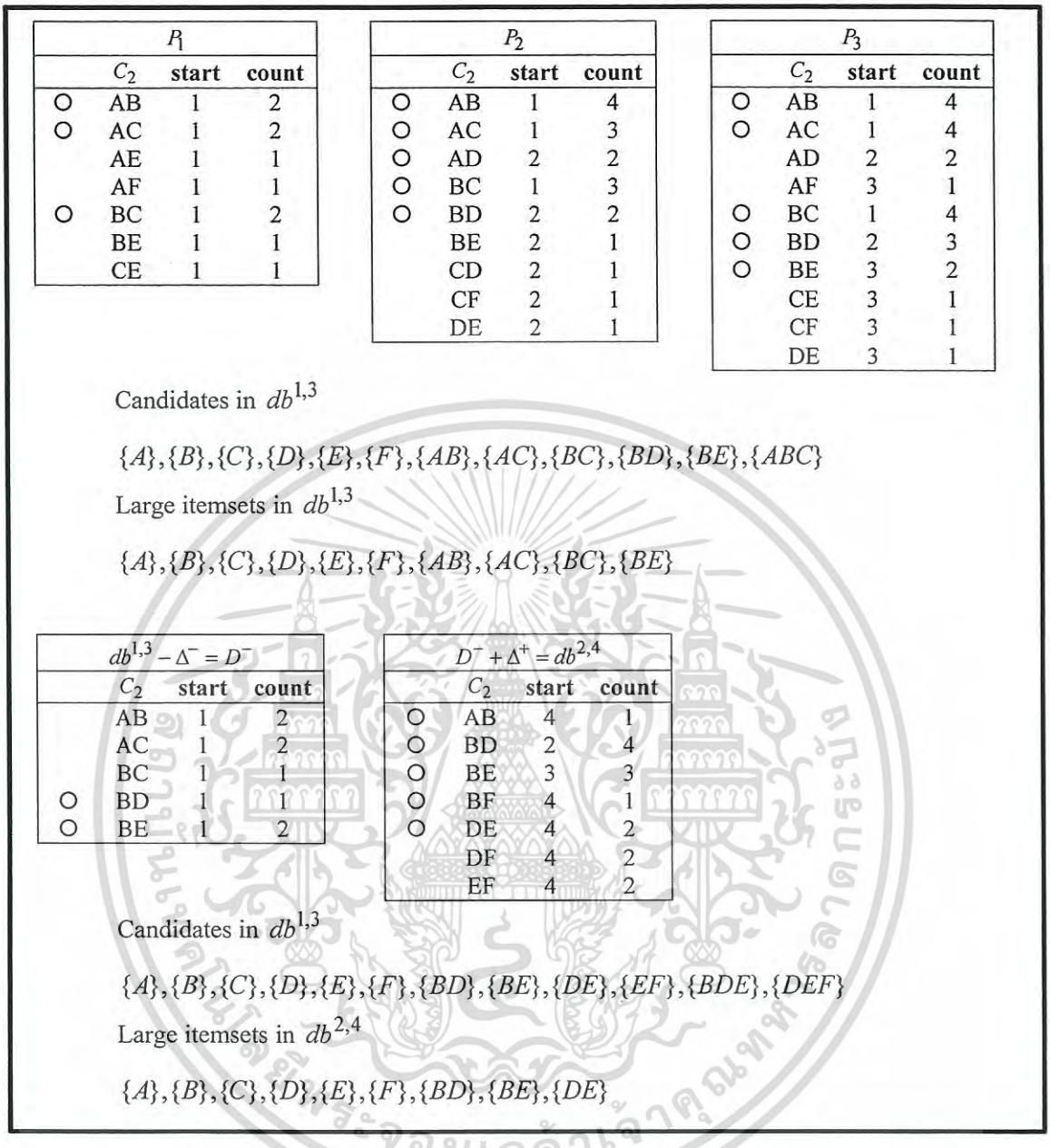
จากผลลัพธ์ที่ได้ในรูปที่ 2.15 แสดงผลการคำนวณหา *CF* ในแต่ละ *partition* โดย P_1 จะมี C_2 ที่เป็นสมาชิกของ *CF* ได้แก่ $\{AB, AC, BC\}$ ซึ่งจะนำผลลัพธ์ที่ได้ไปใช้ในการคำนวณใน P_2 เมื่อมีการคำนวณ P_2 เสร็จสิ้น จะได้ C_2 ที่เป็นสมาชิกของ *CF* ได้แก่ $\{AB, AC, AD, BC, BE\}$ ในรอบนี้จะพบว่ามี C_2 จำนวน 2 ตัวที่ถูกเพิ่มเข้าไปใน *CF* ได้แก่ *AD* และ *BE* ส่วนการคำนวณใน P_3 เมื่อเสร็จสิ้นการคำนวณ จะได้ *CF* คือ $\{AB, AC, BC, BD, BE\}$ จากนั้นจะนำ *CF* ที่ได้ไปคำนวณหา C_k เมื่อ $k \geq 3$ ในขั้นตอนถัดไป

1.3 สร้าง C_k จนกระทั่งไม่สามารถสร้าง C_k ได้ หรือ $C_k = \phi$ โดยในการสร้าง $C_{k\geq 3}$ ของขั้นตอนวิธีเอสดับบลิวเอฟจะสร้างจากการเชื่อมไอเท็มเซต ระหว่าง $C_{k-1} * C_{k-1}$ ซึ่งจากขั้นตอนที่ 1.2 ผลลัพธ์ที่ได้คือ $CF = \{AB, AC, BC, BD, BE\}$ ในขั้นตอนที่ 1.3 จะนำผลลัพธ์ดังกล่าวมาทำการเชื่อมไอเท็มเซตเพื่อให้ได้ $C_{k\geq 3}$ ดังนี้

การคำนวณหา C_3 คำนวณจากการนำ C_2 ที่เป็นสมาชิกของ *CF* มาทำการเชื่อมไอเท็มเซต ผลลัพธ์ที่ได้เป็นดังนี้

$$C_3 = \{ABC,\}$$

เมื่อได้ C_3 แล้ว ให้คำนวณหา C_4 ด้วยการเชื่อมไอเท็มเซต ระหว่าง C_3 แต่ด้วยผลลัพธ์ดังกล่าว มี C_3 เพียงจำนวน 1 ตัว ดังนั้นจึงไม่สร้าง C_4 ได้



P_1

	C_2	start	count
O	AB	1	2
O	AC	1	2
	AE	1	1
	AF	1	1
O	BC	1	2
	BE	1	1
	CE	1	1

P_2

	C_2	start	count
O	AB	1	4
O	AC	1	3
O	AD	2	2
O	BC	1	3
O	BD	2	2
	BE	2	1
	CD	2	1
	CF	2	1
	DE	2	1

P_3

	C_2	start	count
O	AB	1	4
O	AC	1	4
	AD	2	2
	AF	3	1
O	BC	1	4
O	BD	2	3
O	BE	3	2
	CE	3	1
	CF	3	1
	DE	3	1

Candidates in $db^{1,3}$

$\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{AB\},\{AC\},\{BC\},\{BD\},\{BE\},\{ABC\}$

Large itemsets in $db^{1,3}$

$\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{AB\},\{AC\},\{BC\},\{BE\}$

$db^{1,3} - \Delta^- = D^-$

	C_2	start	count
	AB	1	2
	AC	1	2
	BC	1	1
O	BD	1	1
O	BE	1	2

$D^- + \Delta^+ = db^{2,4}$

	C_2	start	count
O	AB	4	1
O	BD	2	4
O	BE	3	3
O	BF	4	1
O	DE	4	2
	DF	4	2
	EF	4	2

Candidates in $db^{1,3}$

$\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{BD\},\{BE\},\{DE\},\{EF\},\{BDE\},\{DEF\}$

Large itemsets in $db^{2,4}$

$\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{BD\},\{BE\},\{DE\}$

รูปที่ 2.15 ผลลัพธ์จากการประมวลผลในส่วนการเตรียมข้อมูลของขั้นตอนวิธีเอสดับบลิวเอฟ

1.4 นำ $C_{k\geq 3}$ ที่ได้ มาสแกนในฐานข้อมูลเดิมอีกครั้งเพื่อหาค่าสนับสนุน

1.5 นำค่าสนับสนุนของ C_k เมื่อ $C_k \in CF$ ที่คำนวณได้ในข้อ 1.2 และ 1.3 มาหาไอเท็มเซตที่เกิดบ่อย โดยเปรียบเทียบกับค่า min_sup นั่นคือ หาก $I.count \geq s \times |db^{1,n}|$ แสดงว่าไอเท็มนั้นเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมที่คำนวณได้จากขั้นตอนวิธี เอสดับบลิวเอฟ

จากตัวอย่าง ไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม ที่ได้จากการคำนวณของขั้นตอนวิธี เอสดับบลิวเอฟ ได้แก่ $\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{AB\},\{AC\},\{BC\},\{BE\}$ ทั้งนี้ $\{A,B,C,D,E,F\}$ เป็น $1-$ ไอเท็มเซต ที่ เอสดับบลิวเอฟ กำหนดไว้แต่เริ่มต้นว่า $1-$ ไอเท็มเซต ทุกตัวให้เป็นสมาชิกของไอเท็มเซตที่เกิดบ่อย

2. การทำงานในส่วนของการปรับปรุงข้อมูล

การทำงานในส่วนของการปรับปรุงข้อมูล เป็นการประมวลผลเมื่อมีรายการธุรกรรมใหม่เพิ่มเติมเข้ามา หรือมีบางรายการธุรกรรมเดิมที่ถูกลบทิ้งไป เพื่อปรับปรุงค่าไอเท็มเซตที่เกิดบ่อย จากตัวอย่างในรูปที่ 2.14 กำหนดให้ รายการธุรกรรม $t_1 - t_3$ อยู่ใน P_1 ถูกลบทิ้งไป (Δ^-) และ $t_{10} - t_{12}$ ที่อยู่ใน P_4 เป็นรายการธุรกรรมใหม่ที่ถูกเพิ่มเข้ามา (Δ^+) การทำงานในส่วนการปรับปรุงข้อมูลมีรายละเอียดดังนี้

2.1 นำค่า $C_2 \in CF$ ที่คำนวณได้จากขั้นตอนของ การเตรียมข้อมูล มาเก็บไว้ในตัวแปร CF

2.2 สแกน Δ^- เพื่อปรับปรุงค่าสนับสนุนให้กับ $2-$ ไอเท็มเซต เมื่อ $I \in CF$ โดยจะลดค่าสนับสนุนให้แก่ $I.count$ และปรับค่า $I.start$ ให้กับไอเท็มดังกล่าว โดย

- ให้ลดค่า $I.count$ ตามจำนวนค่าสนับสนุนที่คำนวณได้
- ปรับค่า $I.start$ ให้ $I.start = k+1$ เมื่อ k คือ หมายเลขกำกับพาร์ทิชัน

จากนั้นให้ไอเท็มมาตรวจสอบโดย ถ้า $I.count < s \times \sum_{m=I.start,k} |P_m|$ ให้ลบไอเท็มดังกล่าวออกจากตัวแปร CF

2.3 สแกน Δ^+ เพื่อปรับปรุงค่าสนับสนุนให้กับ $2-itemset$ โดยจะทำการเพิ่มค่าสนับสนุนให้กับ $I.count$ และปรับค่า $I.start$ ให้กับไอเท็มดังกล่าว โดย

2.3.1 กรณี $I \notin CF$

- ให้ค่า $I.count$ เท่ากับจำนวนค่าสนับสนุนที่คำนวณได้จาก Δ^+
- กำหนดให้ค่า $I.start = k$ เมื่อ k คือ หมายเลขกำกับพาร์ทิชัน

จากนั้นให้นำค่าสนับสนุนที่ได้ไปตรวจสอบกับค่า $s \times |P_k|$

- ถ้า $I.count \geq s \times |P_k|$ ให้นำไอเท็มดังกล่าวไปเก็บไว้ใน CF
- ถ้า $I.count < s \times |P_k|$ ไม่ต้องนำไอเท็มดังกล่าวไปเก็บไว้ใน CF

2.3.2 กรณี $I \in CF$

ให้ปรับปรุงค่าสนับสนุน โดยนำค่าสนับสนุนที่คำนวณได้จาก Δ^+ ไปรวมกับค่าสนับสนุนเดิมของไอเท็มนั้น จากนั้นให้นำค่าสนับสนุนที่ได้ไปตรวจสอบกับค่า $\min\_sup(s)$ โดย

- ถ้า $I.count \geq s \times \sum_{m=I.start,k} |P_m|$ ให้นำไอเท็มดังกล่าวไปเก็บไว้ใน CF คงเดิม

- ถ้า $I.count < s \times \sum_{m=I.start,k} |P_m|$ ให้ลบไอเท็มดังกล่าวออกจาก CF

2.4 คำนวณค่า $C_{k\geq 3}$ ด้วยการนำ $2-$ไอเท็มเซต ที่ $I \in CF$ มาทำการเชื่อมไอเท็มเซตเพื่อสร้าง C_k จนกระทั่งไม่สามารถสร้าง C_k ได้

2.5 การคำนวณหาไอเท็มเซตที่เกิดบ่อย ด้วยการนำ $C_{k\geq 2} \notin CF$ มาสแกนหาค่าสนับสนุนในฐานข้อมูลปรับปรุง $db^{i,j}$ อีกครั้ง แล้วทำการตรวจสอบ หากพบว่า $I.count \geq s \times |db^{i,j}|$ ให้เก็บค่าไอเท็มเซตดังกล่าวไว้ในไอเท็มเซตที่เกิดบ่อย

ผลลัพธ์จากการประมวลผลในส่วนของการปรับปรุงข้อมูลแสดงในรูปที่ 2.14 โดยการทำงานของ ขั้นตอนวิธี เอสดับบลิวเอฟ จะเริ่มจากการสแกน Δ^- เพื่อลดค่า $I.count$ และ $I.start$ ของ $2-$ไอเท็มเซต ที่ $I \in CF$ ที่คำนวณได้จาก การเตรียมข้อมูล โดย $2-itemset$ ที่ $I \in CF$ ได้แก่ $\{AB, AC, BC, BD, BE\}$ จะถูกนำไปสแกนใน Δ^- เพื่อหาค่าสนับสนุน และลดค่า $I.count$ และปรับค่า $I.start$ ให้เป็น 2 จากนั้นขั้นตอนวิธีตรวจสอบกับค่า min_sup(s) หากพบว่า $I.count \geq s \times \sum_{m=I.start,k} |P_m|$ ให้ลบไอเท็มเซตดังกล่าวออกจาก CF ซึ่งจากตัวอย่างจะพบว่า ค่า $I.count$ ของ $\{AB, AC, BC\}$ ไม่ผ่านเงื่อนไขดังกล่าว จึงทำให้ไอเท็มทั้ง 3 ตัวถูกลบออกจาก CF ไป ส่วน $\{BD, BE\}$ ยังคงเป็นสมาชิกของ CF คงเดิม โดย $BD \in \alpha\_candidate$ และ $BE \in \beta\_candidate$

จากนั้น ขั้นตอนถัดมา เอสดับบลิวเอฟ จะสแกน ในฐานข้อมลใหม่ Δ^+ เพื่อปรับปรุงค่า $I.count$ ที่ถูกเพิ่มเข้ามาจากฐานข้อมูลใหม่ โดย $2-$ไอเท็มเซต ใน Δ^+ ประกอบด้วย $\{AC, BD, BE, BF, DE, DF, EF\}$ โดย $\{BD, BE\}$เป็น $2-$ไอเท็มเซต ที่เป็นสมาชิกของ CF อยู่ก่อนหน้าแล้ว ดังนั้น เอสดับบลิวเอฟ จะปรับค่า $I.count$ เพิ่มให้กับ $\{BD, BE\}$ และนำค่า $I.count$ ที่ได้ปรับปรุงเรียบร้อยแล้วไปตรวจสอบ $I.count \geq s \times \sum_{m=I.start,k} |P_m|$ ซึ่งพบว่า $\{BD, BE\}$ ยังมีคุณสมบัติเป็น CF เช่นเดิม

ส่วน $\{AC, BF, DE, DF, EF\}$ เป็น $2-$ไอเท็มเซต ที่ไม่ได้เป็นสมาชิกของ CF ถูกนำไปสแกนและพบใน Δ^+ เมื่อคำนวณค่า $I.count$ และกำหนดให้ $I.start = 4$ เรียบร้อยแล้ว ขั้นตอนวิธี เอสดับบลิวเอฟ จะนำ $\{AC, BF, DE, DF, EF\}$ มาตรวจสอบ $I.count \geq s \times |P_m|$ ซึ่งพบว่าไอเท็มที่มีคุณสมบัติเป็น CF ได้แก่ $\{DE, DF, EF\}$

จากนั้นจึงนำเอา C_2 ที่มีคุณสมบัติเป็น CF ได้แก่ $\{BD, BEE, DE, DF, EF\}$ มาทำการเชื่อมไอเท็มเซตเพื่อสร้าง $C_{k\geq 3}$ ซึ่งจะได้ $C_3 = \{BDE, DEF\}$

สำหรับการหาไอเท็มเซตที่เกิดบ่อยในขั้นตอนสุดท้าย ขั้นตอนวิธี เอสดับบลิวเอฟ จะนำ $C_{k\geq 2} \in CF$ ซึ่งได้แก่ $\{BD, BE, DE, DF, EF, BDE, DEF\}$ มาสแกนในฐานข้อมูลปรับปรุง $db^{2,4}$ เพื่อตรวจสอบหาไอเท็มเซตที่มีคุณสมบัติเป็นไอเท็มเซตที่เกิดบ่อย $I.count \geq s \times |db^{2,4}|$ ซึ่งปรากฏว่า ไอเท็มเซตที่มีคุณสมบัติเป็นไอเท็มเซตที่เกิดบ่อยได้แก่ $\{BD, BE, DE\}$

ข้อดีของขั้นตอนวิธีเอสดับบลิวเอฟ

1. มีการใช้ประโยชน์จากความรู้เดิมในเฟสก่อนหน้าเพื่อลดจำนวนไอเท็มเซตคู่แข่ง ส่งผลให้หลักการเทคนิคการลดการสแกนสามารถช่วยเพิ่มประสิทธิภาพในการลดการทำงานของซีพียู

2. ปัญหาข้อมูลผิดปกติ (Skew data) มีผลกระทบต่อประสิทธิภาพของเอสดับบลิวเอฟ เนื่องจาก เอสดับบลิวเอฟ มีการสะสมไอเท็มเซตที่ได้จากการประมวลผลในรอบก่อนหน้าเพื่อกำจัดไอเท็มเซตคู่แข่งที่ไม่จำเป็นออกไปจากข้อมูลในเฟสก่อนหน้า

ข้อเสียของขั้นตอนวิธีเอสดับบลิวเอฟ

1. การกำหนดให้ C_1 ทุกตัวเป็น L_1 ตั้งแต่เริ่มแรก หากมีการคำนวณด้วยวิธีอื่น อาจจะมีไอเท็มเซตบางตัวที่ไม่เป็นสมาชิกของ L_1

2. ต้องมีการสแกนไอเท็มเซตเดิมในฐานข้อมูลเดิมซ้ำอีก 1 ครั้ง

2.2.7 ขั้นตอนวิธีซีไอเอสดับบลิวเอฟ และ เอฟไอเอสดับบลิวเอฟ [14]

แม้ว่าขั้นตอนวิธี เอสดับบลิวเอฟ จะทำงานได้อย่างมีประสิทธิภาพ ด้วยการคำนวณไอเท็มเซตคู่แข่งที่มีค่าใกล้เคียงกับไอเท็มเซตที่เกิดบ่อยก็ตาม แต่ขั้นตอนวิธี เอสดับบลิวเอฟ ใช้ประโยชน์จากไอเท็มเซตคู่แข่งที่จากการประมวลผลในรอบก่อนหน้าเพียงอย่างเดียวเท่านั้น ซีไอเอสดับบลิวเอฟ และ เอฟไอเอสดับบลิวเอฟ จึงเป็นขั้นตอนวิธีที่นำเสนอเพื่อเพิ่มขยายกฎความสัมพันธ์โดยเป็นการผสมผสานแนวคิดระหว่างขั้นตอนวิธี เอฟยูพี2 และ เอสดับบลิวเอฟ ซึ่งเป็นการเพิ่มประสิทธิภาพการทำงานให้ดียิ่งขึ้นด้วยการใช้ประโยชน์จากทั้งไอเท็มเซตคู่แข่งและไอเท็มเซตที่เกิดบ่อยที่คำนวณได้จากฐานข้อมูลเดิม นอกจากนี้ยังช่วยลดขั้นตอนการสแกนไอเท็มเก่าโดยจะสแกนเฉพาะไอเท็มใหม่เท่านั้น

ในส่วนของฐานข้อมูลเดิม เคไอเอสดับบลิวเอฟ จะทำการประมวลผลเพื่อหาไอเท็มเซตที่เกิดบ่อยที่เรียกว่า KI (Know Itemset) นั่นคือ เมื่อมีการเพิ่มข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม เคไอเอสดับบลิวเอฟ จะทำงานด้วยส่วนกระบวนการปรับปรุงข้อมูลซึ่งแบ่งขั้นตอนการทำงานออกเป็น 6 ส่วน ดังนี้

ขั้นตอนที่ 1-2: การปรับปรุงค่า CF

ขั้นตอนการปรับปรุงค่า CF จะมีวิธีการทำงานเช่นเดียวกับกระบวนการปรับปรุงข้อมูลของขั้นตอนวิธีเอสดับบลิวเอฟ ดังนี้

1. การทำงานในส่วนของรายการธุรกรรมที่ถูกลบออกไป (Δ^-)

สแกน Δ^- เพื่อปรับปรุงค่าสนับสนุนให้กับ $2-$ไอเท็มเซต ที่ $I \in CF$ โดยจะลดค่าสนับสนุนให้กับ $I.count$ และปรับค่า $I.start$ ให้กับไอเท็มเซตดังกล่าว โดย

- ให้ลดค่า $I.count$ ตามจำนวนค่าสนับสนุนที่คำนวณได้
- ปรับค่า $I.start$ ให้ค่า $I.start = k+1$ เมื่อ k คือหมายเลขพาร์ทิชัน (P_k)

จากนั้นให้นำแต่ละไอเท็มมาตรวจสอบ โดย ถ้า $I.count < s\times|D^-|$ ให้ลบไอเท็มดังกล่าวออกจาก CF

2. การทำงานในส่วนของรายการธุรกรรมที่เพิ่มเข้ามาใหม่ (Δ^+)

สแกน Δ^+ เพื่อปรับปรุงค่าสนับสนุนให้กับ $2-$ไอเท็มเซต ที่ $I \in CF$ โดยจะเพิ่มค่าสนับสนุนให้กับ $I.count$ และปรับค่า $I.start$ ให้กับไอเท็มเซตดังกล่าว โดย

2.1 กรณี $I \notin CF$

- ให้ค่า $I.count$ เท่ากับจำนวนค่าสนับสนุนที่คำนวณได้จาก Δ^+
- กำหนดให้ $I.start = k$ จากนั้นให้นำค่าสนับสนุนที่ได้ตรวจสอบกับ $s\times|P_k|$
- ถ้า $I.count \geq s\times|P_k|$ ให้นำไอเท็มเซตดังกล่าวเก็บไว้ใน CF
- ถ้า $I.count < s\times|P_k|$ ไม่ต้องนำไอเท็มเซตดังกล่าวเก็บไว้ในตัวแปร CF

2.2 กรณี $I \in CF$

ให้ปรับปรุงค่าสนับสนุน โดยนำค่าสนับสนุนที่คำนวณได้จาก Δ^+ ไปรวมกับค่าสนับสนุนเดิมของไอเท็มนั้นๆ จากนั้นให้นำค่าสนับสนุนที่คำนวณได้ไปตรวจสอบกับค่าสนับสนุนขั้นต่ำ โดย

- ถ้า $I.count \geq s\times\sum_{m=I.start,n}|P_m|$ ให้นำไอเท็มเซตดังกล่าวเก็บไว้ใน CF
- ถ้า $I.count < s\times\sum_{m=I.start,n}|P_m|$ ให้ลบไอเท็มดังกล่าวออกจาก CF

ขั้นตอนที่ 3-4: สแกน Δ^- และ Δ^+ เพื่อปรับปรุงค่า $I.count$ ให้กับ $I \in KI$ เพื่อค้นหาไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

ขั้นตอนนี้เป็นการปรับปรุงค่า $I.count$ ให้กับ $I \in KI$ ทั้งนี้ KI หมายถึง L_2 ที่ได้มาจากการประมวลผลในส่วนของฐานข้อมูลเดิม ซึ่งในขั้นตอนนี้จะแบ่งการประมวลผลออกเป็น 3 ส่วน

คือ ส่วนของการดึงผลลัพธ์จากการประมวลผลในรอบก่อนหน้านี้เข้ามาเก็บไว้ใน KI และส่วนของการปรับปรุงค่า KI ใน Δ^- และ Δ^+

1. ขั้นตอนการดึงผลลัพธ์จากการประมวลผลรอบก่อนหน้าเข้าเป็นไว้ใน KI

ในขั้นตอนนี้ จะเป็นการนำผลลัพธ์ที่ได้จากการประมวลผลรอบก่อนหน้า มาเก็บไว้ในตัวแปร KI ซึ่งแบ่งออกเป็น 2 กรณี คือ

1.1 ถ้าเป็นการดึงค่า $L_2^{m,n}$ มาเก็บไว้ในตัวแปร KI ในกรณีจะเรียกขั้นตอนวิธีเคไอเอสดับบลิวเอฟ ว่า เอฟไอเอสดับบลิวเอฟ (Frequent itemset SWF)

1.2 ถ้าเป็นการดึงค่า $C_2^{m,n}$ มาเก็บไว้ในตัวแปร KI ในกรณีจะเรียกขั้นตอนวิธีเคไอเอสดับบลิวเอฟ ว่า ซีไอเอสดับบลิวเอฟ (Candidate itemset SWF)

สำหรับการทำงานขั้นตอนถัดไป ขั้นตอนวิธีเอฟไอเอสดับบลิวเอฟและซีไอเอสดับบลิวเอฟ จะมีการทำงานเช่นเดียวกันทุกขั้นตอน

2. ขั้นตอนการปรับปรุงค่า KI ใน Δ^-

การปรับปรุงค่า $I.count$ เมื่อ $I \in KI$ ในกรณีที่มีการลบรายการธุรกรรมออกจากฐานข้อมูล จะสแกนเพื่อหาค่า $I.count$ ของไอเท็ม I ที่อยู่ใน Δ^- จากนั้นจะนำค่า $I.count$ ที่คำนวณได้ไปบวกกับค่า $I.count$ เดิม

3. ขั้นตอนการปรับปรุงค่า KI ใน Δ^+

การปรับปรุงค่า $I.count$ เมื่อ $I \in KI$ ในกรณีที่มีการเพิ่มรายการธุรกรรมเข้าไปในฐานข้อมูล จะสแกนเพื่อหาค่า $I.count$ ของไอเท็ม I ที่อยู่ใน Δ^+ จากนั้นจะนำค่า $I.count$ ที่คำนวณได้ไปบวกกับค่า $I.count$ เดิม เมื่อทำการปรับปรุงค่า $I.count$ เรียบร้อยแล้ว จะนำค่า $I.count$ มาตรวจสอบตามเงื่อนไข ดังนี้

ถ้า $I.count \geq s \times |db^{i,j}|$ แสดงว่าให้นำไอเท็ม I ไปเก็บไว้ในเซต FI

เมื่อ $|db^{i,j}|$ คือ จำนวนรายการธุรกรรมในฐานข้อมูลปรับปรุง

FI คือ ไอเท็มเซตที่เกิดบ่อย

ขั้นตอนที่ 5: สร้างไอเท็มเซตคู่แข่งตัวใหม่ขนาด $k-$ ไอเท็มเซต เมื่อ $k \geq 3$

ขั้นตอนนี้ จะเป็นการสร้างไอเท็มเซตคู่แข่งตัวใหม่ เมื่อ $k \geq 3$ ด้วยการเชื่อมไอเท็มเซต สำหรับการเชื่อมไอเท็มเซตของขั้นตอนวิธีนี้ จะแตกต่างจาก เอสดับบลิวเอฟ คือ เอสดับบลิวเอฟจะทำการเชื่อมไอเท็มเซตระหว่าง $C_{k-1} * C_{k-1}$ เพื่อหา C_k แต่สำหรับขั้นตอนวิธีเคไอเอสดับบลิวเอฟ จะสร้างไอเท็มเซตคู่แข่งตัวใหม่ที่เรียกว่า NC ด้วยการเชื่อมไอเท็มเซต จำนวน 3 คู่ ได้แก่

- $NC_k = NC_{k-1} * NC_{k-1}$ ตัวอย่าง $NC_3 = NC_2 * NC_2$
- $NC_k = NC_{k-1} * FI_{k-1}$ ตัวอย่าง $NC_3 = NC_2 * FI_2$
- $NC_k = FI_{k-1} * FI_{k-1}$ ตัวอย่าง $NC_3 = FI_2 * FI_2$

โดยขั้นตอนวิธี เคไอเอสดับบลิวเอฟ จะทำการเชื่อมไอเท็มเซตจนกว่าจะไม่สามารถสร้าง NC_k ในรอบถัดไปได้

สำหรับผลลัพธ์ที่ได้จากการเชื่อมไอเท็มเซตในกรณีที่ 3 ($NC_k = FI_{k-1} * FI_{k-1}$) หากพบว่ามีไอเท็มเซตใดที่เป็นมาชิกของ แล้ว KI ไอเท็มเซตดังกล่าวจะถูกลบออกจากเซตของ NC_3 ทันที เนื่องจากไอเท็มเซตนั้นๆ จะถูกนำไปสแกนในฐานข้อมูลปรับปรุงอีกครั้ง

ขั้นตอนที่ 6: การหาไอเท็มเซตที่เกิดบ่อยด้วยการสแกนฐานข้อมูลปรับปรุง

หลังจากเสร็จสิ้นการทำงานในขั้นตอนที่ 5 ผลลัพธ์ที่ได้คือไอเท็มเซตคู่แข่งใหม่ขนาด $k-$ ไอเท็มเซต (NC_k) เมื่อ $k \geq 2$ ซึ่งในขั้นตอนที่ 6 นี้ จะนำ $NC_{k\geq 2}$ ที่ได้ มาสแกนในฐานข้อมูลปรับปรุง เพื่อหาไอเท็มเซตที่เกิดบ่อย โดยพิจารณาว่า หากไอเท็ม I ใดๆ ที่ $I.count \geq s*|db^{i,j}|$ แล้ว ไอเท็มดังกล่าวจะถูกเก็บไว้ในเซตของ FI

ผลลัพธ์ที่ได้จากขั้นตอนที่ 6 คือ ไอเท็มเซตที่เกิดบ่อยที่ได้จากการคำนวณด้วยขั้นตอนวิธีเคไอเอสดับบลิวเอฟ

จากตัวอย่างของขั้นตอนวิธี เอสดับบลิวเอฟ ในรูปที่ 2.14 กำหนดให้ค่า min_sup=40% เช่นเดิม สามารถอธิบายการหาไอเท็มเซตที่เกิดบ่อยด้วยวิธีการของ เคไอเอสดับบลิวเอฟ ได้ดังนี้

<u>ขั้นตอนที่ 1-2</u>

นำไอเท็มเซต C_2 ที่คำนวณได้ในฐานข้อมูลเดิม $db^{1,3}$ มาเก็บไว้ใน CF_2 ได้แก่ $\{AB, AC, BC, BD, BE\}$ จากนั้นให้นำไอเท็มเซตดังกล่าวไปสแกนใน Δ^- (P_1) เพื่อปรับปรุงค่า $I.count$ และ $I.start$ ซึ่งผลลัพธ์ที่ได้คือ $\{AB, AC, BC\}$ จะถูกลบออกจาก CF_2 เนื่องจากค่า $I.count < s\times|D^-|$ ดังนั้นผลลัพธ์ที่ได้คือ จะเหลือไอเท็มเซตเพียง 2 ตัว ที่ยังคงเป็นสมาชิกของ CF_2 คือ $\{BD, BE\}$

จากนั้น จะสแกน Δ^+ เพื่อปรับปรุงค่า และ โดยแบ่งออกเป็น 2 กรณี คือ กรณี $I \in CF$ และ $I \notin CF$

- กรณี $I \in CF$ ซึ่งมีสมาชิก 2 ตัว คือ $\{BD, BE\}$ จะพบว่า เมื่อทำการปรับปรุงค่า $I.count$ แล้ว $\{BD, BE\}$ ยังคงเป็นสมาชิกของ CF_2 เนื่องจากค่า $I.count \geq s\times|db^{i,j}|$

- กรณี $I \notin CF$ ซึ่งหมายถึงไอเท็มเซตที่ไม่ได้เป็นสมาชิกของ CF_2 แต่ปรากฏอยู่ใน Δ^+ ซึ่งได้แก่ $\{AC, BF, DE, DF, EF\}$ จะกำหนดให้ $I.start = 4$ และค่า $I.count$ ที่คำนวณได้ คือ

(1),(1),(2),(2),(2) ตามลำดับ ดังนั้น ไอเท็มเซตใหม่ที่ถูกเพิ่มเข้าไปใน CF_2 ได้แก่ $\{DE, DF, EF\}$ เนื่องจาก $I.count \geq s \times |\Delta^+|$

เมื่อเสร็จสิ้นขั้นตอนที่ 1-2 แล้ว ผลลัพธ์ที่ได้คือ $CF_2 = \{BD, BE, DE, DF, EF\}$

ขั้นตอนที่ 3-4

นำค่า L_2 ที่คำนวณได้จากฐานข้อมูลเดิม $db^{1,3}$ มาเก็บไว้ใน KI ได้แก่ $\{AB, AC, BC, BE\}$ แล้วสแกนใน Δ^- เพื่อปรับปรุงค่า $I.count$ ด้วยการลบค่าสนับสนุนออกจากค่าเดิม จากนั้นจึงนำไปสแกนใน Δ^+ เพื่อปรับปรุงค่า $I.count$ ด้วยการบวกเพิ่มค่าสนับสนุนเข้าไป ผลลัพธ์ที่ได้คือ $\{BE\}$ จะถูกเก็บไว้ใน FI เนื่องจาก $I.count \geq s \times |\Delta^+|$ ส่วน $\{AB, AC, BC\}$ ยังคงเป็นสมาชิกของ KI เช่นเดิม

เมื่อเสร็จสิ้นขั้นตอนที่ 3-4 ผลลัพธ์ที่ได้คือ $FI = \{BE\}$ และ $KI = \{AB, AC, BC\}$

ขั้นตอนที่ 5

เป็นขั้นตอนการสร้างไอเท็มเซตคู่แข่งใหม่ NC_k จากการเชื่อมไอเท็มเซตทั้ง 3 แบบ โดยเริ่มแรกคือการคำนวณหาไอเท็มเซตที่เป็นสมาชิกของ NC_3 ซึ่งได้จากการเชื่อมไอเท็มเซต ระหว่าง $NC_2 * NC_2, NC_2 * FI_2$ และ $FI_2 * FI_2$ ซึ่งผลลัพธ์การสร้าง NC_3 จากขั้นตอนที่ 3 – 4 ปรากฏดังนี้

$$CF_2 = \{BD, DE, DF, EF\}$$

$$FI_2 = \{BF\}$$

$$KI_2 = \{AB, AC, BC\}$$

จากนั้นจะนำผลลัพธ์ข้างต้นมาคำนวณหา NC_2 ดังนี้

$$NC_2 = CF_2 - KI$$

$$\therefore NC_2 = \{BD, DE, DF, EF\}$$

จากนั้นให้นำค่า NC_2 และ FI_2 มาเชื่อมไอเท็มเซต จำนวน 3 แบบ ผลลัพธ์ที่ได้เป็นดังนี้

1) $NC_2 * NC_2 = \{BDE, DEF\}$
2) $NC_2 * FI_2 = \{BDE\}$
3) $FI_2 * FI_2 = \phi$

เมื่อนำผลการเชื่อมไอเท็มเซต ทั้ง 3 คู่มารวมกัน จะได้ผลลัพธ์ดังนี้

$$NC_3 = \{BDE, DEF\}$$

จากนั้นให้นำ $NC_3 = \{BDE, DEF\}$ และ $FI_3 = \phi$ มาทำการเชื่อมไอเท็มเซต เพื่อหา NC_4 ซึ่งผลลัพธ์คือ $NC_4 = \phi$ ส่งผลการทำงานของขั้นตอนที่ 5 ยุติการทำงานเนื่องจากไม่สามารถสร้าง $NC_{k \geq 4}$ ได้อีกต่อไป

ผลลัพธ์ที่ได้ในขั้นตอนที่ 5 คือ $NC_2 = \{BD, DE, DF, EF\}$ และ $NC_3 = \{BDE, DEF\}$

ขั้นตอนที่ 6

เป็นขั้นตอนการหาไอเท็มเซตที่เกิดบ่อย (FI_3) ด้วยการนำ $NC_{k\geq 2}$ ซึ่งก็คือ NC_2 และ NC_3 ที่คำนวณได้จากขั้นตอนที่ 5 มาสแกนในฐานข้อมูลปรับปรุง $db^{2,4}$ ผลลัพธ์ที่ได้คือ $\{BD, DE\}$ จะถูกนำมาเก็บไว้ใน FI เนื่องจาก $I.count \geq s \times |db^{2,4}|$ และ เมื่อนำผลที่ได้ไปรวมกับ FI เดิม คือ $\{BE\}$ แล้ว จะทำให้ได้ $FI = \{BD, BE\}$

ผลลัพธ์ที่ได้จากขั้นตอนที่ 6 คือ $FI = \{BD, BE\}$

ข้อดีของขั้นตอนวิธีเอฟไอเอสดับบลิวเอฟและซีไอเอสดับบลิวเอฟ

1. มีการเก็บผลลัพธ์การประมวลผลจากฐานข้อมูลเดิม (KI) มาใช้ร่วมกับฐานข้อมูลใหม่ที่ถูกเพิ่มเข้ามา ทำให้ช่วยลดจำนวน C_k ที่ใช้ในการสแกนฐานข้อมูล
2. มีการสแกนเฉพาะฐานข้อมูลที่เพิ่มเข้ามาใหม่เพียงครั้งเดียว

ข้อเสียของขั้นตอนวิธีเอฟไอเอสดับบลิวเอฟและซีไอเอสดับบลิวเอฟ

เช่นเดียวกับขั้นตอนวิธี เอสดับบลิวเอฟ คือ ขั้นตอนวิธี เอฟไอเอสดับบลิวเอฟ และ ซีไอเอสดับบลิวเอฟ ยังคงการกำหนดให้ C_1 ทุกตัวเป็น L_1 ตั้งแต่เริ่มแรก หากมีการคำนวณด้วยวิธีอื่นอาจจะมีไอเท็มเซตบางตัวที่ไม่เป็นสมาชิกของ L_1

2.2.8 ขั้นตอนวิธีดีบีทรีและพีโอทีเอฟพีทรี [15]

ขั้นตอนวิธี ดีบีทรี และ พีโอทีเอฟพีทรี (Potential frequent pattern tree) เป็นขั้นตอนวิธีที่ใช้หลักการพื้นฐานจากเอฟพีทรี (Frequent pattern tree: FP-tree) ในการค้นหากฎความสัมพันธ์ มีวัตถุประสงค์เพื่อลดจำนวนการสแกนฐานข้อมูลเดิมและไม่จำเป็นต้องสร้างไอเท็มเซตคู่แข่งเพื่อคำนวณหาไอเท็มเซตที่เกิดบ่อย โดยขั้นตอนวิธี ดีบีทรี จะเก็บสารสนเทศของฐานข้อมูลในรูปแบบเช่นเดียวกับเอฟพีทรีและไม่จำเป็นต้องกลับไปสแกนฐานข้อมูลเดิมซ้ำ ในกรณีที่มีการเพิ่มข้อมูลใหม่เข้าไป ส่วนขั้นตอนวิธีพีโอทีเอฟพีทรี จะใช้ไอเท็มเซตที่มีความเป็นไปได้ว่าจะเป็นไอเท็มเซตที่เกิดบ่อย (Possible frequent itemset) เพื่อลดจำนวนครั้งของการสแกนฐานข้อมูลเดิม ในกรณีที่ไอเท็มที่ไม่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมเปลี่ยนมาเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

กล่าวโดยสรุปคือ ทั้ง 2 ขั้นตอนวิธีนี้ มีวัตถุประสงค์หลักเพื่อหลีกเลี่ยงการนำไอเท็มเซตกลับไปสแกนฐานข้อมูลทั้งหมด และจะสร้างเอฟพีทรีอีกครั้งในกรณีที่ ไอเท็มที่ไม่เป็นไอเท็มเซตที่

เกิดบ่อยในฐานข้อมูลเดิมเปลี่ยนมาเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง การทำงานของ ดีบีทรี และ พีโอทีเอฟพีทรี อธิบายดังนี้

TID	Items Bought	Ordered Frequent Items
100	f,a,c,d,g,I,m,p	f,c,a,m,p
200	a,b,c,f,l,m,o	f,c,a,b,m
300	b,f,h,j,o	f,b
400	b,c,k,s,p	c,b,p
500	a,f,c,e,l,p,m,n	f,c,a,m,p

รูปที่ 2.16 ตัวอย่างฐานข้อมูลและการเรียงลำดับไอเท็มเซตที่เกิดบ่อย

2.2.8.1 ขั้นตอนวิธีดีบีทรี

การทำงานของ ดีบีทรี จะมีการสร้างโครงสร้างของไอเท็มเซตที่เรียกว่า ดีบีทรี เช่นเดียวกับเอฟพีทรี แต่จะมีการเก็บไอเท็มเซตทุกตัว ทั้งไอเท็มเซตที่เกิดบ่อยและไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย ซึ่งแตกต่างจากเอฟพีทรีที่มีการเก็บเฉพาะ ไอเท็มเซตที่เกิดบ่อยเท่านั้น รายละเอียดการสร้าง ดีบีทรี มีดังนี้

1) สแกนฐานข้อมูลครั้งแรก เพื่อหาไอเท็มเซตที่เกิดบ่อยขนาด 1 ไอเท็มเซต หรือ L_1 ซึ่งจากตัวอย่างในรูปที่ 2.16 กำหนดให้ min_sup=60% จะได้

$$L_1 = \{(f:4),(c:4),(a,3),(b,3),(m:3),(p:3)\}$$

ตัวเลขที่กำกับในแต่ละคู่คือค่าสนับสนุนของไอเท็ม นอกจากนี้ ดีบีทรี จะเก็บไอเท็มที่ไม่ใช่ L_1 ด้วย ซึ่งประกอบด้วย $\{d,e,g,h,i,j,k,l,n,o\}$

2) สแกนฐานข้อมูลครั้งที่สองเพื่อนำ L_1 และไอเท็มที่ไม่ใช่ L_1 ที่หาได้จากขั้นตอนที่ 1 มาเรียงลำดับตามค่าสนับสนุนจากมากไปหาน้อยในแต่ละรายการธุรกรรม ซึ่งผลลัพธ์ที่ได้จะปรากฏในตารางในรูปที่ 2.16 คอลัมน์ที่ 3 ซึ่งแสดงเฉพาะการเรียงลำดับของ L_1

3) จากนั้น นำ L_1 ที่ได้ในแต่ละรายการธุรกรรม (ในคอลัมน์ที่ 3) มาสร้างเป็นเอฟพีทรี โดยเริ่มจากรายการธุรกรรมรายการแรกที่มี L_1 คือ $\{f,c,a,m,p\}$ จะนำไอเท็มทั้งหมดมาสร้างเป็น tree โดยให้ f เป็นโหนดลูก (Child node) ที่เชื่อมกับโหนดที่อยู่บนสุดของต้นไม้ (Root node) และ c ให้เป็นโหนดลูกของ f ส่วน a เป็นโหนดลูกของ c ให้ m เป็นโหนดลูกของ a และ p เป็นโหนดลูกของ m ตามลำดับ และให้ค่าสนับสนุนของแต่ละไอเท็มมีค่าเท่ากับ 1 ส่วนไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยคือ d,g,i จะถูกนำมาสร้างเป็นโหนดลูกต่อจาก p และให้ค่าสนับสนุนเท่ากับ 1

ต่อมาในรายการธุรกรรมที่ 2 มี L_1 คือ $\{f,c,a,b,m\}$ เนื่องจาก มีการสร้างโหนดไว้แล้ว ให้คงไว้ซึ่งรูปแบบของโหนดตามเดิม แต่ให้เพิ่มจำนวนค่าสนับสนุนเข้าไปอีกไอเท็มละ 1 ส่วน b

เป็นโหนดลูกของ a ดังนั้นจึงให้แตกสาขาของ tree ที่โหนด a ออกมาอีกหนึ่งสาขา ส่วน m เป็นโหนดลูกของ b ตามลำดับ และให้ค่าสนับสนุนของ b และ m มีค่าเท่ากับ 1 ส่วนไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย คือ l,o จะถูกนำมาสร้างเป็นโหนดลูกต่อจาก m และให้ค่าสนับสนุนเท่ากับ 1

รายการธุรกรรมที่ 3 มี L_1 คือ $\{f,b\}$ แต่เนื่องจาก f เป็นโหนดลูกที่ต่อจาก root node เป็นโหนดแรก (นั่นคือไม่มีโหนดลูกก่อนหน้า f) ให้คง f ไว้เช่นเดิม แต่เพิ่มค่าให้ f อีก 1 จากนั้นให้สร้างแตกสาขาของ tree ของโหนด f เพื่อให้ b เป็นโหนดลูกของ f และให้ค่าสนับสนุนของ b เท่ากับ 1 ส่วนไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย คือ o,h,j จะถูกนำมาสร้างเป็นโหนดลูกต่อจาก b และให้ค่าสนับสนุนเท่ากับ 1

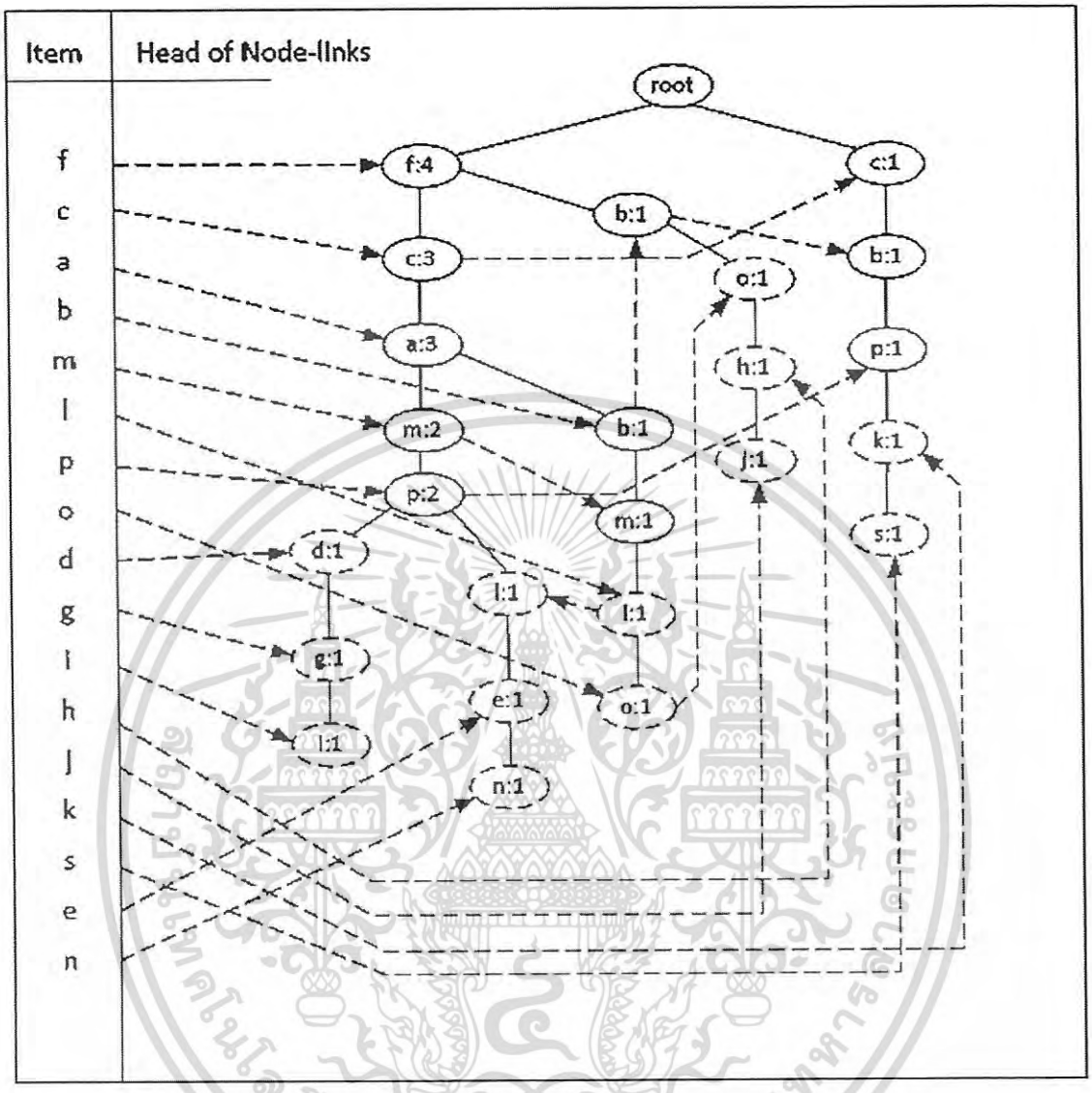
รายการธุรกรรมที่ 4 มี L_1 คือ $\{c,b,p\}$ เนื่องจากเป็นรูปแบบที่เกิดใหม่ใหม่ ซึ่งไม่มี c เป็นโหนดลูกต่อจาก root node ดังนั้นจึงจำเป็นต้องแตกสาขาของ tree จาก root node เพื่อให้ c เป็นโหนดลูกโดยตรง จากนั้นให้ b เป็นโหนดลูกของ c และให้ p เป็นโหนดลูกของ b ตามลำดับ และให้ค่าสนับสนุนของ $\{c,b,p\}$ มีค่าเท่ากับ 1 ส่วนไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย คือ k,s จะถูกนำมาสร้างเป็นโหนดลูกต่อจาก p และให้ค่าสนับสนุนเท่ากับ 1

รายการที่ 5 มี L_1 คือ $\{f,c,a,m,p\}$ เนื่องจากรูปแบบดังกล่าวมีอยู่แล้วซึ่งเป็นรูปแบบเดียวกับรายการธุรกรรมที่ 1 ดังนั้นให้คงรูปแบบเดิมไว้ แล้วเพิ่มค่าสนับสนุนให้กับ f,c,a,m,p อีกไอเท็มละ 1 ส่วนไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยคือ l,e,n จะถูกนำมาสร้างเป็นโหนดลูกต่อจาก m และให้ค่าสนับสนุนเท่ากับ

ผลลัพธ์จากการสร้างดีบีทรี จาก 5 รายการธุรกรรมข้างต้น แสดงได้ดังรูปที่ 2.17 โดยมีสัญลักษณ์อธิบายรูป ดังนี้

- โหนดที่เป็นวงกลมเส้นทึบ หมายถึง โหนดของไอเท็มที่เป็นไอเท็มเซตที่เกิดบ่อย
- โหนดที่เป็นวงกลมเส้นประ หมายถึง โหนดของไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย

ในส่วนของการหาไอเท็มเซตที่เกิดบ่อยทั้งหมดของ ดีบีทรี จะกระทำเช่นเดียวกับเอฟพีทรีคือ หาไอเท็มเซตที่เกิดบ่อยจากโหนดที่เป็นไอเท็มเซตที่เกิดบ่อย (โหนดที่เป็นเส้นทึบ) โดยจะเริ่มจากโหนดที่อยู่ระดับล่างสุด ซึ่งจากรูป โหนดที่อยู่ระดับล่างสุดคือ p ซึ่ง p เป็นโหนดลูกในระดับล่างสุดของรูปแบบ จำนวน 2 รูปแบบ ได้แก่ $\{f,c,a,m\}$ ซึ่งจะให้ค่าสนับสนุนสูงสุดเท่ากับ c ในรูปแบบนี้ คือ 2 ส่วนอีกรูปแบบ ได้แก่ $\{c,b\}$ จะกำหนดให้ค่าสนับสนุนสูงสุดเท่ากับ c ในรูปแบบนี้ คือ 1 ซึ่งแสดงให้เห็นตามตารางในรูปที่ 2.18 คอลัมน์ที่ 3 ของตารางในส่วน condition FP-tree ซึ่งพบว่ามีเพียงไอเท็มเดียวที่ปรากฏใน 2 รูปแบบ คือ c และเมื่อรวมค่าสนับสนุนแล้วจะได้เท่ากับ 3 และได้รูปแบบไอเท็มเซตที่เกิดบ่อย คือ cp:3



รูปที่ 2.17 ตัวอย่างการสร้างดีบีทรี

ไอเท็มที่อยู่ระดับล่างสุดลำดับถัดมาคือ m จากรูปที่ 2.17 จะปรากฏ m เป็นโหนดล่างสุดของ 2 รูปแบบ คือ $\{f,c,a\}$ และ $\{f,c,a,b\}$ ซึ่งมีค่าสนับสนุนเท่ากับ 2 และ 1 ตามลำดับ และเมื่อรวมค่าสนับสนุนของไอเท็มที่ปรากฏในทั้ง 2 รูปแบบแล้ว จะได้ $\{f,c,a\}$ ปรากฏในคอลัมน์ที่ 3 ของตารางในรูปที่ 2.18 ดังนั้น จะได้ รูปแบบของไอเท็มเซตที่เกิดบ่อย คือ $fca:3$

ไอเท็มที่อยู่ในระดับล่างสุดลำดับถัดมาคือ b จากรูปที่ 2.17 จะปรากฏ b เป็นโหนดล่างสุดของ 3 รูปแบบ ได้แก่ $\{f,c,a\}$, $\{f\}$ และ $\{c\}$ ซึ่งมีค่าสนับสนุนเท่ากับ 1 และเมื่อรวมค่าสนับสนุนของไอเท็มที่ปรากฏในทั้ง 2 รูปแบบ แล้วพบว่าไม่มีไอเท็มใดปรากฏอยู่ในทั้ง 3 รูปแบบ

จึงทำให้ค่าในคอลัมน์ที่ 3 ของตารางในรูปที่ 2.18 ปรากฏเป็น ϕ ดังนั้น รูปแบบของไอเท็มเซตที่เกิดบ่อยที่ได้คือ $b:3$

ส่วน a,c และ f เมื่อพิจารณาแล้วจะเห็นได้ว่าทั้ง 3 ไอเท็มเป็นสับเซตของ *fcam* ดังนั้น *fcam* และ *cp* จึงถือว่าเป็นไอเท็มเซตที่เกิดบ่อยที่มีขนาดสูงสุด (Maximal frequent itemset) ซึ่งสับเซตของ *fcam* และ *cp* จะเป็นไอเท็มเซตที่เกิดบ่อยด้วย

Item	Conditional Pattern Base	Conditional FP-tree
p	$\{(f:2,c:2,a:2,m:2),(c:1,b:1)\}$	$\{(c:3)\}\|p$
m	$\{(f:2,c:2,a:2),(f:1,c:1,a:1,b:1)\}$	$\{(f:3,c:3,a:3)\}\|m$
b	$\{(f:1,c:1,a:1,),(f:1),(c:1)\}$	ϕ
a	$\{(f:3,c:3)\}$	$\{(f:3,c:3)\}\|a$
c	$\{(f:3)\}$	$\{(f:3)\}\|c$
f	ϕ	ϕ

รูปที่ 2.18 ผลลัพธ์การประมวลผล conditional pattern base และ condition FP-tree

ในส่วนของการเพิ่มขยายกฎความสัมพันธ์ หรือการทำงานเมื่อมีรายการธุรกรรมใหม่เพิ่มเข้ามา เช่น มีรายการข้อมูลเพิ่มเข้ามา 2 รายการ ดังนี้

$TID = 600$ ประกอบด้วย $\{a,c,f,m,g,o,l\}$

$TID = 700$ ประกอบด้วย $\{f,b,a,c,l,m,o,n\}$

ดีบีทรี จะสแกนเฉพาะ 2 รายการธุรกรรมที่เพิ่มเข้ามาใหม่คือ $TID = 600$ และ $TID = 700$ จากนั้นจึงนำค่าสนับสนุนของไอเท็มเซตในรายการธุรกรรมใหม่ไปรวมกับค่าสนับสนุนของฐานข้อมูลเดิม จะได้ค่าสนับสนุนใหม่ ดังนี้

$(f:6),(c:6),(a:5),(b:4),(m:5),(l:4),(o:4)$

$(p:3),(g:2),(n:2),(j:1),(k:1),(s:1),(e:1),(n:1)$

หลังจากปรับปรุงค่าสนับสนุนของไอเท็มเซตแต่ละตัวแล้ว พบว่าค่าสนับสนุนที่เป็นค่าที่น้อยที่สุดในฐานข้อมูลปรับปรุง คือ 4 จะเห็นได้ว่า p ซึ่งเดิมเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่เมื่อการเพิ่มข้อมูลเข้ามา ทำให้ p เปลี่ยนเป็นไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งการทำงานของ ดีบีทรี ไม่จำเป็นต้องเปลี่ยนโครงสร้างของต้นไม้ทั้งหมด แต่จะปรับเปลี่ยนเฉพาะ 2 กรณี คือ 1) ไอเท็มที่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมเปลี่ยนเป็น

ไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง 2) ไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมเปลี่ยนเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

จากรูปที่ 2.17 จะเห็นได้ว่า p เป็นไอเท็มเซตที่เกิดบ่อย ส่วน l ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามา ส่งผลให้ p เปลี่ยนเป็นไอเท็มที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อย ในขณะที่ l เปลี่ยนเป็นไอเท็มเซตที่เกิดบ่อย ดังนั้น ดีบีทรี จะทำการเรียงลำดับไอเท็มเฉพาะ 2 ไอเท็มนี้ใหม่ โดยให้ l เป็นไอเท็มเซตที่เกิดบ่อย และเป็นโหนดลูกต่อจาก m แสดงดังรูปที่ 2.19 แล้วให้ p ซึ่งไม่ได้เป็นไอเท็มเซตที่เกิดบ่อย เปลี่ยนมาเป็นโหนดลูกต่อจาก l แทน จากนั้นให้หาไอเท็มเซตที่เกิดบ่อยโดยใช้หลักการเช่นเดียวกับเอฟพีทรีคือหาจากโหนดที่เป็นไอเท็มเซตที่เกิดบ่อย

ข้อดีของขั้นตอนวิธี ดีบีทรี

1. เมื่อมีรายการธุรกรรมใหม่เพิ่มเข้ามา ดีบีทรี ไม่จำเป็นต้องสแกนฐานข้อมูลเดิม แต่จะสแกนเฉพาะรายการที่ถูกเพิ่มเข้ามาใหม่เท่านั้น
2. ใช้เวลาในการหาไอเท็มเซตที่เกิดบ่อยน้อยเนื่องจากไม่มีการสร้างไอเท็มเซตคู่แข่ง

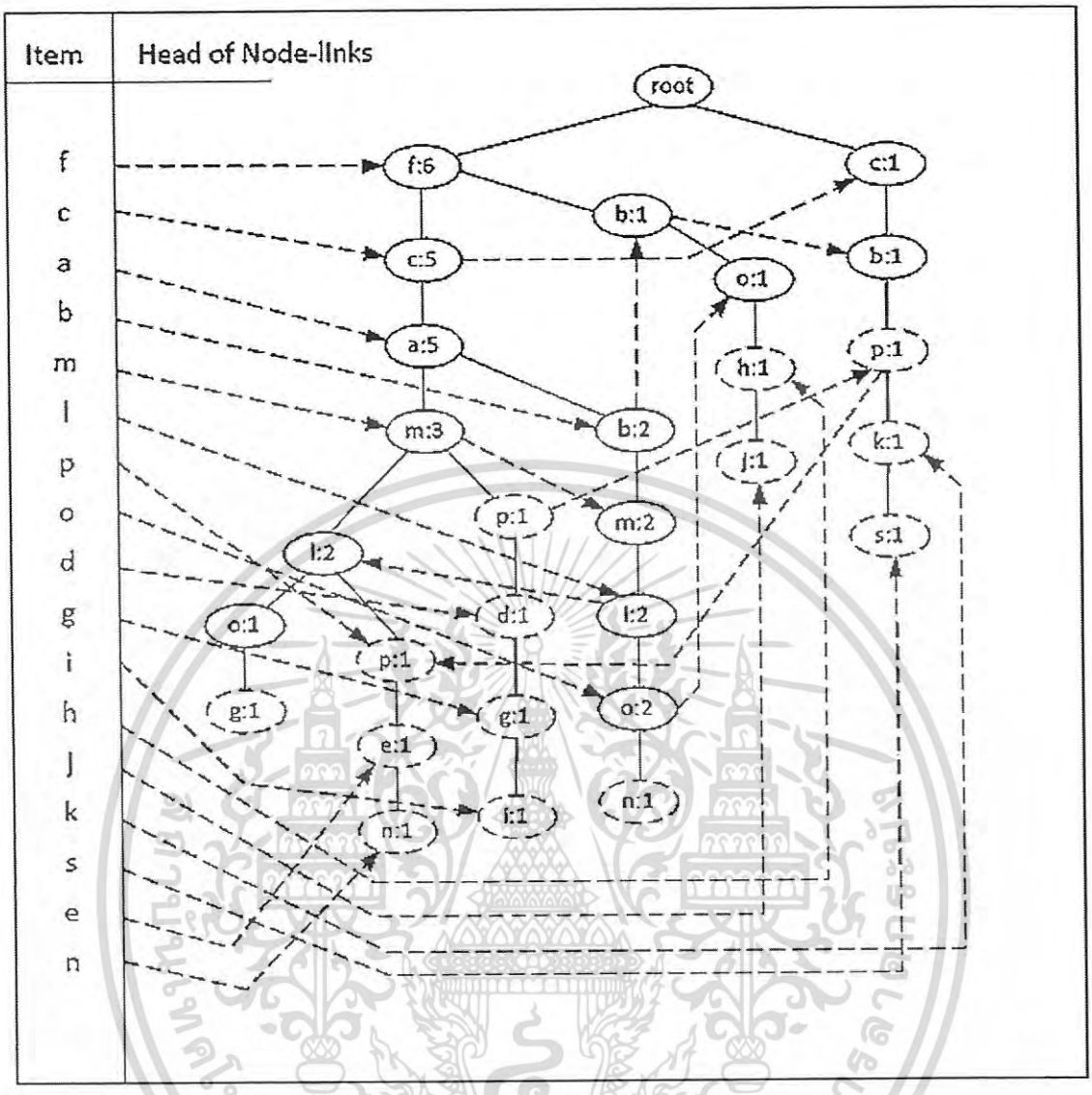
ข้อเสียของขั้นตอนวิธี ดีบีทรี

1. สิ้นเปลืองเนื้อที่ในการจัดเก็บไอเท็มเซต เนื่องจากดีบีทรีจะเก็บไอเท็มทั้งไอเท็มเซตที่เกิดบ่อยและไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย
2. มีขั้นตอนที่ยุ่งยากซับซ้อนในการสร้างดีบีทรี

2.2.8.2 ขั้นตอนวิธีพีโอทีเอฟพีทรี

พีโอทีเอฟพีทรี เป็นขั้นตอนวิธีที่มีการทำงานผสมผสานระหว่างเอฟพีทรี ซึ่งเก็บเฉพาะไอเท็มที่เป็นไอเท็มเซตที่เกิดบ่อย และดีบีทรี ซึ่งเก็บไอเท็มทุกตัวทั้งไอเท็มเซตที่เกิดบ่อยและไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย ทั้งนี้ พีโอทีเอฟพีทรี ได้นำเอาหลักการเบื้องต้นจากทั้ง 2 ขั้นตอนวิธีดังกล่าวมาประยุกต์ใช้โดยมีแนวคิดที่จะจัดเก็บไอเท็มที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงด้วย

หลักการเลือกเก็บไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิมแบ่งออกเป็น 2 ประเภทดังนี้



รูปที่ 2.19 ผลลัพธ์หลังจากเพิ่มรายการธุรกรรมใหม่เข้าไปในฐานข้อมูล ของขั้นตอนวิธีดีบีทรี

1) ไอเท็มที่ไม่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมแต่มีความน่าจะเป็นในระดับสูงที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ใช้สัญลักษณ์ P โดยไอเท็มดังกล่าวคำนวณได้จาก

$$t \le s \le average\text{minsup}\, port$$

เมื่อ t คือ ค่าที่ใช้กำหนดจำนวนรายการธุรกรรมที่คาดว่าจะเพิ่มเข้ามาใหม่

s คือ คือค่าสนับสนุนขั้นต่ำ

$average\text{minsup}\, port$ คือค่าสนับสนุนส่วนใหญ่ที่เกิดขึ้นในการประมวลผลรอบปัจจุบัน

2) ไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม และไม่มีโอกาสเป็นเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง ใช้สัญลักษณ์ M

พีโอทีเอฟพีทรี จะนำไอเท็มประเภท P เข้ามาสร้างใน พีโอทีเอฟพีทรี ด้วยเพื่อลดการสแกนฐานข้อมูลเดิมเมื่อมีการเพิ่มรายการธุรกรรมใหม่เข้ามา อย่างไรก็ตามหากไอเท็มประเภท M เปลี่ยนเป็นไอเท็มเซตที่เกิดบ่อยแล้ว พีโอทีเอฟพีทรี จะต้องสแกนฐานข้อมูลเดิมอีกครั้งเช่นเดียวกันกับเอฟพีทรี

ข้อดีของขั้นตอนวิธีพีโอทีเอฟพีทรี

1. ลดการสแกนฐานข้อมูลเดิมในกรณีที่ไอเท็มประเภท P เปลี่ยนมาเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง
2. ใช้เวลาในการหาไอเท็มเซตที่เกิดบ่อยน้อยลง เนื่องจากไม่มีการสร้างไอเท็มเซตคู่แข่ง
3. ใช้เนื้อที่ในการจัดเก็บไอเท็มเซตน้อยกว่า ดีบีทรี เพราะ พีโอทีเอฟพีทรี จะเก็บไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงเท่านั้น

ข้อเสียของขั้นตอนวิธีพีโอทีเอฟพีทรี

1. หากเกิดกรณีไอเท็มประเภท M เปลี่ยนมาเป็นไอเท็มเซตที่เกิดบ่อยแล้ว พีโอทีเอฟพีทรีจะต้องสแกนฐานข้อมูลเดิมอีกครั้งเช่นเดียวกับเอฟพีทรี

2.3 ทฤษฎีการหาความน่าจะเป็นแบบเบอร์นูลลี

ความน่าจะเป็นแบบเบอร์นูลลี เป็นทฤษฎีทางสถิติที่ใช้ในการพิจารณาการทดลอง ซึ่งผลการทดลองแต่ละครั้งมีผลเพียง 2 แบบเท่านั้นคือ ความสำเร็จ (Success) และล้มเหลว (Failure) [16] เช่น ในการโยนเหรียญ 1 อัน จะปรากฏผลได้ 2 อย่าง คือ ด้านหัวและด้านก้อย ซึ่งอาจกำหนดให้ ถ้าเหรียญออกหัว หมายถึงความสำเร็จ แต่ถ้าออกก้อยหมายถึงความล้มเหลว สำหรับตัวอย่างการทดลองโยนเหรียญนี้ ความน่าจะเป็นที่จะออกหัว เท่ากับ ½ และความน่าจะเป็นที่จะออกก้อยเท่ากับ ½ เช่นกัน นั่นหมายความว่า ความน่าจะเป็นที่จะเกิดความสำเร็จเท่ากับ p และความน่าจะเป็นที่จะเกิดความล้มเหลวเท่ากับ q โดย $p+q=1$ เสมอ

การค้นหากฎความสัมพันธ์ในงานวิจัยการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น [6] ได้นำทฤษฎีเบอร์นูลลีมาประยุกต์ใช้ โดยผลสำเร็จ p จะหมายถึงการที่ไอเท็มที่พิจารณาปรากฏอยู่ในรายการธุรกรรม ในขณะที่ผลล้มเหลว q หรือ $q=1-p$ จะหมายถึงการที่ไอเท็มที่พิจารณาไม่ปรากฏอยู่ในรายการธุรกรรม

เมื่อทำการเพิ่มรายการข้อมูลใหม่เข้าไปในฐานข้อมูลเดิม การคำนวณความน่าจะเป็นจะคำนวณด้วยสูตร ดังนี้

$$P(x)=\binom{n+m}{x}.p^{x}(1-p)^{m+n-x} \tag{2.8}$$

โดย P หมายถึงความน่าจะเป็น

$(n+m)$ หมายถึงขนาดของข้อมูลที่จะทำการทดลอง เมื่อ n คือจำนวนรายการธุรกรรมฐานของข้อมูลเดิม และ m คือจำนวนรายการธุรกรรมในฐานข้อมูลใหม่

x หมายถึงจำนวนครั้งที่การทดลองจะเกิดความสำเร็จ

p หมายถึงค่าความน่าจะเป็นที่การทดลองจะให้ผลสำเร็จ ซึ่งในงานวิจัยการค้นหากฎความสัมพันธ์จะหมายถึงจำนวนรายการธุรกรรมที่ปรากฏไอเท็มที่พิจารณา

$1-p$ หมายถึงค่าความน่าจะเป็นที่การทดลองจะให้ผลล้มเหลว ซึ่งในงานวิจัยการค้นหากฎความสัมพันธ์จะหมายถึงจำนวนรายการธุรกรรมที่ไม่ปรากฏไอเท็มที่พิจารณา

2.4 ทฤษฎีการประมาณค่าทวินามด้วยการแจกแจงปกติ

จากทฤษฎีบทลิมิตของเดอมัวร์และลาปลาซ (Demoivre-Laplace Limit Theorem) [16, 17] 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 $\sqrt{np(-p)}$ แล้วฟังก์ชั่นการแจกแจงของตัวแปรสุ่มมาตรฐานนี้ จะลู่เข้าสู่ฟังก์ชั่นการแจกแจงปกติมาตรฐานเมื่อ $n\to\infty$

ทฤษฎีบทที่ 1 ทฤษฎีบทลิมิตของเดอมัวร์และลาปลาซ (DeMoivre-Laplace Limit Theorem)

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

$$P(X=k)=\binom{n}{k}p^k q^{n-k} \cong \frac{1}{\sqrt{2\pi npq}} e^{\frac{(k-np)^2}{2npq}} \tag{2.9}$$

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 [18]

$$P(a \le X \le b) \approx P\left(\frac{a-\frac{1}{2}-np}{\sqrt{np(1-p)}} \le z \le \frac{b+\frac{1}{2}-np}{\sqrt{np(1-p)}} \right) \tag{2.10}$$

เมื่อ

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

ดังนั้น ในการประมาณค่าการแจกแจกทวินามด้วยการแจกแจงแบบปกติ การนำค่า ½ มาบวกเข้าและลบออกในการคำนวณหาค่าความน่าจะเป็น จะช่วยในการลดค่าความคาดเคลื่อนดังกล่าวได้

ในงานวิจัยนี้ได้นำหลักการประมาณค่าแบบแจกแจงปกติเข้ามาใช้ในการเพิ่มขยายกฎความสัมพันธ์เพื่อแก้ปัญหาการคำนวณค่าความน่าจะเป็นที่ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นประสบ เนื่องจากการคำนวณค่าแฟคทอเรียลเมื่อจำนวนจริงมีค่ามาก ซึ่งในที่นี้คือจำนวนรายการธุรกรรมของฐานข้อมูลซึ่งมีจำนวนมาก นอกจากนี้ยังช่วยลดข้อจำกัดของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นได้เรื่องของการจำกัดขนาดของข้อมูลใหม่ที่จะนำเข้ามา กล่าวอีกนัยหนึ่งคือขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจำเป็นจะต้องทราบขนาดที่แน่นอนของข้อมูลใหม่ที่เข้ามา จึงจะสามารถคำนวณหาความน่าจะเป็นได้ แต่สำหรับงานวิจัยฉบับนี้ จะประยุกต์ใช้การประมาณค่าแบบการแจกแจงปกติเข้ามาเพื่อให้สามารถคำนวณหาความน่าจะเป็นของการเกิดไอเท็มเซตได้โดยไม่จำเป็นต้องทราบขนาดของข้อมูลใหม่ที่จะเข้ามา โดยจะกล่าวในบทต่อไป

บทที่ 3
การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติกรณีไม่จำกัดขนาดของข้อมูลชุดใหม่

การเพิ่มขยายกฎความสัมพันธ์ เป็นการปรับปรุงกฎความสัมพันธ์ของข้อมูลในฐานข้อมูลขนาดใหญ่ เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิม อาจส่งผลให้กฎความสัมพันธ์ที่มีอยู่เดิมเปลี่ยนแปลงไป นั่นคือ กฎความสัมพันธ์เดิมบางกฎอาจจะยังคงอยู่และใช้ได้ดังเดิม ในขณะที่บางกฎอาจจะใช้ไม่ได้เหมือนเดิม นอกจากนี้อาจจะมีกฎความสัมพันธ์ใหม่ที่เกิดขึ้นมา สำหรับงานวิจัยฉบับนี้ จะเสนอขั้นตอนวิธีปรับปรุงกฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิม โดยอาศัยหลักการประมาณค่าแบบการแจกแจงปกติ เพื่อให้ขั้นตอนวิธีสามารถทำงานได้โดยไม่จำเป็นต้องทราบหรือจำกัดขนาดของชุดข้อมูลใหม่ที่ถูกเพิ่มเข้ามา

งานวิจัยนี้ เป็นการนำเสนอแนวคิดของการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ เพื่อแก้ไขปัญหาของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น [6] ดังนี้

1. ปัญหาในการคำนวณค่าความน่าจะเป็นของไอเท็มเซตที่ในฐานข้อมูลเดิมที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงด้วยวิธีการหาความน่าจะเป็นแบบเบอร์นูลลี ผลการศึกษาพบว่า มีปัญหาในการคำนวณความน่าจะเป็นในกรณีที่ต้องคำนวณค่าแฟคทอเรียลจากจำนวนเต็มที่มีค่ามาก โดยขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นแก้ไขปัญหาด้วยการเทียบค่า ซึ่งทำให้ค่าความน่าจะเป็นที่ได้มีโอกาสผิดพลาด กล่าวอีกนัยหนึ่งคือ ค่าความน่าจะเป็นที่ได้อาจจะไม่ใช่ค่าความน่าจะเป็นที่แท้จริง ส่งผลให้ขั้นตอนวิธีนั้นมีโอกาสคำนวณความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงผิดพลาด

2. ปัญหาการจำกัดขนาดของชุดข้อมูลใหม่ที่ถูกเพิ่มเข้ามา ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจำเป็นที่จะต้องทราบขนาดที่แน่นอนของชุดข้อมูลใหม่ที่จะเพิ่มเข้ามาทุกครั้งเพื่อช่วยในการคำนวณหาค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ทำให้เมื่อมีการเปลี่ยนแปลงขนาดของชุดข้อมูลใหม่จะต้องมีการคำนวณในส่วนของการประมวลผลฐานข้อมูลเดิม (Original Mining Phase) ใหม่ทุกครั้ง ทำให้เวลาที่ใช้ในการประมวลผลนานขึ้นกว่าเดิม

3. ปัญหาการคำนวณค่าความน่าจะเป็นของไอเท็มเซตทุกตัว ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นมีการคำนวณหาค่าความน่าจะเป็นของไอเท็มเซตทุกตัวแล้วนำมาเปรียบเทียบกับค่าความน่าจะเป็นอย่างน้อยที่สุดที่ของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ($\Pr ob_{PL}$) ที่ผู้ใช้เป็นผู้กำหนด เพื่อหาไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งเก็บไว้เพื่อใช้ในการประมวลผลเมื่อมีฐานข้อมูลชุดใหม่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิม

เพื่อแก้ไขปัญหาของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นดังกล่าวข้างต้น ในบทนี้จะกล่าวถึงแนวคิดและหลักการที่ใช้ในการแก้ไขปัญหา และขั้นตอนการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติดังนี้

3.1 การประมาณค่าของการแจกแจงทวินามโดยใช้การแจกแจกปกติ (Normal approximation to the binomial distribution)

ในหัวข้อนี้จะกล่าวถึงการแก้ไขปัญหาในการคำนวณค่าความน่าจะเป็นของไอเท็มเซตที่ในฐานข้อมูลเดิมที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงด้วยวิธีการหาความน่าจะเป็นแบบเบอร์นูลลี ซึ่งมีปัญหาในการคำนวณความน่าจะเป็นในกรณีที่ต้องคำนวณค่าแฟคทอเรียลจากจำนวนเต็มที่มีค่ามาก โดยขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นในงานวิจัยนี้ได้ใช้หลักการประมาณค่าของการแจกแจงทวินามโดยใช้การแจกแจงปกติมาแก้ไขปัญหาดังกล่าว อธิบายดังนี้

จากหลักการความน่าจะเป็นแบบเบอร์นูลลีที่ประกอบด้วยการทดลองจำนวน n ครั้งที่มีความเป็นอิสระต่อกัน และการเกิดเหตุการณ์ในการทดลองแต่ละครั้งจะประกอบด้วย 2 เหตุการณ์ คือผลสำเร็จของการทดลอง และผลความล้มเหลวของการทดลอง เมื่อนำหลักการของเบอร์นูลลี่มาประยุกต์ใช้กับการค้นหากฎความสัมพันธ์ สามารถนำมาหลักการดังกล่าวมาพิจารณาการเกิดของไอเท็มเซตในฐานข้อมูลได้คือ ให้การทดลอง n ครั้ง หมายถึง จำนวนรายการธุรกรรมในฐานข้อมูลจำนวน n รายการธุรกรรม และผลการทดลองที่เกิดขึ้นได้แก่ การเกิดหรือการปรากฏขึ้นของไอเท็มเซตนั้นแต่ละรายการธุรกรรมของฐานข้อมูล ซึ่งถ้าไอเท็มเซตที่พิจารณาปรากฏอยู่ในรายการธุรกรรม จะหมายถึงผลการทดลองที่สำเร็จ ในทางกลับกัน หากไอเท็มเซตที่พิจารณาไม่ปรากฏอยู่ในรายการธุรกรรม จะหมายถึงผลการทดลองที่ล้มเหลว

จาก ฟังก์ชั่นการกระจายตัวของตัวแปรสุ่มทวินาม ดังสมการที่ 3.1

$$f(x)=\binom{n}{x}p^x(1-p)^x \quad , \quad x=0,1,2,3,...,n \tag{3.1}$$

ค่าความน่าจะเป็นของการแจกแจงแบบทวินามที่ตัวแปรสุ่ม X จะให้ผลที่สำเร็จจำนวนน้อยกว่า k ครั้ง หรือ $P(X<k)$ ด้วยการทดลองจำนวน n ครั้ง สามารถคำนวณหาความน่าจะเป็นได้จากสมการที่ 3.2

$$P(X<k)=\sum_{x=0}^{k-1}\binom{n}{x}p^x(1-p)^{n-x} \tag{3.2}$$

จากสมการที่ 3.2 ความน่าจะเป็นที่ตัวแปรสุ่ม X จะให้ผลสำเร็จจำนวนมากกว่าหรือเท่ากับ k ครั้ง หรือ $P(X\geq k)$ ด้วยการทดลองจำนวน n ครั้ง สามารถคำนวณหาได้จากสมการที่ 3.3

$$P(X\geq k)=1-\sum_{x=0}^{k-1}\binom{n}{x}p^x(1-p)^{n-x} \tag{3.3}$$

จากสมการที่ 3.3 เมื่อนำมาประยุกต์ใช้ในงานวิจัยด้านการเพิ่มขยายกฎความสัมพันธ์ของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจะสามารถคำนวณค่าความน่าจะเป็นของเท็มเซตในฐานข้อมูลเดิมที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงตามสมการที่ 3.4

$$P(X\geq k)=1-\sum_{x=0}^{k-1}\binom{n+m}{x}p^x(1-p)^{n+m-x} \tag{3.4}$$

ตัวแปรที่ใช้ในการคำนวณหาค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตในฐานข้อมูลปรับปรุง ได้นิยามดังนี้

n แทน จำนวนรายการธุรกรรมทั้งหมดในฐานข้อมูลเดิม

m แทน จำนวนรายการธุรกรรมทั้งหมดในฐานข้อมูลปรับปรุง

p แทน ความน่าจะเป็นของการเกิดไอเท็มเซตที่พิจารณาในฐานข้อมูล

q หรือ $1-p$ แทน ความน่าจะเป็นของการไม่เกิดไอเท็มเซตที่พิจารณาในฐานข้อมูล

k แทน จำนวนครั้งของเกิดไอเท็มเซตที่พิจารณา ในที่นี้จะหมายถึงค่าสนับสนุนขั้นต่ำที่จะทำให้ไอเท็มเซตที่พิจารณาเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

ทั้งนี้ ค่าความน่าจะเป็นของการเกิดไอเท็มเซตที่พิจารณาในฐานข้อมูล หรือค่า p สามารถคำนวณได้ ตามสมการที่ 3.5

$$p = \frac{\delta_x^{DB}}{|DB|} \tag{3.5}$$

เมื่อ δ_x^{DB} แทน จำนวนรายการธุรกรรม ที่ปรากฏไอเท็มเซตที่พิจารณา และ$|DB|$ แทน จำนวนรายการธุรกรรมทั้งหมดของฐานข้อมูล

ด้วยปัญหาที่ค้นพบจากขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นในการคำนวณหาค่าความน่าจะเป็นของไอเท็มเซตด้วยสมการที่ 3.3 จะมีปัญหาการคำนวณค่าแฟคทอเรียลในพจน์ของ $\binom{n+m}{x}$ ซึ่งคำนวณได้จาก $\binom{n+m}{x} = \frac{(n+m)!}{(n+m-x)!x!}$ ปัญหาดังกล่าวคือเมื่อจำนวนเต็มมีค่ามาก จะทำให้ผลลัพธ์ที่ได้คือค่า *null* ซึ่งเกิดจากการข้อจำกัดของตัวแปรทางคอมพิวเตอร์ที่สามารถเก็บค่าได้สูงสุดคือตัวแปรชนิด long integer ซึ่งเก็บค่าได้มากสุดที่ $2^{63}-1$ นั่นคือ สามารถเก็บผลลัพธ์ที่ได้จากการคำนวณค่าแฟคทอเรียลได้สูงสุดคือ 20!

ด้วยปัญหาการคำนวณค่าความน่าจะเป็นดังกล่าวของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นทำให้งานวิจัยฉบับนี้ได้นำเอาหลักการประมาณค่าความน่าจะเป็นแบบทวินามด้วยการแจกแจงปกติมาประยุกต์ใช้ โดยอ้างอิงจากทฤษฎีบทลิมิตของเดมัวร์และลาปลาซ [ภาคผนวก ก] และแนวคิดการปรับแก้ความต่อเนื่อง (Continuity correction) จากตัวแปรสุ่มทวินามโดยใช้การประมาณค่าแบบแจกแจงปกติ ค่าความน่าจะเป็นที่ตัวแปรสุ่ม สุ่ม X จะให้ผลสำเร็จจำนวนน้อยกว่า k ครั้ง หรือ $P(X<k)$ ด้วยการทดลองจำนวน n ครั้ง ดังสมการที่ 3.6

$$P(X<k) \approx \sum_{x=0}^{k-1} \int_{x-0.5}^{x+0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \tag{3.6}$$

จากสมการที่ 3.6 จะได้ว่า ความน่าจะเป็นที่ตัวแปรสุ่ม X จะให้ผลสำเร็จจำนวนมากกว่าหรือเท่ากับ k ครั้ง หรือ $P(X \geq k)$ ด้วยการทดลองจำนวน n ครั้ง สามารถแสดงได้ดังนี้

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

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

$$= 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx$$

ผลที่ได้เมื่อนำมาประยุกต์ใช้การงานวิจัยด้านการค้นหากฎความสัมพันธ์ ค่าความน่าจะเป็นที่ไอเท็มเซตจะมีโอกาสเป็นไอเท็มเซตที่เกิดบ่อย หรือ ไอเท็มเซตมีโอกาสที่จะมีค่าสนับสนุน X มีค่ามากกว่าค่าสนับสนุนขั้นต่ำ k จะสามารถคำนวณหาได้จากสมการที่ 3.7 ดังนี้

$$P(X \geq k) = 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx \qquad (3.7)$$

เมื่อ $\mu = np$ และ $\sigma = \sqrt{npq}$ โดย n คือจำนวนรายการธุรกรรมของฐานข้อมูลปรับปรุง $|UD|$ และ $|UD| = |DB| + |db|$ ทั้งนี้ p สามารถคำนวณได้จากสมการที่ 3.5

สมการที่ 3.7 เป็นสมการที่จะนำมาใช้คำนวณหาค่าความน่าจะเป็นที่ไอเท็มเซตจะมีโอกาสเป็นฟรีเคว้นไอเท็มเซต ซึ่งสามารถแก้ไขปัญหาเรื่องของการคำนวณค่าแฟคทอเรียลของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นได้

3.2 การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติกรณีไม่จำกัดขนาดของข้อมูลชุดใหม่

นอกจากปัญหาการคำนวณค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงโดยใช้ทฤษฎีบททวินามของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจะมีปัญหาด้านการคำนวณแฟคทอเรียลแล้ว ผู้วิจัยได้ศึกษาและพบว่า การคำนวณค่าความน่าจะเป็นของไอเท็มเซตของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นมีความสัมพันธ์กับขนาดของชุดข้อมูลใหม่ นั่นคือขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจำเป็นต้องทราบขนาดที่แน่นอนของข้อมูลชุดใหม่ที่จะถูกเพิ่มเข้ามาเสียก่อน จึงจะคำนวณค่าความน่าจะเป็นได้ และหากขนาดของชุดข้อมูลใหม่ที่ถูกเพิ่มเข้ามามีการเปลี่ยนแปลง ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจะต้องประมวลผลในส่วนของการประมวลผลฐานข้อมูลเดิมเสียก่อน จึงจะประมวลผลในส่วนของการประมวลผลในฐานข้อมูลปรับปรุงได้ ซึ่งในความเป็นจริงข้อมูลชุดใหม่ที่ถูกเพิ่มเข้ามาในแต่ละครั้ง อาจจะมีขนาดของชุดข้อมูลไม่เท่ากันเสมอไป ในหัวข้อนี้ จะอธิบายถึงการนำหลักการประมาณค่าแบบการแจกแจงปกติมาประยุกต์ใช้ในขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยไม่จำกัดขนาดของชุดข้อมูลใหม่ ดังนี้

จากสมการที่ 3.7 ซึ่งเป็นสมการที่ใช้สำหรับคำนวณหาค่าความน่าจะเป็นที่ไอเท็มที่พิจารณาซึ่งมีค่าสนับสนุน X จะมีค่ามากกว่าค่าสนับสนุน k ในทางปฏิบัติ การคำนวณค่า X และค่า k จะเป็นค่าสนับสนุนที่เป็นจำนวนนับ (Support count) โดย X คือจำนวนรายการธุรกรรมที่ปรากฏไอเท็มเซตที่พิจารณา และ k คือค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด

โดยปกติแล้ว ค่าสนับสนุนขั้นต่ำจะถูกกำหนดให้อยู่ในรูปของเปอร์เซนต์ เช่น $\min\_sup = 40\%$ เมื่อนำมาคำนวณหาค่าความน่าจะเป็นในสมการที่ 3.4 และ 3.7 ค่า k จะถูกแปลงค่าให้อยู่ในรูปของจำนวนนับ เพื่อให้สะดวกต่อการทำงาน เช่น กำหนดให้มีจำนวนรายการธุรกรรมในฐานข้อมูลทั้งหมด 50 รายการธุรกรรม จำนวนรายการธุรกรรมที่ไอเท็มเซตปรากฏและจะทำให้ไอเท็มเซตนั้นกลายเป็นไอเท็มเซตที่เกิดบ่อย จะต้องมีจำนวนรายการธุรกรรมที่ปรากฏไอเท็มเซตนั้นอย่างน้อย $\min\_sup = 50 \times \frac{40}{100} = 20$ รายการธุรกรรม เป็นต้น

อย่างไรก็ตาม จากสมการที่ 3.4 และ 3.7 จะเห็นได้ว่า การคำนวณหาความน่าจะเป็นที่ไอเท็มที่พิจารณาซึ่งมีค่าสนับสนุน X จะมีค่ามากกว่าค่าสนับสนุน k หรือ $P(X \geq k)$ นั้น ค่า k ซึ่งอยู่ในรูปของจำนวนนับจะต้องทราบจำนวนชุดข้อมูลใหม่เสียก่อนจึงจะคำนวณหาค่า k ได้ ดังนั้นเพื่อแก้ไขปัญหาดังกล่าว ผู้วิจัยนำหลักการประมาณค่าความน่าจะเป็นด้วยการแจกแจงปกติ มาประยุกต์ใช้ดังนี้

กำหนดให้ x คือค่าสนับสนุนที่เป็นจำนวนนับ (Support count) ของไอเท็มเซต และ k คือค่าสนับสนุนขั้นต่ำที่เป็นจำนวนนับ (Minimum support count) ดังนั้น ความน่าจะเป็นที่ไอเท็มเซตจะมีค่าสนับสนุน x ที่มากกว่าค่าสนับสนุนขั้นต่ำ k ที่จะทำให้ไอเท็มเซตนั้นเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง สามารถคำนวณได้จากสมการ

$$P(x \geq k) = 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma_x \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2} dx \qquad (3.8)$$

เมื่อ $\mu_x = (n+m)p$ และ $\sigma_x = \sqrt{(n+m)pq}$ โดย n คือขนาดหรือจำนวนรายการธุรกรรมของฐานข้อมูลเดิม และ m คือขนาดหรือจำนวนรายการธุรกรรมของฐานข้อมูลใหม่

ในการประยุกต์ใช้สมการที่ 3.8 เพื่อหาความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงนั้น ค่า X และค่า k จะอยู่ในรูปแบบของจำนวนนับ ซึ่งเป็นการนับจำนวนรายการธุรกรรมที่มีไอเท็มที่พิจารณาปรากฏ จากรูปสมการที่ 3.8 จะเห็นได้ว่า ค่าความน่าจะเป็นจะแปรผันตามค่า m นั่นหมายความว่าจะต้องทราบจำนวน m หรือจำนวนรายการธุรกรรมของฐานข้อมูลใหม่ จึงจะคำนวณได้

อย่างไรก็ตาม ด้วยลักษณะทั่วไปของการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลในแต่ละครั้งจำนวนรายการธุรกรรมที่ถูกเพิ่มอาจจะไม่เท่ากันทุกครั้ง ดังนั้น งานวิจัยฉบับนี้ได้เสนอแนวทางการแก้ไขปัญหาการจำกัดขนาดของชุดข้อมูลใหม่ที่เพิ่มเข้ามา หรือ ค่า m ในสมการที่ 3.8 โดยประยุกต์ใช้หลักการของค่าสนับสนุนซึ่งเป็นแนวคิดพื้นฐานของการค้นหากฎความสัมพันธ์ ดังนี้

ค่าสนับสนุน (Support factor) หมายถึง ค่าความน่าจะเป็นที่ไอเท็ม X และ Y จะเกิดหรือปรากฏขึ้นพร้อมกันในรายการธุรกรรมเดียวกันภายในฐานข้อมูลทั้งหมด คำนวณได้จากสมการที่ 3.9 [20]

$$Support, s(X \rightarrow Y) = \frac{freq(X \cup Y)}{N} \qquad (3.9)$$

เมื่อ

$freq(X \cup Y)$ หมายถึง จำนวนรายการธุรกรรมในฐานข้อมูลที่ปรากฏไอเท็ม X และไอเท็ม Y ในรายการธุรกรรมเดียวกัน

N หมายถึง จำนวนรายการธุรกรรมในฐานข้อมูลทั้งหมด

จากสมการที่ 3.9 จะเห็นได้ว่าค่าสนับสนุนที่คำนวณได้ ซึ่งหมายถึงความน่าจะเป็นของโอกาสที่ไอเท็ม X และไอเท็ม Y ในรายการธุรกรรมเดียวกันจะมีค่าอยู่ระหว่าง $0-1$ ซึ่งไม่ใช่เลขจำนวนนับ ดังนั้นในงานวิจัยจึงประยุกต์ใช้ค่าสนับสนุนแทนค่าสนับสนุนที่เป็นจำนวนนับ

ด้วยแนวคิดของค่าสนับสนุน (support factor) ในงานวิจัยนี้จึงประมาณค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง จะมีค่าใกล้เคียงกับค่าความน่าจะเป็นที่ไอเท็มเซตเกิดขึ้นในฐานข้อมูลเดิม ดังนั้น จึงใช้ค่าสนับสนุนแทนค่าสนับสนุนที่เป็นจำนวนนับ จากสมการที่ 3.8 ซึ่งเป็นการหาค่าความน่าจะเป็นที่ไอเท็มเซตจะมีค่าสนับสนุน X มากกว่าค่าสนับสนุนขั้นต่ำ k เมื่อ k คือค่าสนับสนุนขั้นต่ำที่จะทำให้ไอเท็มเซตที่พิจารณากลายเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

กำหนดให้

y แทน ค่าสัดส่วนของค่าสนับสนุน (Support factor) และ $y=\frac{x}{n+m}$ ซึ่งมีค่าอยู่ระหว่าง $0-1$

ms แทน สัดส่วนค่าสนับสนุนขั้นต่ำ (Minimum support factor) ซึ่งกำหนดโดยผู้ใช้ ซึ่งมีค่าอยู่ระห่าง $0-1$

แทนค่า y และ ms แทนในสมการที่ 3.8 จะได้

$$P(y \geq ms) = 1 - \int_{0}^{ms} \frac{1}{\sigma_y \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu_y}{\sigma_y}\right)^2} dy \qquad (3.10)$$

เมื่อ $\mu_y = p$ และ $\sigma_y^2 = \frac{pq}{n}$

การหาค่า $\mu_y = p$ และ $\sigma_y^2 = \frac{pq}{n}$ แสดงได้ดังนี้

$$\begin{aligned} \mathrm{E}(y) &= \mu_y \\ &= \frac{\mathrm{E}(x)}{n+m} \\ &= \frac{\mu_x}{n+m} \\ &= \frac{(n+m)p}{(n+m)} \\ &= p \end{aligned}$$

$$\therefore \mu_y = p \qquad (3.11)$$

และ

$$
\begin{aligned}
Var(y) &= \sigma_y^2 \\
&= \frac{1}{(n+m)^2}\sigma_x^2 \\
&= \frac{(n+m)pq}{(n+m)^2} \\
&= \frac{pq}{(n+m)} \\
\therefore \sigma_y^2 &\le \frac{pq}{(n+m)}
\end{aligned} \tag{3.12}
$$

จากสมการที่ 3.12 จะเห็นได้ว่า ค่าความแปรปรวน σ_y^2 มีความจำเป็นต้องทราบค่า m ดังนั้นเพื่อให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติที่นำเสนอในงานวิจัยนี้ สามารถประมวลผลได้โดยไม่จำกัดขนาดของฐานข้อมูลใหม่ที่จะถูกเพิ่มเข้ามา ผู้วิจัยได้ประมาณค่า $\sigma_y^2 = \frac{pq}{(n+m)}$ ด้วย $\sigma_y^2 = \frac{pq}{n}$ เนื่องจาก เมื่อนำค่า $\sigma_y^2 = \frac{pq}{n}$ และ $\sigma_y^2 = \frac{pq}{(n+m)}$ ไปแทนค่าเพื่อหาค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อย ค่า $\sigma_y^2 = \frac{pq}{n}$ จะให้ผลลัพธ์จากการคำนวณความน่าจะเป็นที่สูงกว่า $\sigma_y^2 = \frac{pq}{(n+m)}$ เสมอ

ดังนั้น เพื่อให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติสามารถประมวลผลได้โดยไม่จำเป็นต้องทราบขนาดของฐานข้อมูลชุดใหม่ที่จะถูกเพิ่มเข้ามาในฐานข้อมูลเดิม ผู้วิจัยจึงได้ประมาณค่า $\sigma_y^2 = \frac{pq}{(n+m)}$ ด้วย $\sigma_y^2 = \frac{pq}{n}$

นั่นคือ

ถ้า $\sigma_y^2 = \frac{pq}{(n+m)} \le \frac{pq}{n}$ แล้ว

จะได้

$$
P(y > ms) \quad \le \quad \bar{\mathrm{P}}(y \ge ms \mid \mu_y = p, \sigma_y^2 = \frac{pq}{n}) \tag{3.13}
$$

เมื่อ $P(y > ms)$ คือความน่าจะเป็นของไอเท็มเซตที่เกิดขึ้นจริงเมื่อมีชุดข้อมูลใหม่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิม และ $\bar{P}(y \geq ms)$ คือความน่าจะเป็นของไอเท็มเซตที่ได้จากการประมาณค่าด้วยสมการที่ 3.10 ด้วยค่า $\mu_y = p$ และ $\sigma_y^2 = \frac{pq}{n}$

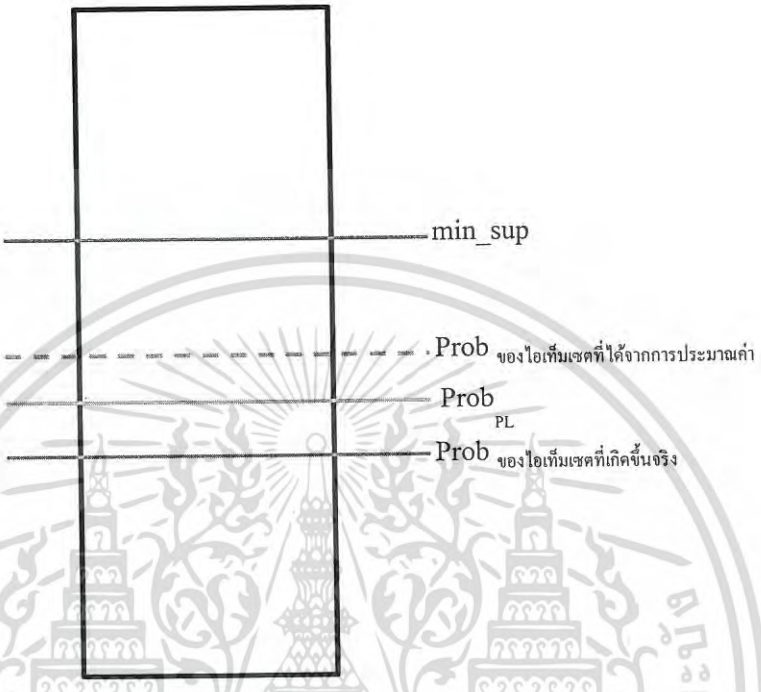
จากสมการที่ 3.13 เมื่อนำมาประยุกต์ใช้ในการประมวลผลของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ผลลัพธ์จากการนำคำนวณค่าความน่าจะเป็นด้วยสมการที่ 3.10 ด้วยค่า $\sigma_y^2 = \frac{pq}{n}$ จะทำให้ค่าความน่าจะเป็นที่คำนวณได้มีค่ามากกว่าการคำนวณค่าความน่าจะเป็นด้วยค่า $\sigma_y^2 = \frac{pq}{(n+m)}$ ส่งผลให้ความน่าจะเป็นของไอเท็มเซตที่คำนวณด้วยการประมาณค่ามีค่ามากกกว่าความน่าจะเป็นของไอเท็มเซตที่เกิดขึ้นจริง ทำให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ เก็บไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงมากกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

ด้วยแนวคิดของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ มีการทำงานบนพื้นฐานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นดังนั้นจึงมีการกำหนดค่า Prob_{PL} เพื่อใช้ในการค้นหาไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมแต่มีโอกาสเป็นไอเท็มเซตในฐานข้อมูลปรับปรุง ดังนั้น ในรูปที่ 3.1 ได้แสดงความสัมพันธ์ของของความน่าจะเป็นของไอเท็มที่ได้จากการประมาณค่าและความน่าจะเป็นของไอเท็มเซตที่เกิดขึ้นจริง ดังนี้

ค่าสนับสนุนขั้นต่ำ min_sup จะใช้เป็นค่าทดสอบไอเท็มที่เป็นไอเท็มเซตที่เกิดบ่อย หากไอเท็มใดๆ ที่พิจารณามีค่าสนับสนุนมากกว่าหรือเท่ากับค่า min_sup จะเรียกไอเท็มเซตนั้นว่าไอเท็มเซตที่เกิดบ่อย ในขณะเดียวกัน ไอเท็มเซตใดๆ ที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย แต่มีค่าความน่าจะเป็นมากกว่าค่า Prob_{PL} ที่กำหนดโดยผู้ใช้ ไอเท็มเซตนั้นจะเรียกว่า ไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

ทั้งนี้ในการประมาณค่าความน่าจะเป็นของการเกิดไอเท็มเซตด้วยสมการที่ 3.10 ที่ $\sigma_y^2 = \frac{pq}{n}$ จะทำให้ค่าความน่าจะเป็นของไอเท็มเซตที่ได้การประมาณค่า (เส้นประ) มีค่ามากกว่าค่าความน่าจะเป็นของไอเท็มเซตที่เกิดขึ้นจริง เสมอ (เส้นทึบ) นั่นหมายความว่า เมื่อมีข้อมูลชุดใหม่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิมแล้ว ค่าความน่าจะเป็นที่เกิดขึ้นจริงของไอเท็มเซตที่พิจารณา อาจจะมีค่าต่ำกว่า Prob_{PL} ที่กำหนดโดยผู้ใช้ จะทำให้ไอเท็มเซตนั้นถูกตัดทิ้งและไม่ถูกเก็บเพื่อนำไปประมวลผลในรอบถัดไป แต่หากไอเท็มเซตนั้นถูกคำนวณด้วยสมการที่ 3.10 ที่ $\sigma_y^2 = \frac{pq}{n}$ แล้ว จะ

ทำให้ค่าความน่าจะเป็นของไอเท็มเซตนั้นมีโอกาสที่สูงกว่าค่า $Prob_{PL}$ ซึ่งไอเท็มเซตดังกล่าวจะถูกเก็บไว้เพื่อนำไปประมวลผลในรอบถัดไปที่มีข้อมูลชุดใหม่เพิ่มเข้ามา



รูปที่ 3.1 ความสัมพันธ์ของความน่าจะเป็นของไอเท็มเซตที่ได้จากการประมาณค่าและความน่าจะเป็นของไอเท็มเซตที่เกิดขึ้นจริง

ดังนั้น ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติที่นำเสนอในงานวิจัยนี้ จะเก็บไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงได้ครอบคลุมกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น ซึ่งจะช่วยลดจำนวนไอเท็มเซตที่จะถูกนำไปสแกนในฐานข้อมูลเดิม และทำให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติสามารถประมวลผลได้โดยไม่จำเป็นต้องทราบขนาดฐานข้อมูลชุดใหม่ที่จะถูกเพิ่มเข้ามา

3.3 การคำนวณค่าทดสอบที่ใช้ในการค้นหาไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

การเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติที่นำเสนอในงานวิจัยฉบับนี้ มีวัตถุประสงค์ในการออกแบบขั้นตอนวิธีการทำงานเพื่อแก้ไขปัญหาของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นได้แก่ ปัญหาการคำนวณค่าแฟคทอเรียลกรณี n มีค่ามากในการคำนวณค่าความน่าจะเป็นด้วยวิธีการของเบอร์นูลลี โดยวิธีการแก้ปัญหาได้ใช้หลักการประมาณค่าการแจกแจงทวินามด้วยการแจกแจงปกติ ซึ่งได้อธิบายในหัวข้อที่ 3.1 ส่วนการแก้ไขปัญหาการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติกรณีไม่จำกัดขนาดของฐานข้อมูลใหม่ ได้อธิบายไว้ในหัวข้อที่ 3.2

สำหรับหัวข้อนี้จะนำเสนอวิธีการแก้ไขปัญหาการคำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัวเพื่อหาไอเท็มเซตที่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นซึ่งหากฐานข้อมูลมีไอเท็มจำนวนมากจะทำให้อัลกอริทึมสูญเสียเวลาในการคำนวณค่าความน่าจะเป็นของไอเท็มเซตดังกล่าว งานวิจัยฉบับนี้จะนำเสนอวิธีการแก้ไขปัญหาการคำนวณค่าความน่าจะเป็นของไอเท็มเซตทุกตัวกล่าวให้เหลือเพียงค่าทดสอบเพียงค่าเดียวที่ใช้ทดสอบไอเท็มเซตที่พิจารณา ว่ามีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงหรือไม่ โดยอธิบายได้ดังนี้

ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นนอกจากผู้ใช้จะต้องกำหนดค่าสนับสนุนขั้นต่ำเพื่อใช้หาไอเท็มเซตที่เกิดบ่อยแล้ว ผู้ใช้จะต้องกำหนดค่า $\mathrm{Pr}ob_{PL}$ อีกหนึ่งค่าเพื่อใช้พิจารณาไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิมแต่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง และจะทำการเก็บไอเท็มเซตดังกล่าวไว้ใช้ในการประมวลผลเพื่อหาไอเท็มเซตที่เกิดบ่อยเมื่อมีฐานข้อมูลชุดใหม่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิม

ขั้นตอนการพิจารณาไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจะเริ่มจากการคำนวณค่าความน่าจะเป็นของไอเท็มเซตทุกตัว แล้วนำค่าความน่าจะเป็นที่คำนวณได้มาเปรียบเทียบกับค่า $\mathrm{Pr}ob_{PL}$ หากไอเท็มเซตใดมีค่ามากกว่าหรือเท่ากับค่า $\mathrm{Pr}ob_{PL}$ ที่กำหนด ไอเท็มเซตนั้นจะถูกเก็บไว้ และค่าคาดหวังขั้นต่ำที่ทำให้ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง หรือ σ^{DB} จะคำนวณจากจำนวนนับของไอเท็มเซตที่เป็นไอเท็มเซตที่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงที่มีค่าความน่าจะเป็นที่น้อยที่สุดแต่มีค่ามากกว่าค่า $\mathrm{Pr}ob_{PL}$ ที่กำหนด

จากกระบวนการหาค่าคาดหวังขั้นต่ำที่ทำให้ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม แต่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง หรือ σ^{DB} ของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นดังกล่าวข้างต้น จะเห็นได้ว่าจะต้องมีการคำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัวแล้วจึงนำมาเปรียบเทียบกับค่า $\Pr ob_{PL}$ เพื่อหาค่า σ^{DB} ตัวอย่าง หากมีจำนวนไอเท็มเซตที่เป็น C_1, C_2 และ C_3 เป็น 40, 100 และ 250 ไอเท็มเซตตามลำดับ ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจำเป็นต้องคำนวณค่าความน่าจะเป็นของไอเท็มเซตทั้งหมด 390 ไอเท็มเซต เพื่อที่จะหาว่าไอเท็มเซตตัวใดที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยซึ่งเป็นขั้นตอนที่สูญเสียเวลาเนื่องจากไอเท็มเซตบางตัวไม่จำเป็นต้องคำนวณหาค่าความน่าจะเป็นที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง เนื่องจากเป็นไอเท็มเซตที่เกิดบ่อยอยู่แล้ว หรือบางไอเท็มเซตมีค่าความน่าจะเป็นน้อยมากจนไม่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อย

ดังนั้นงานวิจัยฉบับนี้ ได้นำเสนอการคำนวณหาค่าคาดหวังขั้นต่ำของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง σ^{DB} สามารถคำนวณได้ดังนี้

กำหนดให้ ค่าความน่าจะเป็นที่น้อยที่สุดที่ทำให้ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมแต่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง คำนวณได้จาก

$$P(y \geq ms) = \Pr ob_{PL} \tag{3.14}$$

ดังนั้น เมื่อแทนค่า $P(y \geq ms)$ ด้วยสมการที่ 3.10 จะคำนวณค่าความน่าจะเป็นที่น้อยที่สุดที่ทำให้ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมแต่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ได้จากสมการที่ 3.15

$$1 - \int_0^{ms} \frac{1}{\sigma_y \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu_y}{\sigma_y}\right)^2} dy = \Pr ob_{PL} \tag{3.15}$$

เมื่อ $\mu_y = p$ และ $\sigma_y^2 = \frac{pq}{n}$

เนื่องจากค่าความน่าจะเป็นที่ไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงซึ่งคำนวณได้จากสมการที่ 3.15 ไม่ได้อยู่ในรูปของฟังก์ชันสมการเส้นตรง

(Non-linear function) จำเป็นต้องคำนวณด้วยระเบียบวิธีทางตัวเลข (Numerical method) เช่น วิธีการของ Newton-Raphson เป็นต้น

เมื่อคำนวณหาค่าความน่าจะเป็นที่น้อยที่สุดที่ทำให้ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมแต่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงจากสมการที่ 3.16 เรียบร้อยแล้ว ผลลัพธ์ที่ได้จะถูกนำมาคูณกับจำนวนรายการธุรกรรมของฐานข้อมูลเดิม $|DB|$ เพื่อหาค่าคาดหวังขั้นต่ำของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง σ^{DB}

3.4 ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจงแจงปกติ กรณีไม่จำกัดขนาดของข้อมูลชุดใหม่

การเพิ่มขยายการค้นหากฎความสัมพันธ์ในงานวิจัยฉบับนี้ จะใช้หลักการค้นหากฎความสัมพันธ์โดยใช้ขั้นตอนวิธี อะพริโอริ เป็นฐาน ซึ่งเป็นการค้นหาข้อมูลตามลำดับจำนวนสมาชิกของไอเท็มเซตจากน้อยไปหามาก ($k = 1,2,3,..., n$) โดยนำเอาไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-1$ ไอเท็มเซตมาใช้ในการสร้าง แคนดิเดต k ไอเท็มเซต ด้วยขั้นตอนการเชื่อมไอเท็มเซต (Join) และการตัด (Prune) ไอเท็มที่ไม่สามารถเป็นไอเท็มเซตที่เกิดบ่อยออกไป นอกจากนี้ยังได้นำหลักการประมาณค่าการแจกแจงทวินามด้วยการแจกแจงปกติ (Normal approximation to the binomial) มาประยุกต์ใช้เพื่อคำนวณหาค่าความน่าจะเป็นของไอเท็มเซตในฐานข้อมูลเดิมที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงเพื่อแก้ไขปัญหาการคำนวณค่าแฟคทอเรียลที่ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นประสบ อีกทั้งช่วยลดข้อจำกัดด้านความจำเป็นที่จะต้องทราบขนาดของฐานข้อมูลใหม่ที่จะถูกเพิ่มเข้ามา ทำให้ขั้นตอนวิธีที่นำเสนอในงานวิจัยนี้สามารถรักษากฎความสัมพันธ์ได้โดยไม่จำเป็นต้องประมวลผลฐานข้อมูลเดิมทุกครั้งเมื่อจำนวนข้อมูลชุดใหม่ที่เข้ามาเปลี่ยนแปลง

ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ จะแบ่งการทำงานออกเป็น 2 ขั้นตอนหลักด้วยกัน ได้แก่

1. การค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม

2. การค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่และปรับปรุงให้เป็นปัจจุบัน

รายละเอียดการทำงานของขั้นตอนวิธีทั้ง 2 ขั้นตอนหลัก มีตัวแปรหรือสัญลักษณ์ต่างๆ กำหนดไว้ ดังนี้

ตารางที่ 3.1 สัญลักษณ์ที่ใช้ในขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ

สัญลักษณ์	ความหมาย
$\|DB\|$	ขนาดหรือจำนวนรายการธุรกรรมของฐานข้อมูลเดิม (Original database size)
$\|db\|$	ขนาดหรือจำนวนรายการธุรกรรมของฐานข้อมูลใหม่ (Increment database size)
$\|UD\|$	ขนาดหรือจำนวนรายการธุรกรรมของฐานข้อมูลปรับปรุง (Updated database size)
F_k^{DB}	ไอเท็มเซตที่เกิดบ่อย $k-$ ไอเท็มเซต ของฐานข้อมูลเดิม
F_k^{db}	ไอเท็มเซตที่เกิดบ่อย $k-$ ไอเท็มเซต ของฐานข้อมูลใหม่
F_k^{UD}	ไอเท็มเซตที่เกิดบ่อย $k-$ ไอเท็มเซต ของฐานข้อมูลปรับปรุง
EF_k^{DB}	ไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย $k-$ ไอเท็มเซตในฐานข้อมูลเดิม
EF_k^{UD}	ไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย $k-$ ไอเท็มเซตในฐานข้อมูลปรับปรุง
δ_x^{DB}	ค่าสนับสนุนของไอเท็มที่พิจารณา ในฐานข้อมูลเดิม
δ_x^{db}	ค่าสนับสนุนของไอเท็มที่พิจารณา ในฐานข้อมูลใหม่
δ_x^{UD}	ค่าสนับสนุนของไอเท็มที่พิจารณา ในฐานข้อมูลปรับปรุง
C_1^{DB}	แคนดิเดต 1- ไอเท็มเซต ของฐานข้อมูลเดิม
C_1^{db}	แคนดิเดต 1- ไอเท็มเซต ของฐานข้อมูลใหม่
C_1^{UD}	แคนดิเดต 1- ไอเท็มเซต ของฐานข้อมูลปรับปรุง
s	ค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด
σ^{DB}	ค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม
σ^{UD}	ค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง
Prob_{PL}	ค่าความน่าจะเป็นที่น้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นไอเท็มเซตที่เกิดบ่อย กำหนดโดยผู้ใช้

3.4.1 การค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม

โดยทั่วไปไอเท็มเซตที่เกิดบ่อยจะพิจารณาจากไอเท็มเซตใดๆ ที่มีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำที่ผู้ใช้เป็นผู้กำหนด แต่เมื่อมีการเพิ่มข้อมูลใหม่เข้ามาแล้วทำการประมวลผลต่อไป

โดยหลีกเลี่ยงการสแกนฐานข้อมูลเดิม การจัดเก็บแต่ไอเท็มเซตที่เกิดบ่อยเพียงอย่างเดียวไม่เพียงพอต่อการประมวลผล ดังนั้น การจัดเก็บไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง (Expected frequent itemset) ซึ่งแทนด้วยสัญลักษณ์ EF จะช่วยให้การทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ ทำงานได้ดีขึ้น โดยลดจำนวนการสแกนฐานข้อมูลเดิมให้เหลือเพียงแค่ครั้งเดียว

การค้นหาไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของขั้นตอนวิธีที่นำเสนอในงานวิจัยนี้ ผู้ใช้จะต้องกำหนดค่าทดสอบขึ้นมาอีกหนึ่งค่า คือ ค่าความน่าจะเป็นน้อยที่สุดที่คาดว่าไอเท็มจะกลายเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งแทนด้วยสัญลักษณ์ Prob_{PL} โดยค่าที่กำหนดจะอยู่ช่วงระหว่าง $0-1$ เพื่อนำไปใช้ในการคำนวณค่า σ^{DB} ซึ่งจะเป็นค่าที่ช่วยในการพิจารณาไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงที่จะต้องจัดเก็บ

การทำงานของขั้นตอนวิธีการค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมแสดงดังรูปที่ 3.2 โดยขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ จะเริ่มคำนวณค่า σ^{DB} เพื่อใช้ทดสอบไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซต สำหรับการทำงานในรอบแรก $(k=1)$ ไอเท็มเซตที่พิจารณาทุกตัวจะถูกนำไปสแกนในฐานข้อมูลเดิม (DB) เพื่อคำนวณหาค่าสนับสนุน (δ_x^{DB}) ของแต่ละไอเท็มเซต จากนั้นจึงนำค่าสนับสนุนที่ได้มาเปรียบเทียบกับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด s หากไอเท็มเซตใดที่มีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำ ไอเท็มนั้นจะถูกเก็บไว้ในเซตของไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม (F_k^{DB}) ส่วนไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย จะถูกนำมาทดสอบกับค่า σ^{DB} ที่คำนวณไว้ตั้งแต่ตอนต้น เพื่อพิจารณาไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย (EF_k^{DB}) ในรอบที่ $k=1$ นี้ ไอเท็มทุกตัวทั้งที่เป็น F_k^{DB}, EF_k^{DB} และไอเท็มที่ไม่ใช่ทั้ง F_k^{DB} และ EF_k^{DB} จะถูกเก็บไว้ใน C_1^{DB} ดังบรรทัดที่ $4-8$ เพื่อใช้ในการประมวลผลในส่วนของ Incremental mining phase

ตั้งแต่รอบที่ 2 ขึ้นไป $(k \geq 2)$ ไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อย F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย EF_k^{DB} จะถูกนำมาดำเนินการเชื่อมไอเท็มเซต เพื่อสร้างแคนดิเดต $k-$ ไอเท็มเซต ตามบรรทัดที่ 11 เมื่อได้แคนดิเดต $k-$ ไอเท็มเซต เรียบร้อยแล้ว ขั้นตอนวิธีทำการค้นหาไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อย F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย EF_k^{DB} เช่นเดียวกับรอบแรก (บรรทัดที่ $3-8$)

เมื่อเสร็จสิ้นการประมวลผลในส่วน Original mining phase ผลลัพธ์ที่ได้คือ F_k^{DB}, EF_k^{DB} และ C_1^{DB}

Algorithm1: Original Mining Phase

Input: $DB, \Pr ob_{PL}, s$

Output: $F_k^{DB}, EF_k^{DB}, C_1^{DB}, \sigma^{DB}$

1 calculate σ^{DB} // using equation 3.15

2. $k=1$

3 scan DB for all $X \in C_k$ and obtain δ_x^{DB}

4 $F_k^{DB} = \{X \mid \sigma_x^{DB} \geq s\times |DB|\}$

5 for $\{X \mid \sigma_x^{DB} < s\times |DB|\}$

6 $EF_k^{DB} = \{X \mid s\times |DB > \sigma^{DB} \geq \rho^{DB}\}$

7 $C_1^{DB} = \{X \mid X \in (F_1^{DB} \cup EF_1^{DB} \cup (F_1^{DB} \cup EF_1^{DB})^c\}$

8 end

9 $k=2$

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

11 $C_k = (F_{k-1}^{DB} \cup EF_{k-1}^{DB}) * (F_{k-1}^{DB} \cup EF_{k-1}^{DB})$

12 repeat line 3-8

14 $k++$

15 end while loop

16 Return $F_k^{DB}, EF_k^{DB}, C_1^{DB}, \sigma^{DB}$

รูปที่ 3.2 การทำงานของขั้นตอนวิธีการค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิม

3.4.2 การค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่และปรับปรุงให้เป็นปัจจุบัน

ในการเพิ่มขยายกฎความสัมพันธ์ เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิมจะส่งผลให้ไอเท็มเซตที่เกิดบ่อยที่หาได้ก่อนหน้าเปลี่ยนแปลงไป เช่น ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมบางตัวอาจจะไม่ใช่ไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง หรือ ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมอาจเปลี่ยนเป็นเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงก็ได้

งานวิจัยทางด้านการเพิ่มขยายกฎความสัมพันธ์ส่วนใหญ่มีทั้งการสแกนฐานข้อมูลเดิมและฐานข้อมูลใหม่ ดังเช่นขั้นตอนวิธี เอฟยูพี ซึ่งจะมีการปรับปรุงค่าสนับสนุนที่ปรากฏในฐานข้อมูลใหม่ให้แก่ไอเท็มเซตที่เกิดบ่อยตัวเดิมที่หาได้ก่อนหน้า ในกรณีเช่นนี้ขั้นตอนวิธี เอฟยูพี ไม่จำเป็นต้องนำไอเท็มเซตดังกล่าวไปสแกนในฐานข้อมูลเดิม เนื่องจากทราบค่าสนับสนุนเดิมและสนับสนุนใหม่ จึงสามารถปรับปรุงได้ทันที แต่ในกรณีที่ไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลเดิมแต่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่ ในกรณีนี้ขั้นตอนวิธี เอฟยูพี จะต้องนำไอเท็มดังกล่าวไปสแกนในฐานข้อมูลเดิมเพื่อหาค่าสนับสนุน จากนั้นจึงดำเนินการปรับปรุงค่าสนับสนุนให้เป็นค่าปัจจุบัน เนื่องจากส่วนใหญ่ข้อมูลชุดใหม่จะมีขนาดเล็กกว่าฐานข้อมูลเดิม ดังนั้นอาจจะพบจำนวนไอเท็มที่เป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่จำนวนมาก ดังนั้น การสแกนฐานข้อมูลเดิมเพื่อหาค่าสนับสนุนให้กับไอเท็มเซตที่เกิดบ่อยใหม่จำนวนมากจะทำให้ใช้เวลานาน โดยไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่นี้อาจจะไม่สามารถกลายเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงได้เลย เป็นต้น

สำหรับงานวิจัยฉบับนี้ ขั้นตอนวิธีที่นำเสนอ จะช่วยลดจำนวนครั้งของการสแกนฐานข้อมูลเดิม โดยมีการเก็บทั้งไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยบางส่วนที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง โดยมีกระบวนการทำงานตาม Algorithm1 ซึ่งเป็นกระบวนการทำงานในการประมวลผลฐานข้อมูลเดิม (Original mining phase)

เมื่อข้อมูลชุดใหม่ถูกเพิ่มเข้ามา ขั้นตอนวิธีที่นำเสนอจะมีกระบวนการทำงานเพื่อหาไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง แสดงตาม Algorithm2 (Incremental mining phase) ในรูปที่ 3.3

จากรูปที่ 3.3 แสดงการทำงานของขั้นตอนวิธีการค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่ โดยในการทำงานของขั้นตอนวิธีนี้จำเป็นต้องทราบไอเท็มเซตที่เกิดบ่อย F_k^{DB} ไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย EF_k^{DB} แคนดิเดต 1-ไอเท็มเซต C_1^{DB} และค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม หรือ σ^{DB} ของฐานข้อมูลเดิมเสียก่อน ซึ่งหาได้จากขั้นตอนวิธีที่แสดงดังภาพที่ 3.2

ในการค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลใหม่และปรับปรุงให้เป็นปัจจุบันของงานวิจัยนี้สามารถแบ่งการทำงานได้เป็น 4 ส่วนหลักดังนี้

Algorithm2: Incremental Mining Phase

Input: $DB, db, \Pr ob_{PL}, s, F_k^{DB}, EF_k^{DB}, C_1^{DB}, \sigma^{DB}$

Output: $F_k^{UD}, EF_k^{UD}, C_1^{UD}, \sigma^{UD}$

1. calculate σ^{DB} //using equation 3.15
1 $k=1$
2 Updating $1-itemset$ ( ) // call algorithm 3
3 for $k=2$
4 while $F_{k-1}^{UD} \neq \phi$
5 Generating Candidate itemset ( ) // call algorithm 4
6 Updating $k-itemset$ ( ) // call algorithm 5
7 $k++$
8 end while loop
9 if $Temp\_scanDB \neq \phi$
10 Rescanning original database ( ) // call algorithm 6
11 endif
12 clear $Temp\_scanDB$
13 Return $F_k^{UD}, EF_k^{UD}, C_1^{UD}, \sigma^{UD}$

รูปที่ 3.3 การทำงานของขั้นตอนวิธีการค้นหาไอเท็มเซตที่เกิดบ่อย และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลใหม่

3.4.2.1 การปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยขนาด 1 – ไอเท็มเซต และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด 1 – ไอเท็มเซต

ในรอบแรก $k=1$ เมื่อมีข้อมูลชุดใหม่ถูกเพิ่มเข้ามา ขั้นตอนวิธีจะเริ่มปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เป็นไอเท็มเซตที่เกิดบ่อยและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย โดยมีกระบวนการทำงานตาม Algorithm3 ดังรูปที่ 3.4

การปรับปรุงค่าสนับสนุนของไอเท็มเซตจะเริ่มสแกนฐานข้อมูลใหม่เพื่อหาค่าสนับสนุนของไอเท็มเซต จากนั้นจึงปรับปรุงค่าสนับสนุนให้แก่ไอเท็มเซตทุกตัวที่เป็นไอเท็มเซตที่เกิดบ่อยขนาด 1 – ไอเท็มเซตและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด 1 – ไอเท็มเซต ซึ่งจะถูกกำหนดให้เป็น แคนดิเดต 1 – ไอเท็มเซตของฐานข้อมูลเดิมทุกตัว แทนด้วย C_1^{DB} จากนั้นจะทำการปรับปรุงค่าสนับสนุน ดังบรรทัดที่ 2-6

Algorithm3: Updating $1-itemset$

Input: $DB, db, \Pr ob_{PL}, s, C_1^{DB}, \sigma^{DB}$

Output: $F_1^{UD}, EF_1^{UD}, C_1^{UD}$

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

10 $C_1^{UD} = \{X \mid X \in (F_1^{UD} \cup EF_1^{UD} \cup (F_1^{UD} \cup EF_1^{UD})^c)\}$

11 end

12 Return $F_1^{UD}, EF_1^{UD}, C_1^{UD}$

รูปที่ 3.4 การปรับปรุงค่าไอเท็มเซตที่เกิดบ่อยขนาด $1-$ ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $1-$ ไอเท็มเซต

เมื่อ $1-$ ไอเท็มเซตทุกตัวที่ได้รับการปรับปรุงค่าสนับสนุนเรียบร้อยแล้ว จะถูกนำมาเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ s ที่ผู้ใช้กำหนด หากไอเท็มเซตใดที่มีค่าสนับสนุนมากกว่าค่าสนับสนุนขั้นต่ำ ไอเท็มนั้นจะถูกเก็บไว้ในเซตของไอเท็มเซตที่เกิดบ่อย F_k^{UD} ตามบรรทัดที่ 7 ส่วนไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อย จะถูกนำมาคำนวณหาไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย EF_k^{UD} ดังบรรทัดที่ 9 โดยจะเปรียบเทียบค่าสนับสนุนกับค่า σ^{UD} ถ้าไอเท็มนั้นมีค่าสนับสนุนมากกว่า σ^{UD} ก็จะถูกเก็บไว้ในตัวแปร EF_k^{UD} ในรอบที่ $k=1$ นี้ ไอเท็มทุกตัวทั้งที่เป็น F_1^{DB}, EF_1^{DB} และไอเท็มที่ไม่ใช่ทั้ง F_1^{DB} และ EF_1^{DB} จะถูกเก็บไว้ใน C_1^{UD} ตามการทำงานในบรรทัดที่ 10

เมื่อเสร็จสิ้นการประมวลผลในส่วน Updating $1-itemset$ ผลลัพธ์ที่ได้คือ F_1^{UD}, EF_1^{UD} และ C_1^{UD} โดย F_1^{UD} จะถูกนำไปสร้างไอเท็มเซตคู่แข่งของ C_2^{UD} เป็นลำดับถัดไป ส่วน EF_1^{UD} และ C_1^{UD} จะถูกนำไปใช้ในการประมวลผลเมื่อมีข้อมูลชุดใหม่ชุดต่อไปเพิ่มเข้ามา

3.4.2.2 การสร้างไอเท็มเซตคู่แข่ง

การสร้างไอเท็มเซตคู่แข่งของงานวิจัยนี้ อยู่บนพื้นฐานหลักการสร้างไอเท็มเซตคู่แข่งของขั้นตอนวิธี อะพริโอริ แต่สิ่งที่แตกต่างจากขั้นตอนวิธี อะพริโอริ คือไอเท็มเซตที่จะนำมาเข้ากระบวนการเชื่อมไอเท็มเซต เพื่อให้ได้มาซึ่งไอเท็มเซตคู่แข่ง ในกรณีของ อะพริโอริ จะทำการเชื่อมไอเท็มเซตระหว่างไอเท็มเซตที่เกิดบ่อยขนาด $k-1$ ไอเท็มเซต กับ ไอเท็มเซตที่เกิดบ่อยขนาด $k-1$ ไอเท็มเซตด้วยกัน $(F_{k-1} * F_{k-1})$ แต่สำหรับขั้นตอนวิธีที่นำเสนอในงานวิจัยนี้จะใช้ไอเท็มเซตในการเชื่อมไอเท็มเซตเหมือนขั้นตอนวิธีอะพริโอริ แค่เพียงรอบที่ $k=2$ เท่านั้น เนื่องจากเราทราบค่าสนับสนุนที่แน่นอนของ $1-$ ไอเท็มเซตทุกตัว แสดงดังบรรทัดที่ $1-2$ ในรูปที่ 3.5 จากนั้น เมื่อได้แคนดิเดต $2-$ ไอเท็มเซต C_2^{db} เรียบร้อยแล้ว ขั้นตอนวิธีจะคำนวณหาแคนดิเดต $2-$ ไอเท็มเซตตัวใหม่ หรือ C_2^{new} ตามบรรทัดที่ 3 โดย C_2^{new} นี้ เป็นไอเท็มเซตที่ไม่ได้เป็นสมาชิกของ F_k^{DB} และ EF_k^{DB} ในฐานข้อมูลเดิม ดังนั้นค่าสนับสนุนที่ได้ในฐานข้อมูลใหม่จึงยังไม่เพียงพอที่จะระบุได้ว่า C_2^{new} จะมีโอกาสเป็น F_k^{UD} หรือ EF_k^{UD} ในฐานข้อมูลปรับปรุงหรือไม่ ดังนั้น C_2^{new} จะถูกนำไปพิจารณาใน Algorithm5 ดังรูปที่ 3.6 เพื่อพิจารณาว่าไอเท็มเซต C_2^{new} ดังกล่าวควรจะจัดเก็บให้อยู่ในตัวแปร *Temp_scanDB* หรือไม่ ซึ่ง *Temp_scanDB* จะถูกนำไปสแกนฐานข้อมูลเดิมในรอบสุดท้ายเพื่อปรับปรุงค่าสนับสนุนต่อไป

ตั้งแต่รอบที่ $k=3$ เป็นต้นไป การสร้างไอเท็มเซตคู่แข่งของงานวิจัยนี้จะนำไอเท็มเซตที่เป็น ไอเท็มเซตที่เกิดบ่อย และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิมและแคนดิเดต $k=1$ ไอเท็มเซตของฐานข้อมูลใหม่ มาตรวจสอบค่าสนับสนุนว่า มีค่ามากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำที่กำหนดหรือไม่ ดังบรรทัดที่ 6 หากมีค่ามากกว่าค่าสนับสนุนขั้นต่ำที่กำหนด ไอเท็มเซตนั้นๆ จะถูกเก็บไว้ในเซตชั่วคราว แทนด้วยสัญลักษณ์ FX_{k-1} ตามบรรทัดที่ 7 จากนั้นในบรรทัดที่ 8 จึงนำเอาไอเท็มเซตที่เป็นสมาชิกของ FX_{k-1} มาดำเนินการเชื่อมไอเท็มเซตเพื่อสร้างไอเท็มเซตคู่แข่งขึ้นมา

เมื่อได้ไอเท็มเซตคู่แข่งเรียบร้อยแล้ว ขั้นตอนวิธีจะคำนวณหาไอเท็มเซตคู่แข่งตัวใหม่ หรือ C_k^{new} ตามบรรทัดที่ 9 เพื่อนำไปใช้ในการหาไอเท็มเซตที่ควรจัดเก็บอยู่ในเซตของ *Temp_scanDB* ในขั้นตอนวิธีถัดไป

3.4.2.3 การปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซตและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซต เมื่อ $k \geq 2$

ในการปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซตและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซตในฐานข้อมูลปรับปรุง เมื่อ $k \geq 2$

มีกระบวนการทำงานตามรูปที่ 3.6 ซึ่งจะดำเนินการหลังจากที่ได้ไอเท็มเซตคู่แข่งซึ่งหาได้จาก Algorithm4 ในรูปที่ 3.5 ที่ได้อธิบายในหัวข้อก่อนหน้า

Algorithm4: 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 $C_2^{new} = \{X \in C_2^{db} \mid X \notin (F_2^{DB} \cup F_2^{DB})\}$

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 $FX_{k-1} = \{X \mid \sigma_x^{UD} \geq s \times |db|\}$

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

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

10 end if

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

รูปที่ 3.5 การสร้างไอเท็มเซตคู่แข่ง

การปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซตและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซตในฐานข้อมูลปรับปรุง สำหรับตั้งแต่รอบที่ 2 เป็นต้นไป ($k \geq 2$) การปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยจะเริ่มจาก การนำไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซต F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซต EF_k^{DB} ของฐานข้อมูลเดิม และ ไอเท็มเซตคู่แข่งใหม่ C_k^{new} ที่หาได้จาก Algorithm4 มาทำการสแกนในฐานข้อมูลข้อมูลใหม่เพื่อหาค่าสนับสนุน δ_x^{db} ตามบรรทัดที่ 1 จากนั้นจะทำการปรับปรุงค่าสนับสนุนให้แก่ไอเท็มเซตเป็นสมาชิกของไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซต F_k^{DB} และไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซต EF_k^{DB} ของฐานข้อมูลเดิม แต่ไม่เป็นสมาชิกของไอเท็มเซตคู่แข่งใหม่ C_k^{new} ดังบรรทัดที่ 2 – 4 จากนั้นจึงนำไอเท็มเซตที่ได้รับการปรับปรุงค่าสนับสนุนเรียบร้อยแล้วมาทดสอบกับค่าสนับสนุนขั้นต่ำเพื่อหาไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง F_k^{UD} ดังบรรทัดที่ 5 ส่วนไอเท็มตัวที่เหลือที่ไม่ได้เป็นไอเท็มเซตที่เกิด

บ่อยของฐานข้อมูลปรับปรุง จะถูกนำไปหาไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง EF_k^{UD} ในบรรทัดที่ 6 – 8 ซึ่งเป็นขั้นตอนเดียวกับการหาไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของ Algorithm1

ส่วนไอเท็มเซตที่เป็นสมาชิกของแคนดิเดตใหม่ C_k^{new} จะถูกนำไปคำนวณด้วย ค่าสนับสนุนบวกด้วย $\sigma^{DB}-1$ แล้วนำไปทดสอบกับค่าสนับสนุนขั้นต่ำที่ผู้ใช้กำหนด หากมีค่ามากกว่าหมายความว่าไอเท็มเซตนั้นมีโอกาสที่จะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง ดังนั้นไอเท็มเซตนี้จะถูกเก็บไว้ในตัวแปร *Temp_scanDB* เพื่อนำไปใช้ในการสแกนฐานข้อมูลเก่าอีกครั้งหนึ่ง หลังจากที่ได้ ไอเท็มเซตที่เกิดบ่อยและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยครบทุกตัวแล้ว ขั้นตอนนี้แสดงตามบรรทัดที่ 9 – 11

Algorithm5: Updating k-itemset

Input: $DB, db, \Pr ob_{PL}, s, F_k^{DB}, EF_k^{DB}, \sigma^{DB}$

Output: $F_k^{UD}, EF_k^{UD}, C_1^{UD}, \Pr ob_{EF}^{UD}$

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

4 endif

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

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

7 $\quad EF_1^{UD} = \{X \mid s \times |UD| > \delta_x^{UD} \geq \sigma^{UD}\}$

8 end for

9 for $X \notin (F_1^{DB} \cup EF_1^{DB})$ and $X \in C_k^{new}$

10 $\quad Temp\_scanDB = \{X \mid (\delta_x^{db} + (\sigma^{DB} - 1)) \geq s \times |UD|\}$

11 end for

12 Return $F_k^{UD}, EF_k^{UD}, Temp\_scanDB$

รูปที่ 3.6 การปรับปรุงค่าสนับสนุนของไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซตและไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยขนาด $k-$ ไอเท็มเซต เมื่อ $k \geq 2$

3.4.2.4 การสแกนฐานข้อมูลเดิมซ้ำ

จาก Algorithm 2 – 5 ดังรูปที่ 3.3 – 3.6 ซึ่งเป็นขั้นตอนวิธีของการค้นหาไอเท็มเซตที่เกิดบ่อยและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง เมื่อ Algorithm5 ทำงานเสร็จสิ้น จะได้ ไอเท็มเซตที่เกิดบ่อย F_k^{UD} และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย EF_k^{UD} ของฐานข้อมูลปรับปรุง ที่ปรับปรุงเรียบร้อยในระดับหนึ่ง ยังคงเหลือการหาไอเท็มเซตที่เกิดบ่อยและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยครั้งสุดท้ายจากไอเท็มเซตที่ถูกเก็บไว้ในตัวแปร *Temp_scanDB* ไอเท็มเซตดังกล่าว เป็นไอเท็มเซตชุดใหม่ที่เกิดขึ้นในฐานข้อมูลใหม่ ที่คาดว่ามีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยหรือไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย

ใน Algorithm6 แสดงดังรูปที่ 3.7 จะนำเสนอขั้นตอนการนำไอเท็มเซตที่อยู่ในตัวแปร *Temp_scanDB* มาสแกนฐานข้อมูลเดิม เพื่อปรับปรุงค่าสนับสนุน ซึ่งในการสแกนฐานข้อมูลเดิมนี้จะกระทำเพียงครั้งเดียว หลังจาก Algorithm 2 – 5 ทำงานครบ k รอบเป็นที่เรียบร้อยแล้ว ไอเท็มเซตที่เป็นสมาชิกของ *Temp_scanDB* จะถูกนำไปสแกนในฐานข้อมูลเพื่อหาค่าสนับสนุนตามบรรทัดที่ 1 จากนั้นค่าสนับสนุนจะถูกปรับปรุง ตามบรรทัดที่ 2 เมื่อได้ค่าสนับสนุนที่ปรับปรุงแล้ว ไอเท็มทุกตัวจะถูกนำไปทดสอบกับค่าสนับสนุนขั้นต่ำเพื่อหาไอเท็มเซตที่เกิดบ่อยใหม่ของฐานข้อมูลปรับปรุง F_k^{UD} ตามบรรทัดที่ 3 และ ไอเท็มตัวที่เหลือจะถูกนำไปทดสอบกับค่าคาดหวังน้อยที่สุดที่คาดว่าไอเท็มเซตจะกลายเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม หรือ σ^{UD} เพื่อค้นหาไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุง F_k^{UD} ตามบรรทัดที่ 4

Algorithm6: Rescanning original database

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

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

1 scan DB for all $DB \in Temp\_scanDB$ to obtain δ_x^{DB}

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

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

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

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

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

รูปที่ 3.7 การสแกนฐานข้อมูลเดิมซ้ำ

ในบรรทัดที่ 5 และ 6 จะเป็นการนำเอาไอเท็มเซตที่เกิดบ่อยใหม่และไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยใหม่ของฐานข้อมูลปรับปรุงผนวกเข้ากับตัวแปรไอเท็มเซตที่เกิดบ่อยและไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยเดิม ตามลำดับ ซึ่งตัวแปรเหล่านี้จะถูกนำไปใช้อีกครั้งเมื่อมีการเพิ่มข้อมูลชุดใหม่ชุดถัดไปเข้ามา

บทที่ 4
การทดลองและวิเคราะห์ผลการทดลอง

การค้นหากฎความสัมพันธ์ (Association rule discovery) เป็นเทคนิคหนึ่งที่สำคัญในการทำเหมืองข้อมูล เพื่อค้นหารูปแบบความสัมพันธ์ระหว่างไอเท็มในฐานข้อมูล ผลลัพธ์ที่ได้จากการประมวลผลจะอยู่ในรูปของกฎความสัมพันธ์ ถ้า....แล้ว.... (IF X THEN Y) มีหลักการทำงาน 2 ขั้นตอน คือการค้นหาไอเท็มเซตที่เกิดบ่อย และ การสร้างกฎความสัมพันธ์

การสร้างกฎความสัมพันธ์ เป็นขั้นตอนที่ง่ายต่อการทดสอบว่าไอเท็มเซตที่เกิดบ่อยชุดใดที่สามารถนำมาสร้างกฎความสัมพันธ์ และมีคุณสมบัติเป็นกฎที่แข็งแรง (Strong rule) โดยเปรียบเทียบกับค่าความเชื่อมั่นขั้นต่ำ (Minimum confidence) ที่กำหนดโดยผู้ใช้ ในขณะที่การค้นหาไอเท็มเซตที่เกิดบ่อยเป็นขั้นตอนที่ซับซ้อนและใช้เวลาในการค้นหาว่าความสัมพันธ์ของไอเท็มเซ็ตชุดใดที่สามารถเป็นไอเท็มเซตที่เกิดบ่อยได้โดยเปรียบเทียบกับค่าสนับสนุนขั้นต่ำ (Minimum support) ที่กำหนดโดยผู้ใช้ ดังนั้น ในงานวิจัยทางด้านการค้นหากฎความสัมพันธ์ส่วนใหญ่ จะศึกษาและนำเสนอแนวคิดเกี่ยวกับการค้นหาไอเท็มเซตที่เกิดบ่อย เช่นเดียวกับขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติ ที่เน้นการค้นหาไอเท็มเซตที่เกิดบ่อยเป็นหลัก

ฐานข้อมูลทั่วไปโดยเฉพาะฐานข้อมูลรายการธุรกรรม (Transaction data) จะมีการเคลื่อนไหวอยู่ตลอดเวลา นั่นคือมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิมเรื่อยๆ ซึ่งนอกจากจะทำให้ฐานข้อมูลเดิมมีขนาดใหญ่ขึ้นกว่าเดิม ยังส่งผลให้ไอเท็มเซตที่เกิดบ่อยที่เคยคำนวณได้ก่อนการเพิ่มชุดข้อมูลใหม่อาจมีการเปลี่ยนแปลง เช่น เมื่อปรับปรุงฐานข้อมูลให้เป็นปัจจุบันแล้วอาจพบว่า ไอเท็มเซตที่เกิดบ่อยที่ค้นพบในฐานข้อมูลเดิมอาจจะยังคงเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงหรืออาจจะไม่ใช่ไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุงก็ได้ ในขณะเดียวกันไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลเดิม อาจจะยังคงเป็นไอเท็มเซตที่ไม่สามารถเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงหรืออาจจะเปลี่ยนเป็นไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุงก็ได้เช่นกัน ซึ่งการเปลี่ยนแปลงสถานะของไอเท็มเซตดังกล่าวจะกระทบต่อกฎความสัมพันธ์ที่ถูกสร้างจากไอเท็มเซตเหล่านี้ ดังนั้นจึงมีนักวิจัยหลายรายพยายามคิดค้นแนวคิดในการปรับปรุงกฎความสัมพันธ์ให้เป็นปัจจุบันโดยหลักการที่ว่า การจะปรับปรุงกฎความสัมพันธ์ให้เป็นปัจจุบันให้ได้ จะต้องปรับปรุงไอเท็มเซตที่เกิดบ่อยให้เป็นปัจจุบันเสียก่อน ซึ่งการปรับปรุงไอเท็มเซตที่เกิดบ่อยเป็นกระบวนการที่ยุ่งยากและซับซ้อนกว่าการสร้างกฎความสัมพันธ์

ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติ เป็นอีกแนวคิดหนึ่งสำหรับการเพิ่มขยายกฎความสัมพันธ์ โดยนำหลักการประมาณค่าความน่าจะเป็นด้วยการแจกแจงปกติมาประยุกต์ใช้เพื่อค้นหาและปรับปรุงไอเท็มเซตที่เกิดบ่อยให้มีความทันสมัยโดยลดขั้นตอนการสแกนฐานข้อมูลเดิม และแก้ไขปัญหาข้อจำกัดด้านการจำกัดขนาดของฐานข้อมูลใหม่ที่จะถูกเพิ่มเข้ามา เนื่องจากในความเป็นจริง ข้อมูลชุดใหม่ที่ถูกเพิ่มเข้ามาในแต่ละครั้งอาจจะมีจำนวนไม่เท่ากัน

ในบทนี้ จะอธิบายถึงการทดลองประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติในด้านความถูกต้องและเวลาที่ใช้ในการประมวลผล (Execution time) โดยเปรียบเทียบกับการทำงานของ 5 ขั้นตอนวิธี ได้แก่ ขั้นตอนวิธีอะพริโอริ ขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

4.1 วัตถุประสงค์ของการทดลอง

เพื่อทดสอบและแสดงให้เห็นถึงประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติเมื่อมีการเพิ่มชุดข้อมูลใหม่เข้ามาในฐานข้อมูลเดิม ผู้วิจัยได้กำหนดวัตถุประสงค์ของการทดลอง ดังนี้

1. เพื่อทดสอบความถูกต้องของการค้นหาไอเท็มเซตที่เกิดบ่อยเมื่อข้อมูลชุดใหม่ถูกเพิ่มเข้าไปในฐานข้อมูลเดิม

การค้นหากฎความสัมพันธ์ ประกอบด้วย 2 ขั้นตอนหลัก ได้แก่ การค้นหาไอเท็มเซตที่เกิดบ่อยและการสร้างกฎความสัมพันธ์ ซึ่งการที่จะสร้างกฎความสัมพันธ์ได้นั้น จะต้องมีการค้นหาไอเท็มเซตที่เกิดบ่อยให้ได้ก่อนจึงจะสามารถสร้างกฎความสัมพันธ์ได้ ดังนั้นการศึกษาด้านการค้นหากฎความสัมพันธ์โดยไปจึงนิยมนำเสนอเกี่ยวกับการค้นหาไอเท็มเซตที่เกิดบ่อยเป็นหลัก เนื่องจากเป็นกระบวนการที่ซับซ้อนและใช้เวลานาน ขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติเป็นขั้นตอนวิธีที่ศึกษาการค้นหาไอเท็มเซตที่เกิดบ่อยเช่นเดียวกัน

ลักษณะการทำงานของขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติมีการประมวลผลด้วยหลักการพื้นฐานของขั้นตอนวิธีอะพริโอริ ดังนั้น การทดสอบความถูกต้องของไอเท็มเซตที่เกิดบ่อยที่ค้นหาได้ จะนำมาเปรียบเทียบกับผลลัพธ์ที่ได้จากการประมวลผลด้วยขั้นตอนวิธีอะพริโอริ

2. เพื่อวัดประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติ

นอกจากการทดสอบความถูกต้องในการค้นหาไอเท็มเซตที่เกิดบ่อยที่ค้นหาได้โดยเปรียบเทียบกับผลที่ได้จากขั้นตอนวิธีอะพริโอริ แล้ว การทดสอบประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติจะพิจารณาจากเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธี โดยมีการเปรียบเทียบกับเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ที่มีการทำงานโดยใช้หลักการพื้นฐานของขั้นตอนวิธีอะพริโอริ เช่นกัน ได้แก่ ขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

4.2 วิธีการทดลอง

ด้วยการเพิ่มขยายกฎความสัมพันธ์ เป็นการปรับปรุงกฎความสัมพันธ์ให้เป็นปัจจุบันเมื่อมีข้อมูลชุดใหม่ถูกเพิ่มเข้ามาในฐานข้อมูลเดิม ซึ่งจะมีผลให้กฎความสัมพันธ์เดิมอาจจะไม่ใช่กฎความสัมพันธ์ของฐานข้อมูลปรับปรุงอีกต่อไป ในขณะที่กฎความสัมพันธ์ใหม่อาจจะเกิดขึ้นมาทดแทนกฎความสัมพันธ์เดิม

ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติที่นำเสนอในงานวิจัยนี้ มีวัตถุประสงค์เพื่อออกแบบขั้นตอนวิธีที่สามารถเพิ่มขยายกฎความสัมพันธ์ได้โดยไม่จำกัดขนาดของหรือไม่จำเป็นต้องทราบจำนวนรายการธุรกรรมของข้อมูลชุดใหม่ที่จะถูกเพิ่มเข้ามา ดังนั้น การทดลองประสิทธิภาพและความถูกต้องของขั้นตอนวิธีที่นำเสนอจึงได้รับการออกแบบมาเพื่อทดลองภายใต้สมมติฐานที่แตกต่างกัน โดยจะทดลองกับชุดข้อมูลสังเคราะห์จำนวน 3 ชุด ที่สร้างขึ้นมาด้วยหลักการสร้างชุดข้อมูลสังเคราะห์ของ Agrawal [7] ซึ่งจะมีการกำหนดค่าพารามิเตอร์ในการสร้างชุดข้อมูลให้เหมาะสมกับสมมติฐานการทดลองที่แตกต่างกัน ดังนี้

4.2.1 การทดลองที่ 1: การทดลองการเพิ่มขยายกฎความสัมพันธ์ในกรณีที่ข้อมูลชุดใหม่ที่เพิ่มเข้าไปมีขนาดเท่ากัน

การทดลองที่ 1 ได้ถูกออกแบบมาเพื่อทดสอบความถูกต้องและประสิทธิภาพของขั้นตอนวิธีการเพิ่มขยายความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ข้อมูลชุดใหม่ที่ถูกเพิ่มเข้ามามีขนาดเท่ากันในทุกๆ ครั้งที่มีการเพิ่มข้อมูลใหม่เข้าไป

เพื่อทดสอบว่าประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติสามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพ ผู้วิจัยได้สร้างชุดข้อมูลสังเคราะห์โดยกำหนดค่าพารามิเตอร์ต่างๆ ที่แตกต่างกัน ดังนี้

กำหนดให้

$|T|$ หมายถึง ค่าเฉลี่ยความยาวของรายการธุรกรรม

$|I|$ หมายถึง ค่าเฉลี่ยของขนาดสูงสุดที่จะเป็นไอเท็มเซตที่เกิดบ่อย (Maximal frequent itemset: MFI)

$|L|$ หมายถึง จำนวนชุดของ MFI ที่จะถูกสุ่มหยิบเพื่อสร้างไอเท็มในแต่ละรายการธุรกรรม

N หมายถึง จำนวนไอเท็ม

ในงานวิจัยนี้ได้จำลองชุดข้อมูลเพื่อทดสอบความถูกต้องและประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการแจกแจงปกติ จำนวน 3 ชุด ดังนี้

ชุดข้อมูล A I4T10N200

เป็นชุดข้อมูลที่ถูกสร้างขึ้นมาจากไอเท็มจำนวน (N) 200 ตัว ซึ่งจะถูกสุ่มเลือกเพื่อนำไปสร้างเป็นชุดของจำนวนที่จะสามารถเป็นไอเท็มเซตที่เกิดบ่อยได้สูงสุด (I) ที่มีความยาวเฉลี่ยเท่ากับ 4 จำนวน 20 ชุด และมีค่าเฉลี่ยความยาวของรายการธุรกรรม (T) เท่ากับ 10

ชุดข้อมูล B I10T10N200

เป็นชุดข้อมูลที่ถูกสร้างขึ้นมาจากไอเท็มจำนวน (N) 200 ตัว ซึ่งจะถูกสุ่มเลือกเพื่อนำไปสร้างเป็นชุดของจำนวนที่จะสามารถเป็นไอเท็มเซตที่เกิดบ่อยได้สูงสุด (I) ที่มีความยาวเฉลี่ยเท่ากับ 10 จำนวน 20 ชุด และมีค่าเฉลี่ยความยาวของรายการธุรกรรม (T) เท่ากับ 10

ชุดข้อมูล C I10T15N200

เป็นชุดข้อมูลที่ถูกสร้างขึ้นมาจากไอเท็มจำนวน (N) 200 ตัว ซึ่งจะถูกสุ่มเลือกเพื่อนำไปสร้างเป็นชุดของจำนวนที่จะสามารถเป็นไอเท็มเซตที่เกิดบ่อยได้สูงสุด (I) ที่มีความยาวเฉลี่ยเท่ากับ 10 จำนวน 20 ชุด และมีค่าเฉลี่ยความยาวของรายการธุรกรรม (T) เท่ากับ 15

ในแต่ละชุดข้อมูลที่นำมาใช้ในการวัดประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ จะประกอบด้วยจำนวนรายการธุรกรรมของฐานข้อมูลเดิม จำนวน 100,000 รายการธุรกรรม และจำนวนรายการธุรกรรมของฐานข้อมูลใหม่ที่ถูกเพิ่มเข้ามาจำนวน 4 ชุด ได้แก่ 1%, 3%, 5% และ 10% หรือด้วยขนาด 1000, 3000, 5000 และ 10000 รายการธุรกรรมตามลำดับ

การทดลอง จะทดสอบการประมวลผลของขั้นตอนวิธีที่นำเสนอเมื่อกำหนดให้ฐานข้อมูลเดิมมีจำนวน 100,000 รายการธุรกรรม และมีการเพิ่มชุดข้อมูลใหม่เข้าไปจำนวน 1 ชุด ด้วยค่าสนับสนุนขั้นต่ำจำนวน 4 ค่า ได้แก่ 3%, 5%, 7% และ 10% โดยจะทดสอบเปรียบเทียบประสิทธิภาพกับการทำงานของขั้นตอนวิธีที่ทำงานด้วยหลักการเดียวกับขั้นตอนวิธีอะพริโอริ ได้แก่ ขั้นตอนวิธีอะพริโอริ ขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

นอกจากนี้ ยังได้มีการทดสอบประสิทธิภาพการทำงานของขั้นตอนวิธีที่นำเสนอด้วยการประมวลผลเมื่อกำหนดให้ฐานข้อมูลเดิมมีจำนวน 100,000 รายการธุรกรม และมีข้อมูลชุดใหม่ขนาดเท่ากัน คือ 1000 รายการธุรกรรมถูกเพิ่มเข้าไปอย่างต่อเนื่องจำนวน 10 ครั้ง ของข้อมูลแต่ละชุดที่ค่าสนับสนุน 3% และทำการทดสอบกับข้อมูลจริง Tafeng ที่มีจำนวนฐานข้อมูลเดิม 50,000 รายการธุรกรรม และเพิ่มข้อมูลเข้าไปใหม่ 5,000 รายการธุรกรรม ที่ค่าสนับสนุน 3%

การทดสอบชุดข้อมูลต่างๆ ดังกล่าวเบื้องต้น จะทำการเปรียบเทียบขั้นตอนวิธีการเพิ่มขยายความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติโดยทำการเปรียบเทียบกับประสิทธิภาพการทำงานของขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นซึ่งในการเปรียบเทียบการทดลองทั้ง 2 แบบจะพิจารณาจาก 3 ปัจจัย ดังนี้

1. จำนวนความถูกต้องของไอเท็มเซตที่เกิดบ่อยที่ค้นพบเปรียบเทียบกับจำนวนไอเท็มเซตที่เกิดบ่อยที่ได้จากการค้นหาโดยขั้นตอนวิธี อะพริโอริ

2. จำนวนไอเท็มเซตคู่แข่งที่ไม่ได้เป็นไอเท็มเซตที่เกิดบ่อยที่ถูกเก็บเพื่อนำไปประมวลผลในรอบของการเพิ่มข้อมูลชุดใหม่รอบถัดไป โดยจะมี 4 ขั้นตอนวิธีที่พิจารณาในปัจจัยข้อนี้ ได้แก่ ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ ขั้นตอนวิธี พรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

3. พิจารณาจำนวนไอเท็มเซตคู่แข่งของฐานข้อมูลปรับปรุงที่ถูกเก็บไว้เพื่อนำไปสแกนในฐานข้อมูลเดิมรอบสุดท้าย โดยจะมี 3 ขั้นตอนวิธีที่พิจารณาในปัจจัยข้อนี้ ได้แก่ ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

4.2.2 การทดลองที่ 2: การทดลองการเพิ่มขยายกฎความสัมพันธ์เมื่อข้อมูลชุดใหม่ที่เพิ่มเข้าไปมีขนาดแตกต่างกัน

การทดลองที่ 2 นี้ ออกแบบมาเพื่อทดสอบความถูกต้องและวัดประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติในกรณีที่ข้อมูลชุดใหม่ที่ถูกเพิ่มเข้ามามีขนาดที่แตกต่างกัน

เพื่อทดสอบประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติสามารถทำงานได้อย่างถูกต้องและมีประสิทธิภาพในกรณีที่ข้อมูลชุดใหม่ที่ถูกเพิ่มเข้ามามีขนาดที่แตกต่างกัน ผู้วิจัยได้ทดสอบโดยใช้ข้อมูลสังเคราะห์ชุดเดิมจำนวน 3 ชุด ดังกล่าวข้างต้น โดยชุดข้อมูลแต่ละชุดจะกำหนดให้ฐานข้อมูลเดิมมีจำนวน 100,000 รายการธุรกรรม ทดสอบด้วยค่าสนับสนุน 3% และจะมีการกำหนดให้มีการเพิ่มข้อมูลเข้าไปจำนวน 10 ครั้ง แต่ละครั้งจะมีการเพิ่มข้อมูลเข้าไปดังนี้

ข้อมูลชุดที่ 1	มีขนาด	1%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		1,000
ข้อมูลชุดที่ 2	มีขนาด	2%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		2,000
ข้อมูลชุดที่ 3	มีขนาด	3%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		3,000
ข้อมูลชุดที่ 4	มีขนาด	4%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		4,000
ข้อมูลชุดที่ 5	มีขนาด	5%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		5,000
ข้อมูลชุดที่ 6	มีขนาด	6%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		6,000
ข้อมูลชุดที่ 7	มีขนาด	7%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		7,000
ข้อมูลชุดที่ 8	มีขนาด	8%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		8,000
ข้อมูลชุดที่ 9	มีขนาด	9%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		9,000
ข้อมูลชุดที่ 10	มีขนาด	10%	ของฐานข้อมูลเดิมจำนวนรายการธุรกรรม		10,000

ในการทดลองที่ 2 นี้ จะวัดประสิทธิภาพการทำงานโดยเปรียบเทียบกับ 3 ขั้นตอนวิธีได้แก่ ขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ เนื่องจากเป็นขั้นตอนวิธีที่สามารถค้นหาไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุงใหม่ได้โดยไม่ต้องคำนึงถึงขนาดของข้อมูลชุดใหม่ ซึ่งไม่จำเป็นเตรียมข้อมูลในกระบวนการของการประมวลผลฐานข้อมูลเดิมเช่นเดียวกับขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

4.3 ผลการทดลองและการวิเคราะห์ผลการทดลอง

ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติถูกออกแบบและพัฒนาขั้นตอนวิธีด้วยโปรแกรม MATLAB R2013b ในการทดสอบความถูกต้องและวัดประสิทธิภาพของขั้นตอนวิธีที่นำเสนอ ผู้วิจัยได้ทดสอบการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติด้วยชุดข้อมูลสังเคราะห์ทั้ง 3 ชุด ได้แก่ ชุดข้อมูล I4T10N200 I10T10N200 และ I10T15N200 ดังกล่าวข้างต้น รวมถึงข้อมูลจริงชุด Tafeng ด้วยค่าสนับสนุนที่แตกต่างกัน

การทดสอบประสิทธิภาพในการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ผู้วิจัยได้ออกแบบการทดลองเพื่อวัดประสิทธิภาพและความถูกต้องของการทำงานออกเป็น 2 การทดลองได้แก่

1. การทดลองการเพิ่มขยายกฎความสัมพันธ์ในกรณีที่ข้อมูลชุดใหม่ที่เพิ่มเข้าไปมีขนาดเท่ากัน

2. การทดลองการเพิ่มขยายกฎความสัมพันธ์ในกรณีที่ข้อมูลชุดใหม่ที่เพิ่มเข้าไปมีขนาดแตกต่างกัน

ในการทดสอบความถูกต้องในการประมวลผลของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติจะประเมินโดยการเปรียบเทียบกับผลลัพธ์การค้นหาไอเท็มเซตที่เกิดบ่อยของขั้นตอนวิธีวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติเปรียบเทียบกับผลลัพธ์ที่ได้จากขั้นตอนวิธีอะพริโอริ

ส่วนการประเมินประสิทธิภาพในการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติจะวัดจากเวลาที่ใช้ในการประมวลผลเปรียบเทียบกับขั้นตอนวิธีอื่นๆ ที่มีการทำงานโดยหลักการพื้นฐานของขั้นตอนวิธีอะพริโอริ อีก 4 ขั้นตอนวิธี ได้แก่ ขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นผลการทดลองเป็นดังนี้

ตารางที่ 4.1 จำนวนไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงชุด I4T10N200 ที่ค่าสนับสนุน 3% 5% 7% และ 10 % ด้วยขนาดข้อมูลชุดใหม่ 1% 3% 5% และ 10%

$\|db\|$	sup	จำนวนไอเท็มเซตที่เกิดบ่อย							
		$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
1,000 (1%)	10%	30	136	201	96	26	3	-	-
	7%	33	222	385	302	88	11	1	-
	5%	36	304	734	672	316	50	2	-
	3%	45	424	1411	1892	1334	491	65	3
3,000 (3%)	10%	30	136	201	97	26	3	-	-
	7%	33	222	384	301	88	11	1	-
	5%	36	304	732	672	314	50	2	-
	3%	45	424	1410	1896	1335	492	66	3
5,000 (5%)	10%	30	136	201	96	26	3	-	-
	7%	33	222	383	301	88	11	1	-
	5%	36	304	734	670	314	50	2	-
	3%	45	424	1409	1890	1328	491	66	3
10,000 (10%)	10%	30	136	202	97	26	3	-	-
	7%	33	222	383	301	87	11	1	-
	5%	36	304	731	673	315	50	2	-
	3%	45	425	1410	1892	1331	493	67	3

ตารางที่ 4.2 จำนวนไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงชุด I10T10N200 ที่ค่าสนับสนุน 3% 5% 7% และ 10 % ด้วยขนาดข้อมูลชุดใหม่ 1% 3% 5% และ 10%

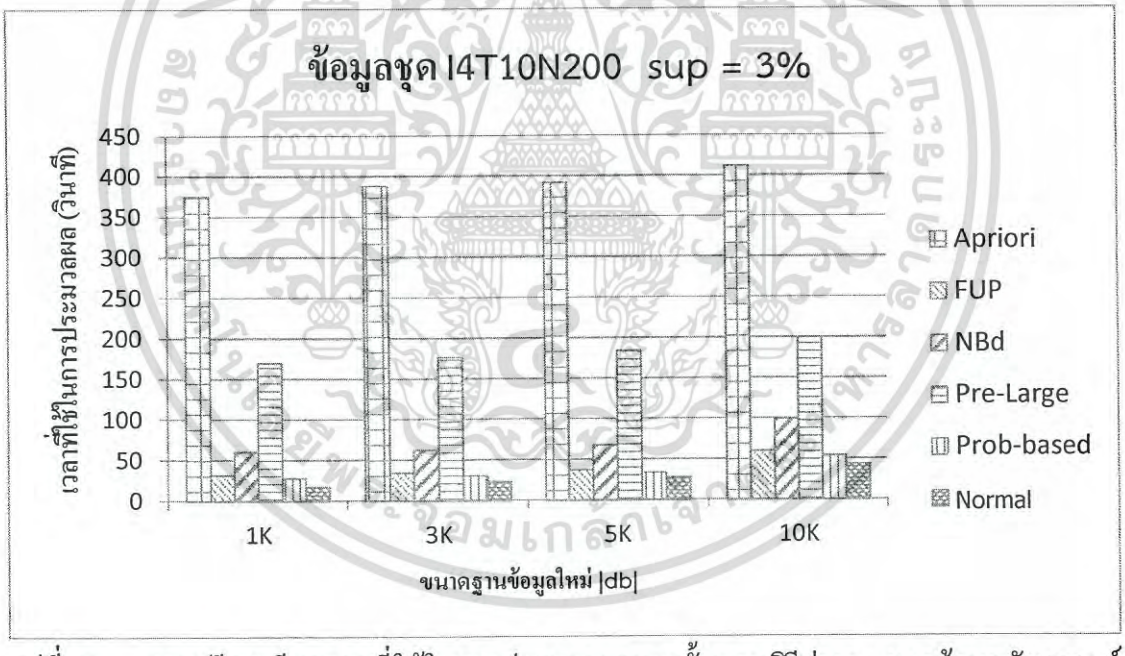
$\|db\|$	sup	จำนวนไอเท็มเซตที่เกิดบ่อย								
		$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$
1,000 (1%)	10%	37	46	57	70	56	28	8	1	-
	7%	47	126	75	71	56	28	8	1	-
	5%	54	195	332	109	61	28	8	1	-
	3%	75	361	552	898	1106	186	72	18	2
3,000 (3%)	10%	37	47	57	70	56	28	8	1	-
	7%	47	126	76	71	56	28	8	1	-
	5%	54	195	333	110	61	28	8	1	-
	3%	75	361	552	898	1118	186	72	18	2
5,000 (5%)	10%	37	47	58	70	56	28	8	1	-
	7%	47	127	77	71	56	28	8	1	-
	5%	54	195	332	110	61	28	8	1	-
	3%	75	361	552	898	1116	186	72	18	2
10,000 (10%)	10%	37	47	57	70	56	28	8	1	-
	7%	47	128	77	71	56	28	8	1	-
	5%	54	195	333	110	61	28	8	1	-
	3%	75	361	552	898	1125	186	72	18	2

ตารางที่ 4.3 จำนวนไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงชุด I10T15N200 ที่ค่าสนับสนุน 3% 5% 7% และ 10 % ด้วยขนาดข้อมูลชุดใหม่ 1% 3% 5% และ 10%

$\|db\|$	sup	จำนวนไอเท็มเซตที่เกิดบ่อย										
		$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	$k=11$
1,000 (1%)	10%	63	188	87	24	1	-	-	-	-	-	-
	7%	68	426	432	90	12	-	-	-	-	-	-
	5%	82	771	912	782	317	219	-	-	-	-	-
	3%	85	1096	3363	4195	2050	968	429	218	65	12	1
3,000 (3%)	10%	63	186	87	24	1	-	-	-	-	-	-
	7%	68	427	433	90	12	-	-	-	-	-	-
	5%	82	772	916	782	317	219					
	3%	85	1096	3367	4193	2045	970	429	218	65	12	1
5,000 (5%)	10%	63	187	87	25	1	-	-	-	-	-	-
	7%	68	430	433	91	12	-	-	-	-	-	-
	5%	82	771	906	787	317	219	-	-	-	-	-
	3%	85	1097	3373	4206	2047	973	429	218	65	12	1
10,000 (10%)	10%	63	187	87	25	1	-	-	-	-	-	-
	7%	68	428	442	91	12	-	-	-	-	-	-
	5%	82	772	914	790	322	219	-	-	-	-	-
	3%	85	1097	3373	4226	2049	977	429	218	65	12	1

ตารางที่ 4.4 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 3%

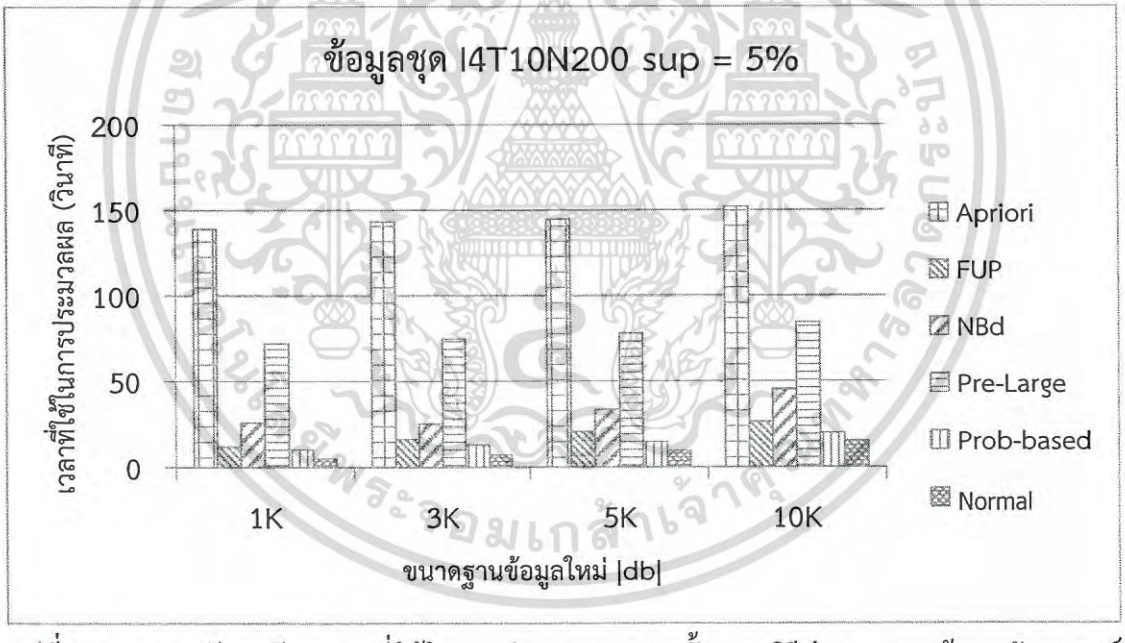
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I4T10N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
3%	1%	375.66	31.67	60.53	169.90	28.07	17.34
	3%	388.68	34.51	62.26	176.84	30.50	23.32
	5%	392.08	37.36	68.08	184.34	34.11	28.09
	10%	412.49	60.01	99.30	200.44	54.13	43.77



รูปที่ 4.1 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 3%

ตารางที่ 4.5 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 5%

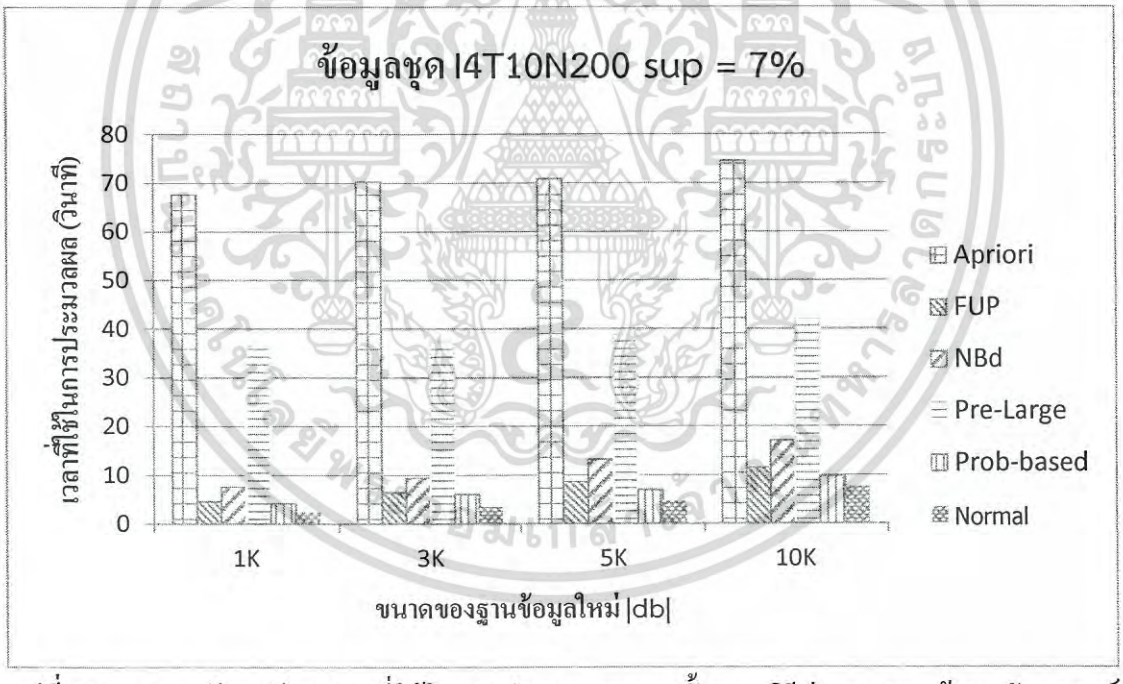
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I4T10N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
5%	1%	139.13	12.30	26.17	72.16	10.53	5.30
	3%	143.51	16.62	25.47	74.92	13.12	7.55
	5%	145.04	20.85	33.67	78.17	15.13	9.82
	10%	152.05	26.50	45.49	84.67	20.18	15.72



รูปที่ 4.2 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 5%

ตารางที่ 4.6 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 7%

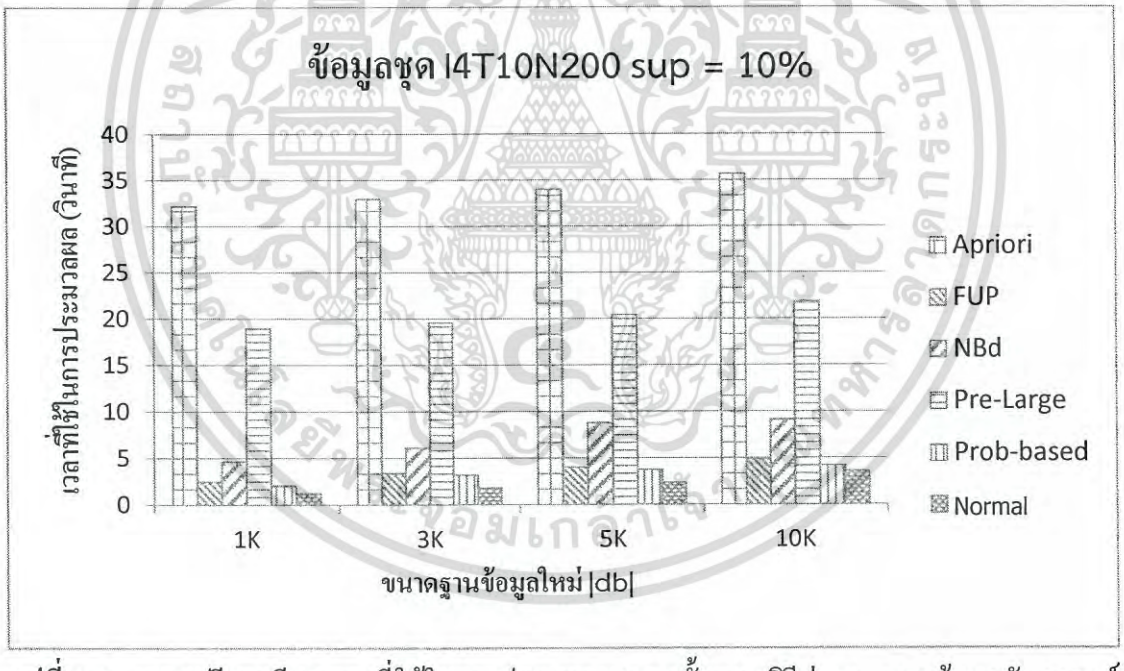
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I4T10N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
7%	1%	67.60	4.58	7.63	36.40	4.17	2.51
	3%	70.28	6.44	9.47	37.54	6.12	3.47
	5%	70.90	8.71	13.28	38.91	6.97	4.74
	10%	74.63	11.51	17.09	41.97	9.75	7.62



รูปที่ 4.3 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 7%

ตารางที่ 4.7 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 10%

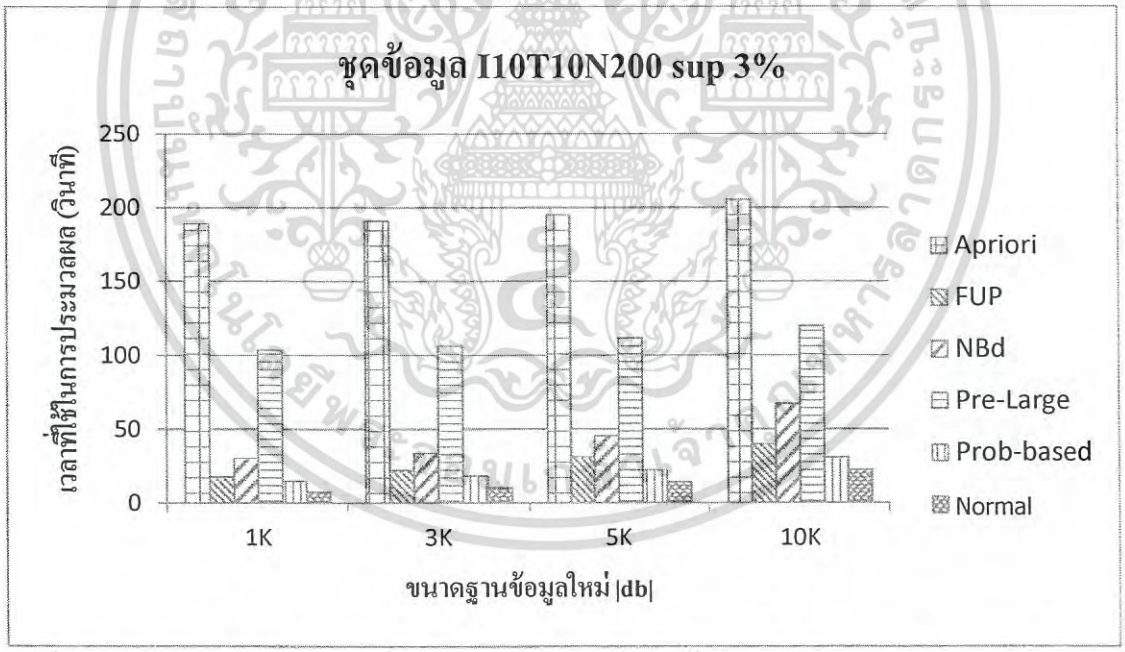
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I4T10N200							
min_sup	|*db*|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
10%	1%	32.21	2.49	4.66	18.95	2.06	1.23
	3%	32.99	3.42	6.12	19.59	3.19	1.81
	5%	34.05	4.05	8.83	20.45	3.76	2.36
	10%	35.71	4.88	9.12	21.87	4.21	3.63



รูปที่ 4.4 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 10%

ตารางที่ 4.8 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 3%

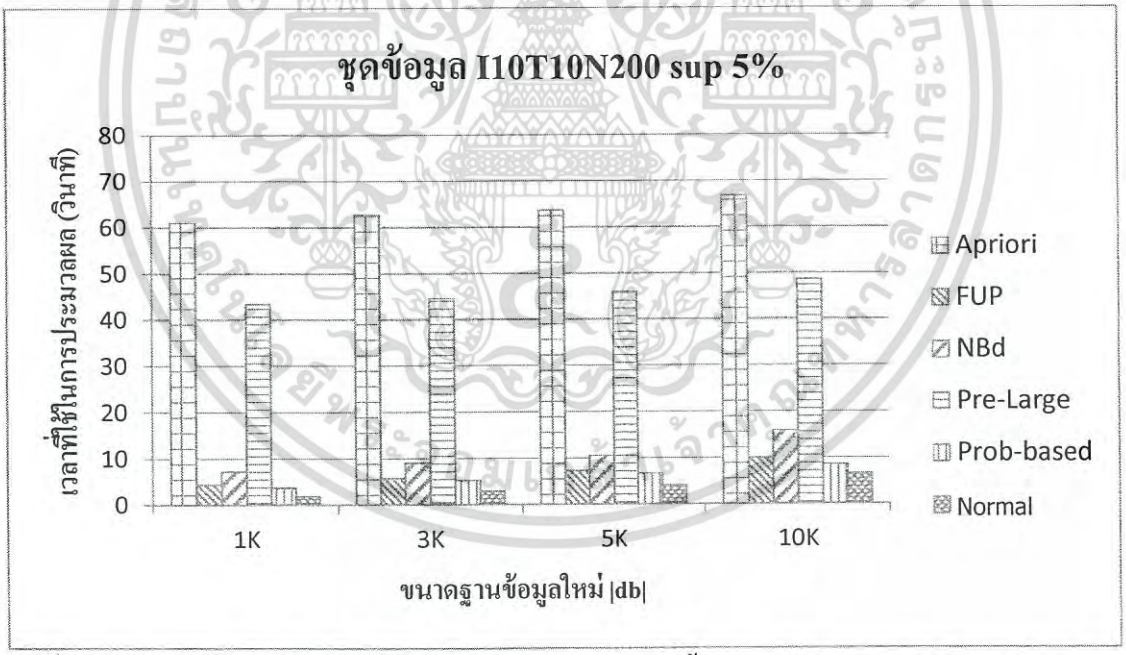
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T10N200							
min_sup	|*db*|	Algorithm					
		Apriori	**FUP**	**NBd**	**Pre-Large**	**Prob-based**	**Normal**
3%	1%	189.21	17.97	30.61	103.67	15.16	7.49
	3%	191.34	22.54	33.91	106.73	18.72	10.83
	5%	195.40	31.38	45.70	111.94	22.56	14.54
	10%	205.48	40.20	67.36	119.95	31.06	22.77



รูปที่ 4.5 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 3%

ตารางที่ 4.9 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 5%

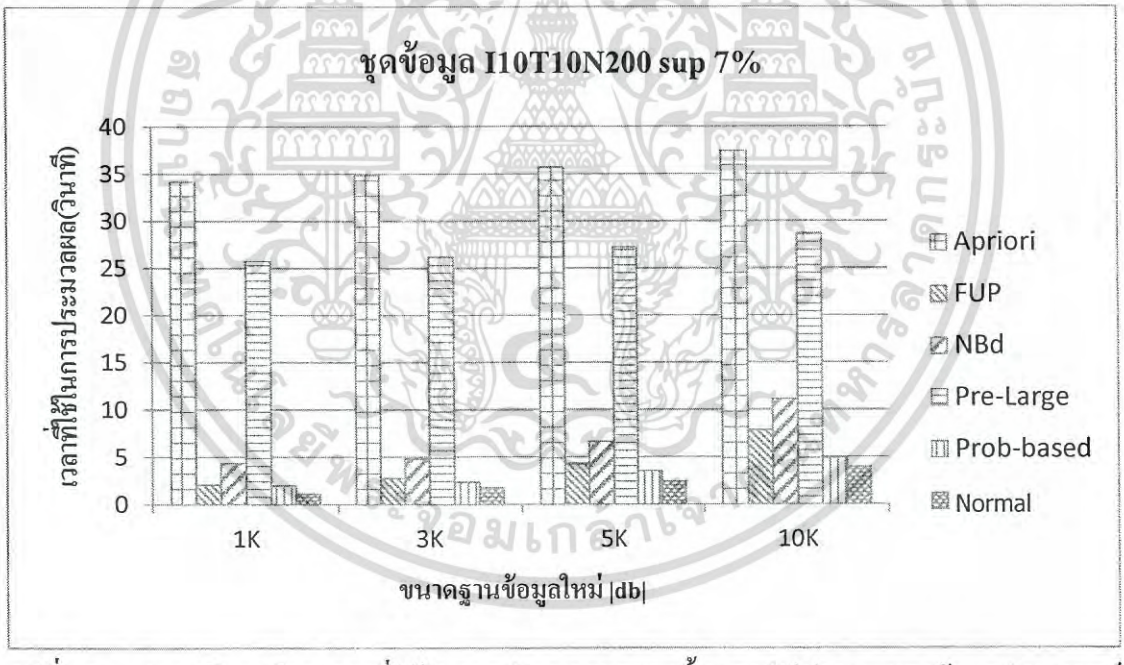
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T10N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
5%	1%	61.05	4.49	7.38	43.39	3.79	1.94
	3%	62.67	5.74	9.08	44.58	5.34	2.97
	5%	63.74	7.32	10.61	45.95	6.75	4.25
	10%	66.79	10.02	15.92	48.65	8.58	6.73



รูปที่ 4.6 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 5%

ตารางที่ 4.10 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 7%

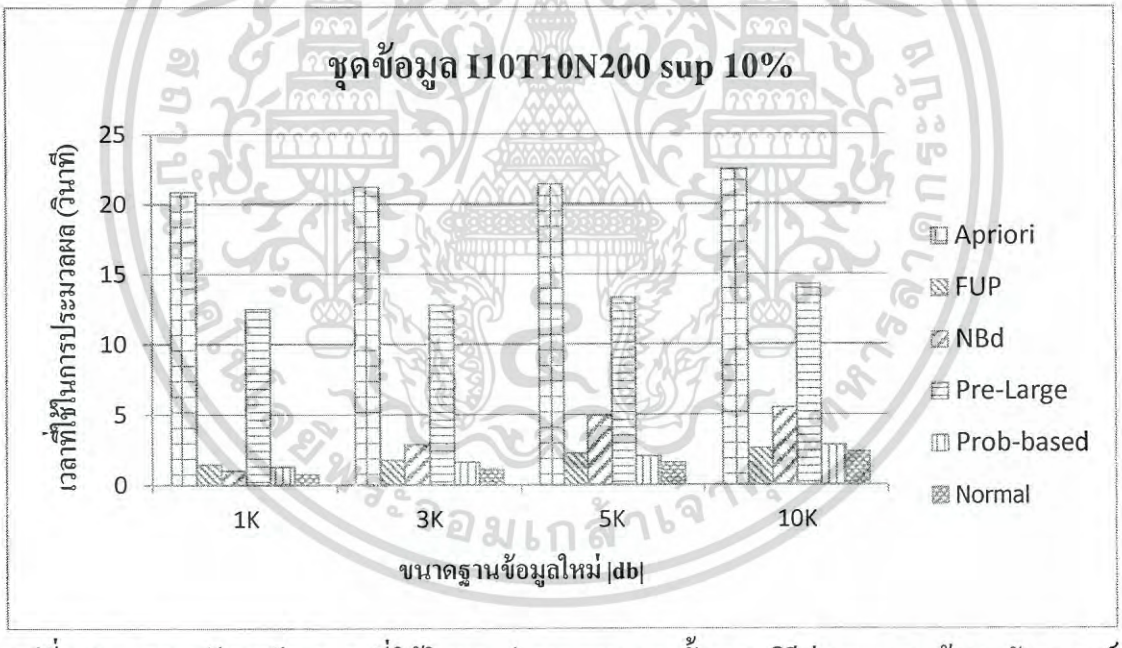
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T10N200							
min_sup	|*db*|	Algorithm					
		Apriori	**FUP**	**NBd**	**Pre-Large**	**Prob-based**	**Normal**
7%	1%	34.23	2.12	4.39	25.79	2.02	1.16
	3%	34.89	2.79	4.84	26.22	2.39	1.77
	5%	35.74	4.32	6.69	27.25	3.59	2.49
	10%	37.44	7.86	11.18	28.69	4.89	3.95



รูปที่ 4.7 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 7%

ตารางที่ 4.11 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 10%

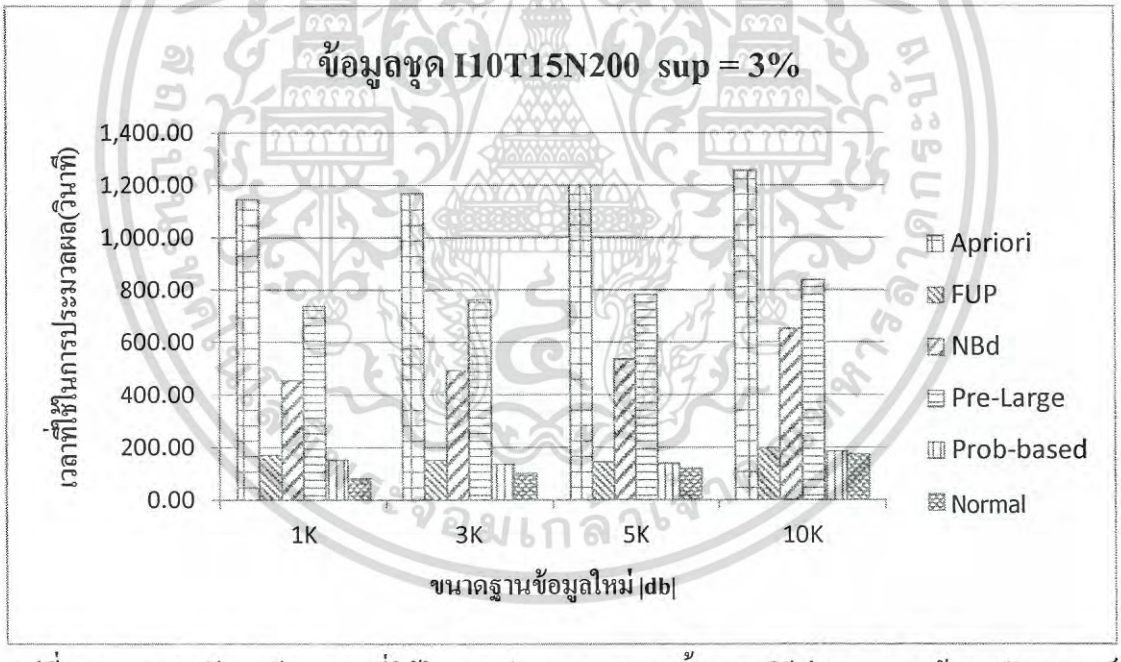
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T10N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
10%	1%	20.88	1.46	1.03	12.52	1.30	0.79
	3%	21.25	1.79	2.92	12.82	1.64	1.14
	5%	21.46	2.24	4.91	13.36	2.07	1.60
	10%	22.51	2.63	5.51	14.30	2.85	2.39



รูปที่ 4.8 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 10%

ตารางที่ 4.12 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 3%

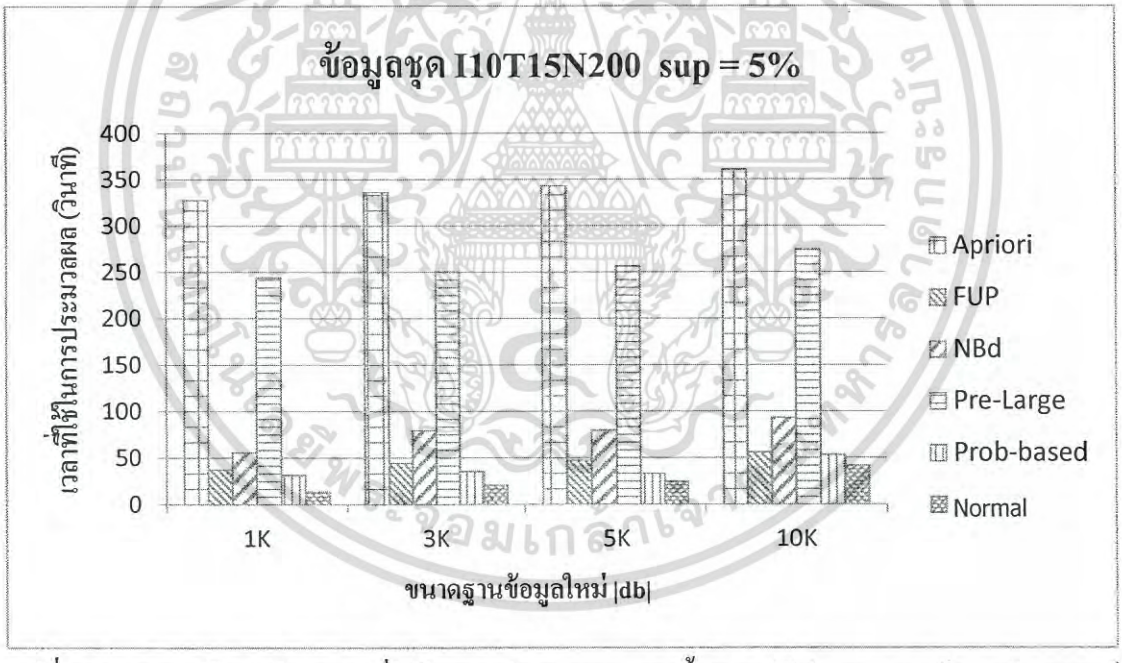
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T15N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
3%	1%	1,146.65	171.76	456.61	740.55	152.91	84.14
	3%	1,169.18	151.88	495.03	763.95	136.70	100.71
	5%	1,200.50	145.39	536.68	783.21	139.88	119.80
	10%	1,257.08	198.15	652.93	839.82	186.87	175.42



รูปที่ 4.9 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 3%

ตารางที่ 4.13 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 5%

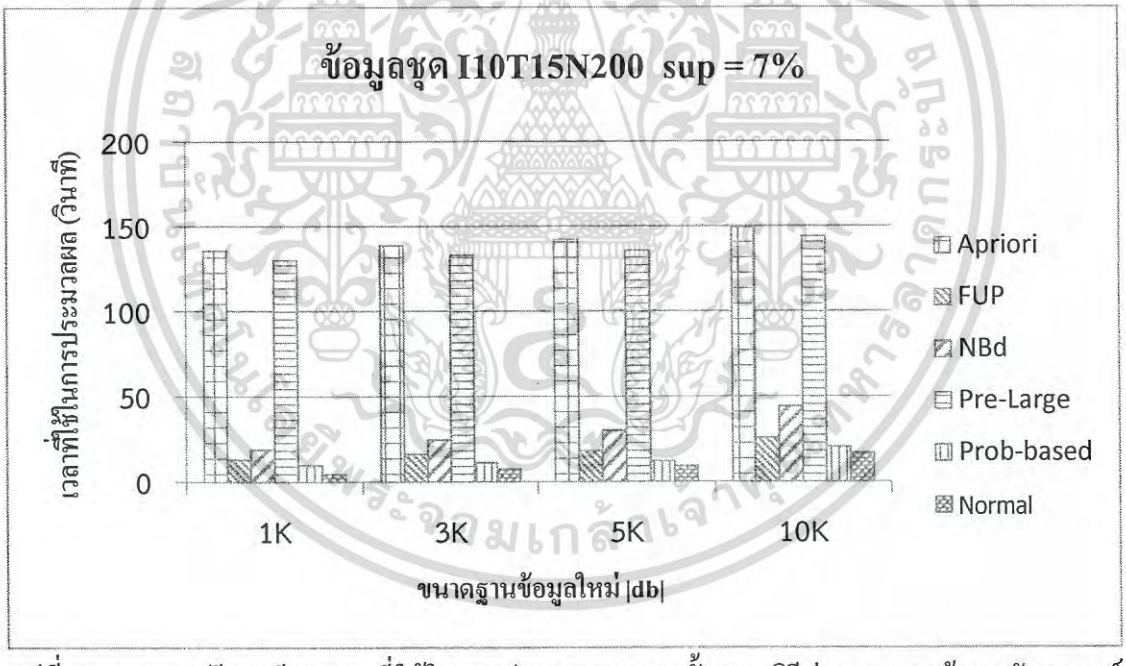
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T15N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
5%	1%	327.85	38.03	55.97	244.12	31.78	14.10
	3%	336.61	44.81	79.62	250.79	35.85	20.78
	5%	343.18	47.64	80.03	256.56	33.43	24.76
	10%	360.77	56.14	93.58	274.51	53.37	42.00



รูปที่ 4.10 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 5%

ตารางที่ 4.14 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 7%

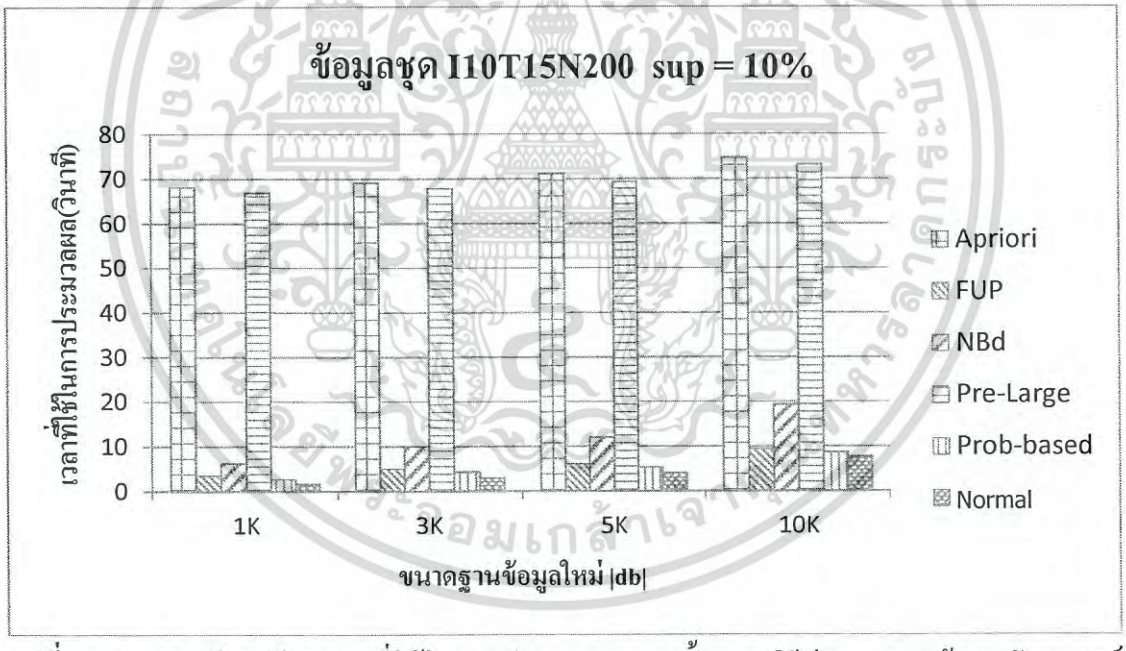
เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T15N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
7%	1%	135.73	13.13	18.61	130.17	9.59	4.57
	3%	138.84	16.22	24.55	132.99	11.14	7.26
	5%	142.51	18.11	29.99	135.77	12.22	9.39
	10%	149.57	25.70	44.33	144.17	20.37	17.03



รูปที่ 4.11 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 7%

ตารางที่ 4.15 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 10%

เวลาที่ใช้ในการประมวลผล (วินาที) ของชุดข้อมูล I10T15N200							
min_sup	|db|	Algorithm					
		Apriori	FUP	NBd	Pre-Large	Prob-based	Normal
10%	1%	68.21	3.61	6.36	66.95	2.73	1.74
	3%	69.21	5.13	10.05	67.88	4.37	3.13
	5%	71.25	6.19	12.10	69.39	5.42	4.25
	10%	74.76	9.52	19.39	73.24	8.69	7.75



รูปที่ 4.12 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง ด้วยค่าสนับสนุน 10%

ตารางที่ 4.16 จำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย และจำนวนไอเท็มเซตที่นำกลับไปสแกนฐานข้อมูลเดิมในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง

|db|	sup	จำนวนไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บ				จำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนฐานข้อมูลเดิม		
		NBd	Pre-Large	Prob-based	Normal	Pre-Large	Prob-based	Normal
1K	3%	4,361	579	182	201	4,628	54	-
	5%	1,919	239	79	92	2,144	22	-
	7%	1,170	115	31	48	1,153	-	-
	10%	628	60	11	22	611	-	-
3K	3%	4,361	579	193	201	4,647	55	-
	5%	1,919	239	79	92	2,142	20	-
	7%	1,170	115	33	48	1,153	1	-
	10%	628	60	9	22	611	-	-
5K	3%	4,361	579	142	201	4,623	50	-
	5%	1,919	239	79	92	2,143	25	-
	7%	1,170	115	34	48	1,122	3	-
	10%	628	60	10	22	611	-	-
10K	3%	4,361	579	203	201	4,614	57	-
	5%	1,919	239	47	92	2,142	27	-
	7%	1,170	115	35	48	1,142	5	-
	10%	628	60	8	22	611	1	-

ตารางที่ 4.17 จำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย และจำนวนไอเท็มเซตที่นำกลับไปสแกนฐานข้อมูลเดิมในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง

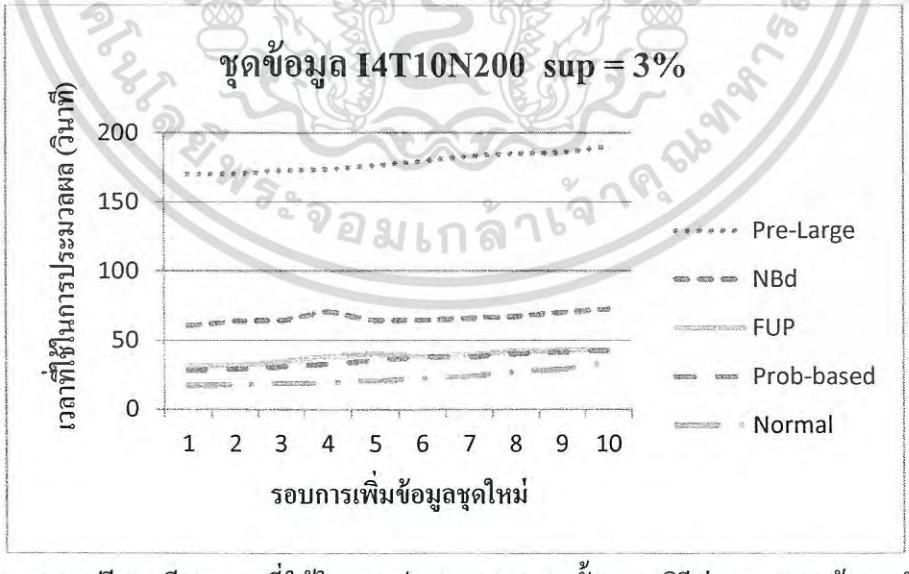
|db|	sup	ไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บ				จำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนฐานข้อมูลเดิม		
		NBd	Pre-Large	Prob-based	Normal	Pre-Large	Prob-based	Normal
1K	3%	4,061	294	52	112	4,315	137	-
	5%	2,034	78	19	43	2,107	27	-
	7%	1,276	40	13	36	1,381	-	-
	10%	701	26	4	8	705	-	-
3K	3%	4,061	294	139	112	4,416	118	-
	5%	2,034	78	19	43	2,110	20	-
	7%	1,276	40	13	36	1,380	-	-
	10%	701	26	4	8	706	-	-
5K	3%	4,061	294	142	112	4,423	72	-
	5%	2,034	78	19	43	2,111	22	-
	7%	1,276	40	15	36	1,386	1	-
	10%	701	26	4	8	703	-	-
10K	3%	4,061	294	143	112	4,410	60	-
	5%	2,034	78	19	43	2,110	17	-
	7%	1,276	40	13	36	1,380	-	-
	10%	701	26	4	8	705	-	-

ตารางที่ 4.18 จำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย และจำนวนไอเท็มเซตที่นำกลับไปสแกนฐานข้อมูลเดิมในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 1 ครั้ง

|db|	sup	ไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บ				จำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนฐานข้อมูลเดิม		
		NBd	Pre-Large	Prob-based	Normal	Pre-Large	Prob-based	Normal
1K	3%	18,135	857	336	504	19,154	33	-
	5%	7,302	233	131	225	7,625	12	-
	7%	3,655	190	45	93	4,361	5	-
	10%	2,312	42	8	18	2,507	-	-
3K	3%	18,135	857	336	504	19,680	35	-
	5%	7,302	233	130	225	7,637	15	-
	7%	3,655	190	44	93	4,343	7	-
	10%	2,312	42	8	18	2,509	-	-
5K	3%	18,135	857	336	504	19,682	43	-
	5%	7,302	233	130	225	7,632	22	-
	7%	3,655	190	44	93	4,211	14	-
	10%	2,312	42	8	18	2,498	2	-
10K	3%	18,135	857	182	504	19,150	97	-
	5%	7,302	233	151	225	7,629	7	-
	7%	3,655	190	43	93	4,292	-	-
	10%	2,312	42	8	18	2,479	2	-

ตารางที่ 4.19 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่เท่ากันที่ค่าสนับสนุน 3%

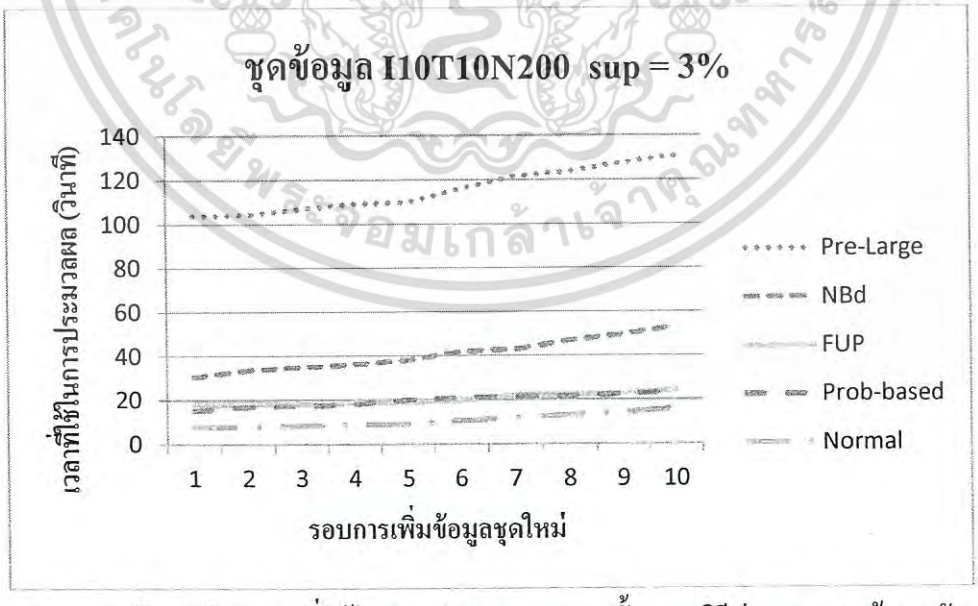
จำนวน|db| ที่เพิ่มเข้าไปในแต่ละรอบ	เวลาที่ใช้ในการประมวลผล (วินาที)				
	Pre-Large	**NBd**	**FUP**	**Prob-based**	**Normal**
1000	169.9	60.53	31.67	28.07	17.34
1000	170.11	63.89	32.01	29.14	18.01
1000	172.45	64.11	34.65	30.99	18.87
1000	173.44	70.82	38.42	32.87	19.45
1000	176.21	64.11	40.22	35.55	20.96
1000	179.38	64.45	38.56	37.88	22.14
1000	182.96	66.12	39.63	38.04	24.11
1000	185.13	67.23	42.45	40.07	26.98
1000	185.56	70.11	42.98	41.47	29.13
1000	190.14	72.56	43.39	42.44	33.74
เวลารวม	**1,785.28**	**663.93**	**383.98**	**356.52**	**230.73**



รูปที่ 4.13 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่เท่ากันที่ค่าสนับสนุน 3%

ตารางที่ 4.20 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่เท่ากันที่ค่าสนับสนุน 3%

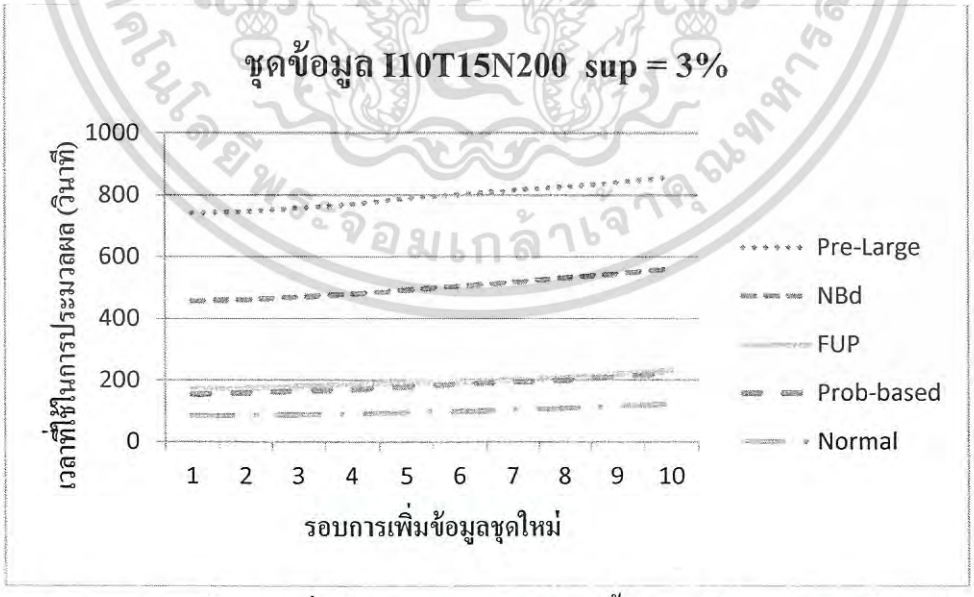
จำนวน|db| ที่เพิ่มเข้าไปในแต่ละรอบ	เวลาที่ใช้ในการประมวลผล (วินาที)				
	Pre-Large	**NBd**	**FUP**	**Prob-based**	**Normal**
1000	103.67	30.61	17.97	15.16	7.49
1000	104.14	33.66	18.04	16.97	7.88
1000	106.69	34.97	18.35	17.11	8.32
1000	109.12	36.25	18.89	18.12	8.69
1000	109.54	38.22	19.11	20.02	9.01
1000	115.69	41.78	19.87	20.84	10.5
1000	121.47	43.02	22.03	21.23	11.87
1000	123.65	46.98	22.17	21.59	13.12
1000	127.41	49.65	22.95	22.48	14.15
1000	130.74	53.23	24.41	23.56	15.98
เวลารวม	**1,152.12**	**408.37**	**203.79**	**197.08**	**107.01**



รูปที่ 4.14 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่เท่ากันที่ค่าสนับสนุน 3%

ตารางที่ 4.21 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่เท่ากันที่ค่าสนับสนุน 3%

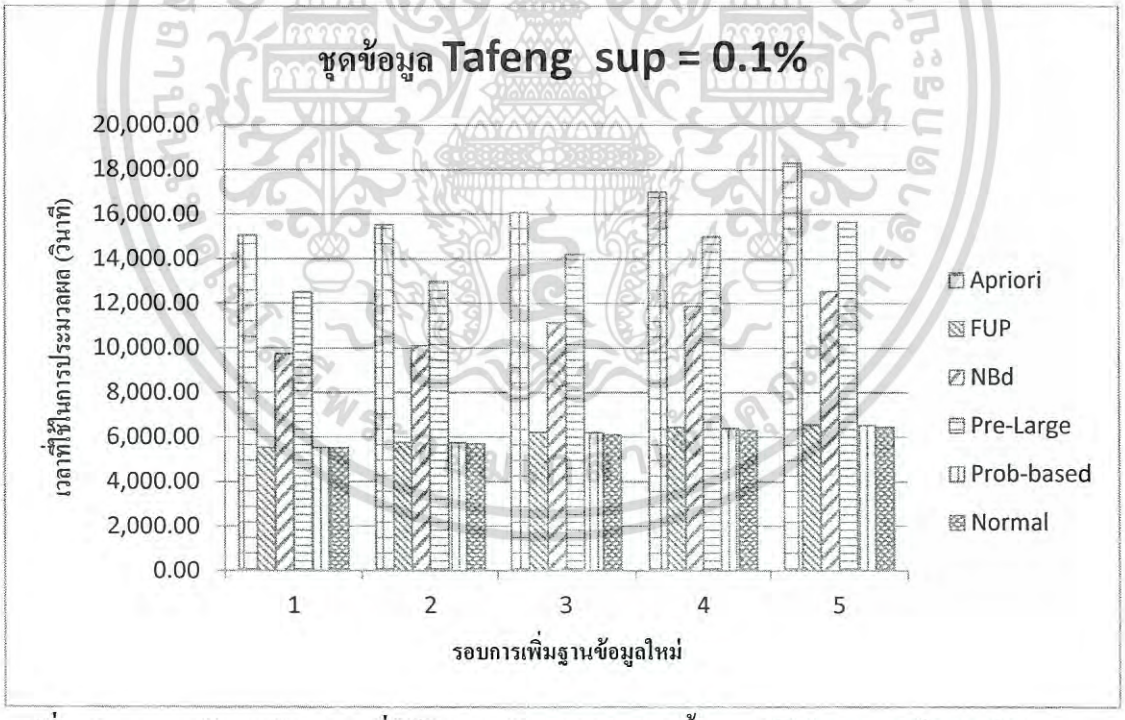
จำนวน|db| ที่เพิ่มเข้าไปในแต่ละรอบ	เวลาที่ใช้ในการประมวลผล (วินาที)				
	Pre-Large	**NBd**	**FUP**	**Prob-based**	**Normal**
1000	740.55	456.61	171.76	152.91	84.14
1000	746.69	460.67	174.44	157.74	85.67
1000	757.15	470.13	179.31	164.98	87.23
1000	768.94	479.58	186.98	169.32	90.11
1000	789.16	492.55	192.57	177.95	94.86
1000	801.36	505.13	195.66	187.21	97.44
1000	815.45	517.41	201.47	193.54	103.41
1000	827.26	532.91	207.63	199.6	108.55
1000	840.66	544.11	220.28	210.81	115.78
1000	858.67	561.74	231.62	220.13	122.97
เวลารวม	**7,945.89**	**5,020.84**	**1,961.72**	**1,834.19**	**990.16**



รูปที่ 4.15 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่เท่ากันที่ค่าสนับสนุน 3%

ตารางที่ 4.22 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลจริงชุด Tafeng เมื่อมีการเพิ่มข้อมูลเข้าไป 5 ครั้ง ด้วยขนาดฐานข้อมูล 5000 รายการธุรกรรมที่ค่าสนับสนุน 0.1%

|db|	เวลาที่ใช้ในการประมวลผล (วินาที)					
	Apriori	**FUP**	**NBd**	**Pre-Large**	**Prob-based**	**Normal**
5000	15,074.42	5,543.11	9,741.36	12,508.58	5,547.13	5,528.51
5000	15,535.89	5,779.63	10,112.63	13,011.12	5,763.87	5,712.48
5000	16,087.24	6,211.45	11,139.14	14,215.77	6,198.12	6,123.56
5000	17,012.65	6,436.89	11,890.66	14,993.85	6,401.72	6,314.46
5000	18,291.74	6,559.82	12,554.51	15,669.36	6,521.17	6,454.88
เวลารวม	**80,001.94**	**30,530.90**	**55,438.30**	**70,398.68**	**30,432.01**	**30133.89**



รูปที่ 4.16 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆของข้อมูลจริงชุด Tafeng เมื่อมีการเพิ่มข้อมูลเข้าไป 5 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่เท่ากันที่ค่าสนับสนุน 0.1%

ตารางที่ 4.23 จำนวนไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลชุด Tafeng เมื่อมีการเพิ่มข้อมูลเข้าไป 5 ครั้ง ครั้งละ 5000 รายการธุรกรรม

|db|	ไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บ				จำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนฐานข้อมูลเดิม		
	NBd	Pre-Large	Prob-based	Normal	Pre-Large	Prob-based	Normal
5000	502,886	1,086	225	413	125,077	62	23
5000	502,714	1,117	241	425	125,223	67	19
5000	502,755	1,110	230	414	125,124	61	18
5000	502,732	1,132	237	426	125,302	73	24

4.3.1 ผลการทดลองที่ 1: การทดลองการเพิ่มขยายกฎความสัมพันธ์ในกรณีที่ข้อมูลชุดใหม่ที่เพิ่มเข้าไปมีขนาดเท่ากัน

การทดลองที่ 1 ต้องการทดสอบความถูกต้องและวัดประสิทธิภาพในการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติในกรณีที่ชุดข้อมูลที่เพิ่มเข้าไปมีขนาดเท่ากัน

ในการทดสอบความถูกต้องของการค้นหาไอเท็มเซตที่เกิดบ่อย ผู้วิจัยได้ทดสอบกับไอเท็มเซตที่เกิดบ่อยที่แต่ละขั้นตอนวิธีค้นหาพบเปรียบเทียบกับไอเท็มเซตที่เกิดบ่อยที่ขั้นตอนวิธีอะพริโอริ ค้นพบ โดยเปรียบเทียบกับขั้นตอนวิธีทั้ง 5 ได้แก่ ขั้นตอนวิธี เอฟยูพี ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ ขั้นตอนวิธี พรีลาร์จ ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นและขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการแจกแจงปกติ

ผลการค้นหาไอเท็มเซตที่เกิดบ่อยหลังจากที่ได้ปรับปรุงฐานข้อมูลใหม่เข้าไปแล้ว โดยไอเท็มเซตที่เกิดบ่อยที่ได้ในแต่ละชุดข้อมูลที่ทดสอบด้วยค่าสนับสนุนที่ 3% 5% 7% และ 10% ขนาดของฐานข้อมูลใหม่จำนวน 1000, 3000, 5000 และ 10000 มีจำนวนเท่ากับจำนวนไอเท็มเซตที่เกิดบ่อยที่ได้ด้วยการประมวลผลของขั้นตอนวิธีอะพริโอริ แสดงตามตารางที่ 4.1 ตารางที่ 4.2 และตารางที่ 4.3 นั่นหมายความว่า ขั้นตอนวิธีที่ทำการเปรียบเทียบทุกขั้นตอนวิธี ให้ผลลัพธ์การค้นหาไอเท็มเซตที่เกิดบ่อยได้อย่างถูกต้อง

การทดสอบประสิทธิภาพการทำงานของขั้นตอนวิธีด้านเวลาที่ใช้ในการประมวลผล โดยทดสอบกับข้อมูลสังเคราะห์ 3 ชุด ได้แก่ ชุด A I4T10N200 ชุด B I10T10N200 และ ชุด C

I10T15N200 ผู้วิจัยได้ทดสอบความเร็วในการทำงานของขั้นตอนวิธีโดยเปรียบเทียบความเร็วในการทำงานที่ค่าสนับสนุนขั้นต่ำ 3% 5% 7% และ 10 % ด้วยขนาดชุดข้อมูลใหม่ที่เพิ่มไปด้วยขนาด 1% 3% 5% และ 10% หรือจำนวนรายการธุรกรรม 1000, 3000, 5000 และ 10000 รายการธุรกรรม ผลการทดสอบด้านเวลาแสดงตามตารางที่ 4.4 ถึงตารางที่ 4.15 และรูปที่ 4.1 ถึงรูปที่ 4.12 และผลการเก็บไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดขึ้นบ่อยและไอเท็มเซตที่จะถูกนำไปสแกนในฐานข้อมูลเดิมในรอบสุดท้าย แสดงดังตารางที่ 4.16 ถึงตารางที่ 4.18

การทดสอบประสิทธิภาพของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติโดยการเพิ่มข้อมูลเข้าไปอย่างต่อเนื่องด้วยขนาดฐานข้อมูลใหม่ที่มีขนาดเท่ากัน ผู้วิจัยได้ทดสอบกับข้อมูลสังเคราะห์ทั้ง 3 ชุด โดยกำหนดให้ฐานข้อมูลเดิมมีขนาด 100,000 รายการธุรกรรม และเพิ่มข้อมูลใหม่เข้าไปอย่างต่อเนื่องจำนวน 10 ครั้ง ครั้งละ 1,000 รายการธุรกรรม ด้วยค่าสนับสนุนขั้นต่ำ 3% ผลการทดสอบประสิทธิภาพด้านเวลาแสดงตามตารางที่ 4.19 ถึงตารางที่ 4.21 และรูปที่ 4.13 ถึงรูปที่ 4.15

นอกจากนี้ ผู้วิจัยได้ทดลองกับข้อมูลจริงชุดข้อมูล Tafeng โดยกำหนดให้ฐานข้อมูลเดิมมีขนาด 50,000 รายการธุรกรรม และมีการเพิ่มชุดข้อมูลใหม่เข้าไปอย่างต่อเนื่องจำนวน 5 ครั้ง ครั้งละ 5,000 รายการธุรกรรม ผลการทดสอบประสิทธิภาพด้านเวลาแสดงตามตารางที่ 4.22 และรูปที่ 4.16 และผลการเก็บไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดขึ้นบ่อยและไอเท็มเซตที่จะถูกนำไปสแกนในฐานข้อมูลเดิมในรอบสุดท้าย แสดงดังตารางที่ 4.23

ผลการทดลองจะเห็นได้ว่า การทดลองประสิทธิภาพการทำงานของขั้นตอนวิธีกับข้อมูลทั้ง 3 ชุด ที่ระดับค่าสนับสนุนที่แตกต่างกัน และขนาดชุดข้อมูลที่แตกต่าง ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติใช้เวลาในการประมวลผลที่ดีกว่าขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นเนื่องมาจากเหตุผลดังนี้

1. ขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าด้วยการแจกแจงปกติ มีการเก็บไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง เพื่อนำไปช่วยในการประมวลผลในครั้งถัดไปที่มีข้อมูลชุดใหม่เพิ่มเข้ามา ทำให้ระหว่างการประมวลผลแต่รอบ k ในส่วนของการประมวลผลฐานข้อมูลปรับปรุงสามารถประมวลผลเฉพาะฐานข้อมูลใหม่โดยไม่ต้องกลับไปสแกนฐานข้อมูลเดิมในทุกๆ รอบ เนื่องจาก ไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บเพียงพอเพื่อนำมาประมวลผลในรอบการปรับปรุงข้อมูล

ในขณะที่ขั้นตอนวิธีเอฟยูพี จำเป็นต้องมีการสแกนฐานข้อมูลเดิมในทุกๆ รอบ เพื่อปรับปรุงค่าสนับสนุนของไอเท็มเซต ซึ่งทำให้สูญเสียเวลาในการกระทำดังกล่าว

2. เพื่อป้องกันการสูญหายของไอเท็มที่มีโอกาสเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง ซึ่งไม่ได้อยู่ในเซตของไอเท็มที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยที่ได้จัดเก็บไว้ก่อนหน้า ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงความน่าจะเป็น ได้กำหนดตัวแปร *Temp_scanDB* ให้จัดเก็บไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในระหว่างการประมวลผลในรอบ Incremental mining phase เมื่อเสร็จสิ้นกระบวนการทั้งหมดจึงจะนำไอเท็มที่อยู่ในตัวแปร *Temp_scanDB* ไปสแกนฐานข้อมูลเดิมเป็นรอบสุดท้าย เพื่อป้องกันการสูญหายของไอเท็มเซต

ด้วยหลักการเดียวกันนี้ ขั้นตอนวิธีที่เก็บไอเท็มเซตเพื่อนำไปสแกนฐานข้อมูลเดิมรอบสุดท้ายได้แก่ ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ ขั้นตอนวิธี พรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นก็มีการเก็บไอเท็มดังกล่าวเช่นกัน จากการทดลองพบว่า ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ และ พรีลาร์จ มีการเก็บไอเท็มไว้เพื่อสแกนฐานข้อมูลเดิมรอบสุดท้ายจำนวนมาก แสดงดังตารางที่ 4.16 ถึงตารางที่ 4.18 ทำให้สูญเสียเวลาในการกระทำดังกล่าว ในขณะที่ขั้นตอนวิธีที่ผู้วิจัยเสนอและขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นมีการเก็บ *Temp_scanDB* น้อยมากจนในบางรอบการทดลองไม่มีจำนวน *Temp_scanDB* ที่จะนำไปสแกนฐานข้อมูลเดิมเลย จึงทำให้ลดเวลาในการประมวลผลส่วนนี้ไป

3. เมื่อเปรียบเทียบกับเฉพาะขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น จะเห็นได้ว่า ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ มีเวลาในการประมวลผลที่เร็วกว่า ทั้งนี้เนื่องจาก ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจำเป็นต้องมีการคำนวณค่าความน่าจะเป็นของไอเท็มที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยทุกๆ ตัว และในทุกๆ รอบ ในขณะขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ มีการคำนวณเพียงครั้งเดียว ทำให้มีเวลาในการประมวลผลที่ดีกว่า

นอกจากนี้เมื่อพิจารณาตารางที่ 4.16 – 4.18 และตารางที่ 23 ซึ่งแสดงผลการเก็บไอเท็มเซตที่ไม่ได้เป็นไอเท็มเซตที่เกิดขึ้นบ่อยและไอเท็มเซตที่จะถูกนำไปสแกนในฐานข้อมูลเดิมในรอบสุดท้าย จะเห็นได้ว่า แม้ว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงแบบปกติจะมีการเก็บไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงมากกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นซึ่งขั้นตอนวิธีที่นำเสนอมีการเก็บไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงได้ครอบคลุมกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นส่งผลให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติมีการนำไอเท็มเซตกลับไปสแกนในฐานข้อมูลเดิมน้อยมากจนการทดลองบางการทดลองแทบไม่มีการนำไอเท็ม

เซตกลับสแกนในฐานข้อมูลเดิมเลย จึงทำให้อัลกอริทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ประมวลผลได้รวดเร็วกว่า

อย่างไรก็ดี เมื่อพิจารณาผลการทดลองด้วยชุดข้อมูลจริง Tafeng จะเห็นได้ว่า ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ขั้นตอนวิธีเอฟยูพี และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นมีเวลาที่แตกต่างกันไม่มากนัก สาเหตุเนื่องมาจากไอเท็มเซตที่เกิดบ่อยที่ค้นหาได้ มีขนาดสูงสุดคือ 3 จึงทำให้ขั้นตอนวิธีเอฟยูพีมีการนำไอเท็มเซตกลับไปสแกนในฐานข้อมูลเดิมเพียง 3 รอบ ซึ่งแตกต่างจากการทดลองในชุดข้อมูลสังเคราะห์ที่ขั้นตอนเอฟยูพี ต้องนำไอเท็มเซตกลับไปสแกนฐานข้อมูลเดิมจำนวนหลายรอบ

นอกจากนี้ สาเหตุที่ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ มีเวลาในการประมวลผลที่เร็วกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นไม่มากนัก เนื่องจากไอเท็มเซตที่ถูกนำกลับไปสแกนในฐานข้อมูลเดิมมีจำนวนแตกต่างกันไม่มาก ทำให้เวลาที่ใช้ในการประมวลผลมีความใกล้เคียงกัน

4.3.2 ผลการทดลองที่ 2: การทดลองการเพิ่มขยายกฎความสัมพันธ์เมื่อข้อมูลชุดใหม่ที่เพิ่มเข้าไปมีขนาดแตกต่างกัน

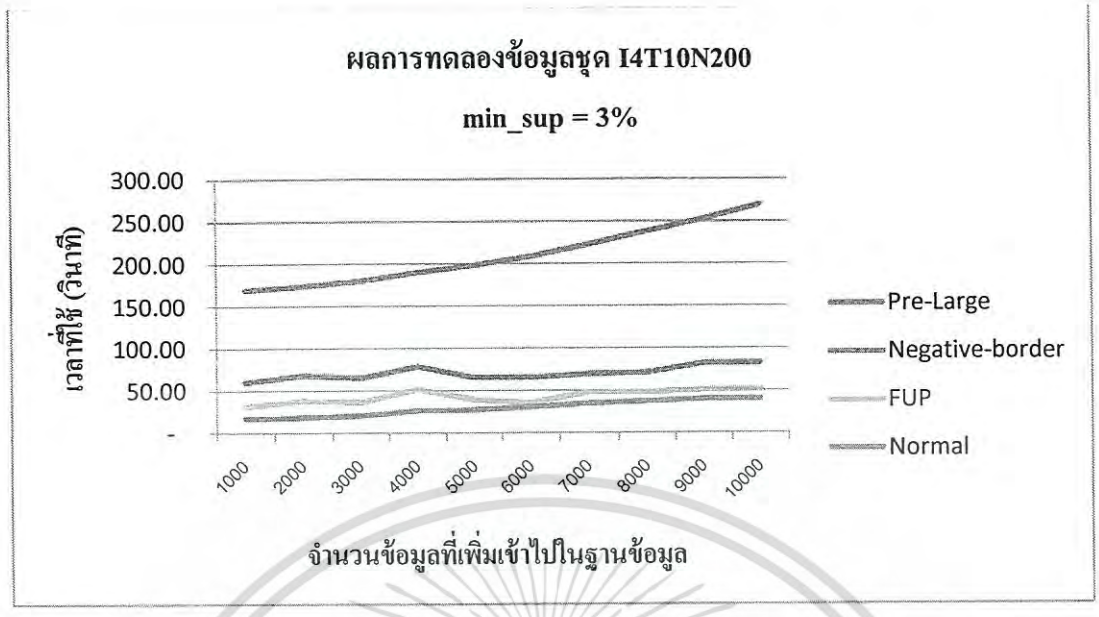
เนื่องจากขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติถูกออกแบบการทำงานมาเพื่อทำค้นหาไอเท็มเซตที่เกิดบ่อยของฐานข้อมูลปรับปรุงโดยสามารถนำเข้าชุดข้อมูลได้หลายขนาดและไม่จำเป็นต้องทราบขนาดของฐานข้อมูลใหม่ล่วงหน้า ดังนั้น การทดลองที่ 2 เป็นการทดลองเพื่อทดสอบความถูกต้องและประสิทธิภาพในการค้นหาไอเท็มเซตที่เกิดบ่อยของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ เมื่อมีการนำเข้าข้อมูลชุดใหม่ต่อเนื่องกันหลายๆ ขนาด โดยในการทดลองมีการทดสอบนำเข้าข้อมูลจำนวน 10 ชุด ที่มีขนาดแตกต่างกัน ผลการทดลองปรากฏดังตารางที่ 4.24 ตารางที่ 4.26 และตารางที่ 4.28 และรูปที่ 4.17 – 4.19 นอกจากนี้ผลการเก็บจำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย และจำนวนไอเท็มเซตที่นำกลับไปสแกนฐานข้อมูลเดิมในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลแต่ละชุด แสดงดังตารางที่ 4.25 ตารางที่ 4.27 และตารางที่ 4.29

การทดลองเพิ่มข้อมูลเข้าไปด้วยขนาดต่างกันจำนวน 10 ชุด อย่างต่อเนื่อง ถูกทดลองด้วยชุดข้อมูล I4T10N200 I10T10N200 และ I10T15N200 ที่ค่าสนับสนุน 3% ผลการทดลองพบว่า ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติให้ผลลัพธ์การค้นหาไอเท็มเซตที่เกิดบ่อยที่ถูกต้องเมื่อเทียบกับขั้นตอนวิธี อะพริโอริ และใช้เวลา

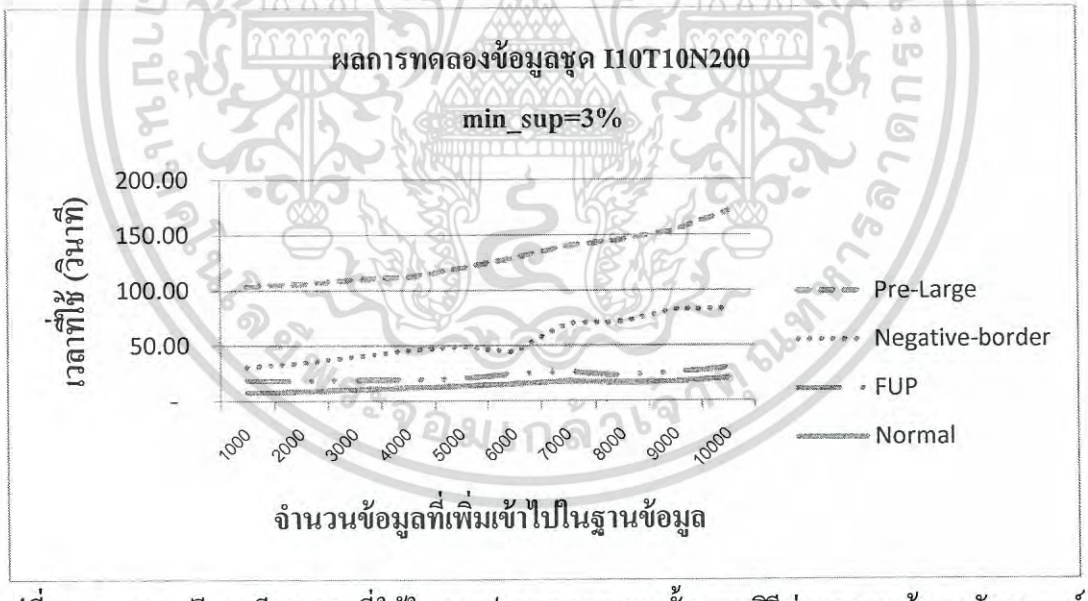
ในการประมวลผลรวมกันทุกรอบที่เร็วกว่าขั้นตอนวิธี เนกาทีฟบอร์เดอร์ ขั้นตอนวิธี พรีลาร์จ และขั้นตอนวิธี เอฟยูพี เนื่องจากไม่ต้องนำไอเท็มเซตกลับไปสแกนในฐานข้อมูลเดิมทุกครั้ง และจำนวนไอเท็มใน *Temp_scanDB* ที่จะถูกนำกลับไปสแกนในฐานข้อมูลเดิม มีน้อยกว่าเมื่อเทียบกับขั้นตอนวิธี และขั้นตอนวิธี พรีลาร์จ ซึ่งแสดงดังตารางที่ 4.25 ตารางที่ 4.27 และตารางที่ 4.29

ตารางที่ 4.24 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไปจำนวน 10 ครั้งด้วยขนาดฐานข้อมูลใหม่ที่แตกต่างกัน ที่ค่าสนับสนุน 3%

จำนวน|db| ที่เพิ่มเข้าไปในแต่ละรอบ	เวลาที่ใช้ในการประมวลผล (วินาที)			
	Pre-Large	**Negative-border**	**FUP**	**Normal**
1000	169.90	60.53	31.67	17.34
2000	174.55	68.81	38.52	18.90
3000	180.34	65.45	36.53	20.75
4000	190.22	79.28	51.78	26.83
5000	198.70	65.77	39.70	27.55
6000	208.93	66.04	35.80	31.00
7000	222.82	70.12	48.51	35.14
8000	238.06	71.55	48.10	37.61
9000	252.72	82.69	51.01	40.20
10000	269.45	83.01	51.45	40.29
เวลารวม	**2,105.69**	**713.25**	**433.07**	**295.61**



รูปที่ 4.17 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไปจำนวน 10 ครั้งด้วยขนาดฐานข้อมูลใหม่ที่แตกต่างกันที่ค่าสนับสนุน 3%



รูปที่ 4.18 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไปจำนวน 10 ครั้งด้วยขนาดฐานข้อมูลใหม่ที่แตกต่างกันที่ค่าสนับสนุน 3%

ตารางที่ 4.25 จำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย และจำนวนไอเท็มเซตที่นำกลับไปสแกนฐานข้อมูลเดิมในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I4T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่ต่างกัน ที่ค่าสนับสนุน 3%

|db|	จำนวนไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บ			จำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนฐานข้อมูลเดิม	
	NBd	Pre-Large	Normal	Pre-Large	Normal
1000	4,361	579	201	4,628	-
2000	4,361	579	197	4,635	-
3000	4,363	584	197	4,634	-
4000	4,365	584	198	4,635	-
5000	4,365	582	197	4,635	-
6000	4,365	584	197	4,639	-
7000	4,363	584	197	4,635	-
8000	4,365	582	198	4,637	-
9000	4,365	585	197	4,635	-
10000	4,363	584	198	4,635	-

ตารางที่ 4.26 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไปจำนวน 10 ครั้งด้วยขนาดฐานข้อมูลใหม่ที่แตกต่างกัน ที่ค่าสนับสนุน 3%

จำนวน|db| ที่เพิ่มเข้าไปในแต่ละรอบ	เวลาที่ใช้ในการประมวลผล (วินาที)			
	Pre-Large	**Negative-border**	**FUP**	**Normal**
1000	103.67	30.61	17.97	7.49
2000	105.78	34.14	18.24	9.14
3000	109.95	39.71	18.86	10.55
4000	112.34	45.60	19.17	12.14
5000	120.01	48.99	20.15	13.65
6000	129.10	44.14	24.60	15.46
7000	140.25	70.12	26.66	18.20
8000	145.77	71.55	23.04	16.59
9000	153.97	82.69	25.78	17.63
10000	170.90	83.01	30.04	20.34
เวลารวม	**1,291.74**	**550.56**	**224.51**	**141.19**

ตารางที่ 4.27 จำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย และจำนวนไอเท็มเซตที่นำกลับไปสแกนฐานข้อมูลเดิมในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T10N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่ต่างกัน ที่ค่าสนับสนุน 3%

|db|	จำนวนไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บ			จำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนฐานข้อมูลเดิม	
	NBd	**Pre-Large**	**Normal**	**Pre-Large**	**Normal**
1000	4,061	294	112	4,315	-
2000	4,061	294	112	4,320	-
3000	4,061	315	111	4,322	-
4000	4,067	310	113	4,318	-
5000	4,067	305	112	4,321	-
6000	4,065	305	111	4,320	-
7000	4,065	308	113	4,319	-
8000	4,072	305	114	4,316	-
9000	4,069	307	112	4,320	-
10000	4,068	307	112	4,321	-

ตารางที่ 4.28 เวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไปจำนวน 10 ครั้งด้วยขนาดฐานข้อมูลใหม่ที่แตกต่างกัน ที่ค่าสนับสนุน 3%

จำนวน|db| ที่เพิ่มเข้าไปในแต่ละรอบ	เวลาที่ใช้ในการประมวลผล (วินาที)			
	Pre-Large	**Negative-border**	**FUP**	**Normal**
1000	740.55	456.61	171.76	84.14
2000	743.77	458.66	172.84	87.66
3000	746.26	459.14	174.95	89.01
4000	748.99	461.35	175.87	93.14
5000	753.67	463.69	177.16	96.81
6000	757.48	466.56	178.29	100.11
7000	762.26	470.14	180.73	104.64
8000	767.13	471.15	182.35	111.32
9000	772.31	475.69	184.55	118.97
10000	782.59	480.76	187.46	121.36
เวลารวม	**7,575.01**	**4,663.75**	**1,785.96**	**1,007.16**

รูปที่ 4.19 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไปจำนวน 10 ครั้งด้วยขนาดฐานข้อมูลใหม่ที่แตกต่างกันที่ค่าสนับสนุน 3%

ตารางที่ 4.29 จำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย และจำนวนไอเท็มเซตที่นำกลับไปสแกนฐานข้อมูลเดิมในการประมวลผลของขั้นตอนวิธีต่างๆ ของข้อมูลสังเคราะห์ชุด I10T15N200 เมื่อมีการเพิ่มข้อมูลเข้าไป 10 ครั้ง ด้วยขนาดฐานข้อมูลใหม่ที่ต่างกัน ที่ค่าสนับสนุน 3%

|db|	จำนวนไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่ถูกจัดเก็บ			จำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนฐานข้อมูลเดิม	
	NBd	Pre-Large	Normal	Pre-Large	Normal
1000	18,135	857	504	19,154	-
2000	18,141	860	504	19,156	-
3000	18,150	858	506	19,160	-
4000	18,152	858	505	19,160	-
5000	18,145	858	506	19,162	-
6000	18,145	861	504	19,165	-
7000	18,149	860	505	19,160	-
8000	18,152	859	505	19,162	-
9000	18,151	860	505	19,165	-
10000	18,150	858	506	19,163	-

4.4 สรุปผลการทดลอง

จากทดสอบวัดความถูกต้องในการค้นหาไอเท็มเซตที่เกิดบ่อยและวัดประสิทธิภาพการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติ ซึ่งมีการทดสอบโดยใช้ชุดข้อมูลสังเคราะห์จำนวน 3 ชุด คือข้อมูลชุด A (I4T10N200) ข้อมูลชุด B (I10T10N200) และข้อมูลชุด C (I10T15N200) ทดสอบกับค่าสนับสนุน 4 ค่า คือ 3% 5% 7% และ 10% และทดสอบกับขนาดของชุดข้อมูลใหม่จำนวน 4 ขนาดได้แก่ 1000, 3000, 5000 และ 10000 รายการธุรกรรม ผลการทดลองพบว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติให้ผลลัพธ์ที่ถูกต้องคือสามารถค้นหาไอเท็มเซตที่เกิดบ่อยได้ตรงตามที่ขั้นตอนวิธีอะพริโอริ ค้นพบ สำหรับประสิทธิภาพทางด้านเวลาในการประมวลผลเมื่อเทียบกับขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยาย

กฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นผลการทดลองพบว่า ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติให้เวลาในการประมวลผลที่รวดเร็วกว่า

เมื่อทดสอบการเพิ่มข้อมูลชุดใหม่เข้าไปในชุดข้อมูลสังเคราะห์ทั้ง 3 ชุด และข้อมูลจริง Tafeng อย่างต่อเนื่องด้วยขนาดข้อมูลใหม่ที่มีขนาดเท่ากัน ผลการทดลองพบว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติให้เวลาในการประมวลผลที่รวดเร็วกว่า แต่ด้วยไอเท็มเซตที่เกิดบ่อยมีขนาดเท่ากับ 3 และจำนวนไอเท็มเซตที่ถูกนำกลับไปสแกนในฐานข้อมูลเดิมมีความแตกต่างไม่มากนักเมื่อเทียบกับขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น จึงทำให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติมีให้เวลาในการประมวลที่เร็วกว่าขั้นตอนวิธีเอฟยูพี และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นไม่มากนัก

นอกจากนี้ ได้มีการทดลองเพิ่มข้อมูลเข้าไปอย่างต่อเนื่องด้วยขนาดของข้อมูลชุดใหม่ที่มีขนาดต่างกันจำนวน 10 ขนาด เพื่อทดสอบประสิทธิภาพการทำงานของขั้นตอนวิธีที่นำเสนอ ผลการทดลองพบว่า ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบแจกแจงปกติสามารถค้นหาไอเท็มเซตที่เกิดบ่อยได้อย่างถูกต้องในทุกชุดข้อมูลทดลอง และมีเวลาในการประมวลผลที่เร็วกว่าขั้นตอนวิธีที่ทำการเปรียบเทียบทั้ง 4 ขั้นตอนวิธี เนื่องจากไม่ต้องนำไอเท็มเซตกลับไปสแกนในฐานข้อมูลเดิมทุกครั้ง และจำนวนไอเท็มใน *Temp_scanDB* ที่จะถูกนำกลับไปสแกนในฐานข้อมูลเดิม มีน้อยกว่าเมื่อเทียบกับขั้นตอนวิธีเนกาทีฟบอร์เดอร์ และขั้นตอนวิธีพรีลาร์จ และไม่ต้องมีการคำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัวในทุกรอบและมีไอเท็มเซตที่ถูกนำไปสแกนในฐานข้อมูลเดิมน้อยมาก เมื่อเทียบกับขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นทั้งนี้เนื่องจากขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการแจกแจงปกติมีการเก็บไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงได้ครอบคลุมกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

บทที่ 5
สรุปผลการวิจัย

5.1 สรุปผลการวิจัย

การค้นหากฎความสัมพันธ์ (Association rule discovery) เป็นหนึ่งในเทคนิคที่สำคัญและได้รับความนิยมในกระบวนการทำเหมืองข้อมูล เพื่อค้นหารูปแบบความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูล ผลที่ได้จากการประมวลผลจะเป็นการสกัดรูปแบบข้อมูลที่น่าสนใจให้ออกมาในรูปแบบของกฎความสัมพันธ์ ถ้า....แล้ว.... (IF X THEN Y) ซึ่งกฎความสัมพันธ์ดังกล่าวสามารถใช้ในการวิเคราะห์พฤติกรรม เช่น การทำนายสินค้าที่ผู้บริโภคนิยมซื้อพร้อมกัน ซึ่งผลลัพธ์ของกฎความสัมพันธ์ที่ได้จะสามารถสร้างความได้เปรียบในการแข่งขันแก่องค์กรและช่วยให้ผู้บริหารขององค์กรมีสารสนเทศที่สำคัญที่ช่วยสนับสนุนการตัดสินใจ

หลักการค้นหากฎความสัมพันธ์มี 2 ขั้นตอนหลักได้แก่ 1) การค้นหาไอเท็มเซตที่เกิดบ่อย ซึ่งเป็นไอเท็มที่มีค่าสนับสนุนมากกว่าหรือเท่ากับค่าสนับสนุนขั้นต่ำ (Minimum support) ที่ผู้ใช้กำหนด 2) นำไอเท็มเซตที่เกิดบ่อยดังกล่าวมาสร้างกฎความสัมพันธ์โดยเปรียบเทียบกับค่าความเชื่อมั่นมากกว่าหรือค่าความเชื่อมั่นขั้นต่ำ (Minimum Confidence) ที่ผู้ใช้กำหนด หากกฎความสัมพันธ์ชุดใดที่มีค่าความเชื่อมั่นมากกว่าหรือเท่ากับค่าความเชื่อมั่นขั้นต่ำ กฎนั้นจะเรียกว่ากฎที่น่าสนใจและนำไปใช้ประโยชน์ต่อไป

สำหรับงานวิจัยทางด้านการค้นหากฎความสัมพันธ์ ขั้นตอนวิธีที่ได้รับการยอมรับและเป็นที่นิยมสำหรับการหากฎความสัมพันธ์ คือ ขั้นตอนวิธี อะพริโอริ ซึ่งมีลักษณะการทำงานแบบ Level-wise-step และมีขั้นตอนหลัก 2 ขั้นตอนคือการสร้างไอเท็มเซตคู่แข่งด้วยการเชื่อมไอเท็มเซต (Join) และการตัด (Prune) ไอเท็มเซตที่ไม่สามารถเป็นไอเท็มเซตที่เกิดบ่อยทิ้งไป

อย่างไรก็ตาม เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้าไปในฐานข้อมูลเดิม การค้นหากฎความสัมพันธ์ด้วยวิธีการของ อะพริโอริ จะต้องทำการประมวลข้อมูลในฐานข้อมูลทั้งหมดอีกครั้งเพื่อสร้างไอเท็มเซตคู่แข่งใหม่ และสแกนฐานข้อมูลเพื่อหาค่าสนับสนุนของไอเท็มเซตคู่แข่งใหม่ในทุกๆ รอบ ส่งผลต่อเวลาที่ใช้ในการประมวลผลถูกใช้มากเกินความจำเป็นและไม่เกิดประสิทธิภาพในการทำงาน จึงได้มีผู้นำเสนอขั้นตอนวิธี เอฟยูพี เพื่อการปรับปรุงกฎความสัมพันธ์เมื่อมีการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิม ซึ่งเป็นขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ (Incremental association rule discovery) การทำงานของ เอฟยูพี จะใช้ไอเท็มเซตที่เกิดบ่อยที่ได้จากการประมวลผลในฐานข้อมูลเดิมมาช่วยลดจำนวนไอเท็มเซตคู่แข่งที่จะถูกนำไปสแกนในฐานข้อมูลเดิม ด้วยวิธีการดังกล่าว ทำให้ เอฟยูพี ใช้เวลาที่ใช้ในการประมวลผลน้อยกว่า อะพริโอริ

อย่างไรก็ตาม เอฟยูพี ยังมีการสแกนฐานข้อมูลเดิมในทุกรอบ k ซึ่งหากข้อมูลมีขนาดใหญ่มาก เอฟยูพี จะสูญเสียเวลาในการประมวลผลให้กับการทำงานส่วนนี้ค่อนข้างมาก ดังนั้นเพื่อลดจำนวนการสแกนฐานข้อมูลเดิมให้เหลือจำนวนรอบการสแกนที่น้อยที่สุด ได้มีนักวิจัยนำเสนอขั้นตอนวิธีเพื่อลดปัญหาการกลับไปสแกนฐานข้อมูลเดิม อาทิ ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ ขั้นตอนวิธี พรีลาร์จ และขั้นตอนวิธีการการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นเป็นต้น

ขั้นตอนวิธี เนกาทีฟบอร์เดอร์ จะมีการเก็บทั้งไอเท็มเซตที่เกิดบ่อยและไอเท็มที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยที่เรียกว่าบอร์เดอร์ไอเท็มเซต (Border itemset) เพื่อลดจำนวนรอบการสแกนฐานข้อมูล ซึ่งแม้ว่าจะลดจำนวนการสแกนให้เหลืออย่างมากเพียง 1 ครั้ง แต่ก็แลกด้วยการเก็บบอร์เดอร์ไอเท็มเซตจำนวนมหาศาล และต้องใช้เวลาในการเชื่อมไอเท็มเซตมากขึ้นกว่าเดิม ขั้นตอนวิธี พรีลาร์จ และการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจึงเสนอแนวคิดเพื่อลดจำนวนบอร์เดอร์ไอเท็มเซต โดย พรีลาร์จ จะกำหนดให้มีพารามิเตอร์เพิ่มอีก 2 ตัว คือ ขอบเขตบน (Upper support) และขอบเขตล่าง (Lower support) เพื่อกรองไอเท็มเซตที่ไม่ผ่านค่าพารามิเตอร์นี้ออกไป ทำให้จำนวนบอร์เดอร์เซตลดลง ส่วนขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น ได้นำเสนอเทคนิคการประมาณค่าความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงสำหรับเก็บไว้เพื่อนำไปประมวลผลเมื่อมีการเพิ่มข้อมูลชุดใหม่ในรอบถัดไปและสามารถลดจำนวนครั้งของการสแกนฐานข้อมูลเดิมให้เหลือเพียงครั้งเดียว

แม้ว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจะทำงานได้รวดเร็วและลดจำนวนไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อย แต่ยังมีข้อจำกัด 3 ด้าน ได้แก่ 1) ปัญหาในการหาความน่าจะเป็นในกรณีที่ต้องคำนวณค่าแฟคทอเรียลของจำนวนจริงที่มีค่ามาก 2) ปัญหาข้อจำกัดในด้านขนาดของชุดข้อมูลใหม่ที่จะเพิ่มเข้ามาที่จะต้องทราบแน่นอน จึงจะสามารถคำนวณค่าความน่าจะเป็นได้ และ 3) การคำนวณความน่าจะเป็นให้กับไอเท็มเซตทุกตัวเพื่อค้นหาไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง

ด้วยปัญหาดังกล่าวข้างต้นของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นงานวิจัยฉบับนี้จึงดำเนินการศึกษาค้นคว้าเพื่อปรับปรุงอัลกอทึมการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจกปกติ เพื่อนำหลักการประมาณค่าการแจกแจงแบบปกติมาประยุกต์ใช้เพื่อให้ขั้นตอนวิธีที่นำเสนอในงานวิจัยนี้สามารถแก้ไขปัญหาการคำนวณค่าแฟคทอเรียล และทำงานได้โดยไม่จำเป็นต้องจำกัดขนาดของชุดข้อมูลใหม่ที่จะถูกเพิ่มเข้ามารวมถึงไม่จำเป็นต้องคำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัว

แนวคิดของขั้นตอนวิธีที่นำเสนอในงานวิจัยนี้ ได้ใช้หลักการประมาณค่าแบบการแจกแจงปกติมาช่วยแก้ปัญหาในเรื่องการคำนวณค่าแฟคทอเรียลของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นและนำหลักคิดเกี่ยวกับค่าสนับสนุน (Support factor) ของไอเท็มเซตมาใช้แทนค่าสนับสนุนที่เป็นค่านับ (Support count) เพื่อให้การคำนวณหาความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงไม่จำเป็นทราบขนาดของฐานข้อมูลใหม่ก็สามารถคำนวณได้

ในการทดลองเพื่อวัดความถูกต้องของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ จะทดสอบโดยพิจารณาจากจำนวนไอเท็มเซตที่เกิดบ่อยที่ขั้นตอนวิธีค้นหาพบ มีจำนวนเท่ากับจำนวนไอเท็มเซตที่ขั้นตอนวิธี อะพริโอริ ค้นหาพบ และเป็นไอเท็มตัวเดียวกัน ซึ่งผลลัพธ์ในการทดลองข้อมูลทุกชุดทั้งชุดข้อมูลสังเคราะห์และชุดข้อมูลจริง Tafeng ปรากฏว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ สามารถค้นหาไอเท็มเซตที่เกิดบ่อยได้ถูกต้องและครบถ้วน

สำหรับการทดสอบประสิทธิภาพด้านเวลาที่ใช้ในการทำงานของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ได้มีการทดสอบโดยใช้ข้อมูลสังเคราะห์จำนวน 3 ชุดที่สร้างขึ้นโดยกำหนดค่าพารามิเตอร์ที่แตกต่างกัน และชุดข้อมูลจริง Tafeng ซึ่งนำไปทดสอบกับค่าสนับสนุนขั้นต่ำจำนวน 4 ค่า และข้อมูลชุดใหม่ที่ถูกเพิ่มเข้ามาหลายขนาด โดยจะมีการทดลองทั้งการเพิ่มชุดข้อมูลที่มีขนาดเท่ากันอย่างต่อเนื่อง และขนาดไม่เท่ากันอย่างต่อเนื่อง

ผลการทดลองพบกว่า ขั้นตอนวิธีขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ใช้เวลาในการประมวลผลเร็วกว่าขั้นตอนวิธีเอฟยูพี ขั้นตอนวิธีเนกาทีฟบอร์เดอร์ ขั้นตอนวิธีพรีลาร์จ และขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นจากการวิเคราะห์ผลการทดลอง พบว่า การที่ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติทำงานได้รวดเร็วกว่า มีสาเหตุต่างๆ ดังนี้

1. ขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าด้วยการแจกแจงปกติมีการเก็บไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุง เพื่อนำไปช่วยในการประมวลผลในครั้งถัดไปที่มีข้อมูลชุดใหม่เพิ่มเข้ามา โดยไม่ต้องสแกนฐานข้อมูลเดิมทุกรอบเหมือนขั้นตอนวิธี เอฟยูพี เว้นเสียแต่มีไอเท็มที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยเกิดขึ้นระหว่างกระบวนการ จึงจะถูกนำไปสแกนฐานข้อมูลเดิมเพียงครั้งเดียวในขั้นตอนสุดท้าย

2. ขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าด้วยการแจกแจงปกติมีการเก็บไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยแต่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงน้อยกว่าขั้นตอนวิธี เนกาทีฟบอร์เดอร์ และ พรีลาร์จ ทำให้ช่วยลดเวลาในการเชื่อมไอเท็ม

เซตเพื่อสร้างไอเท็มเซตคู่แข่ง ส่งผลให้ขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าด้วยการแจกแจงปกติมีเวลาในการประมวลผลที่รวดเร็วกว่าขั้นตอนวิธี เนกาทีฟบอร์เดอร์ และ พรีลาร์จ

3. แม้ว่าขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าด้วยการแจกแจงปกติจะมีการเก็บไอเท็มเซตที่ไม่ใช่ไอเท็มเซตที่เกิดบ่อยแต่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงมากกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น แต่ก็เป็นการเก็บที่ครอบคลุมกว่า จึงทำให้ไอเท็มในเซต *Temp_scanDB* ที่จะถูกนำไปสแกนในฐานข้อมูลเดิมมีจำนวนน้อยกว่ามากเมื่อเทียบกับขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นอีกทั้งขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าด้วยการแจกแจงปกติมีการคำนวณค่าความน่าจะเป็นของไอเท็มเซตที่มีโอกาสจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงเพียงครั้งเดียว จึงทำให้ขั้นตอนวิธีเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าด้วยการแจกแจงปกติมีเวลาในการประมวลผลที่รวดเร็วกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

อย่างไรก็ตามในการทดสอบด้วยชุดข้อมูลจริง Tafeng จะเห็นได้ว่า ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ ขั้นตอนวิธีเอฟยูพี มีเวลาที่แตกต่างกันไม่มากนัก สาเหตุเนื่องมาจากไอเท็มเซตที่เกิดบ่อยที่ค้นหาได้ มีขนาดสูงสุดคือ 3 จึงทำให้ขั้นตอนวิธีเอฟยูพีมีการนำไอเท็มเซตกลับไปสแกนในฐานข้อมูลเดิมเพียง 3 รอบ ซึ่งแตกต่างจากการทดลองในชุดข้อมูลสังเคราะห์ที่ขั้นตอนเอฟยูพี ต้องนำไอเท็มเซตกลับไปสแกนฐานข้อมูลเดิมจำนวนหลายรอบ ในขณะที่ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติ มีเวลาในการประมวลผลที่เร็วกว่าขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นไม่มากนัก เนื่องจากไอเท็มเซตที่ถูกนำกลับไปสแกนในฐานข้อมูลเดิมมีจำนวนแตกต่างกันไม่มาก ทำให้เวลาที่ใช้ในการประมวลผลมีความใกล้เคียงกัน

5.2 ข้อเสนอแนะ

แนวคิดของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติถูกออกแบบมาเพื่อแก้ปัญหาในเรื่องการคำนวณค่าแฟคทอเรียลด้วยความน่าจะเป็นแบบทวินามของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็นและนำหลักคิดเกี่ยวกับค่าสนับสนุน (Support factor) ของไอเท็มเซตมาใช้แทนค่าสนับสนุนที่เป็นค่านับ (Support count) เพื่อให้การคำนวณหาความน่าจะเป็นของไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตที่เกิดบ่อยในฐานข้อมูลปรับปรุงไม่จำเป็นทราบขนาดของฐานข้อมูลใหม่ก็สามารถคำนวณได้ นอกจากนี้ ยังได้

เสนอการคำนวณที่ใช้ทดสอบไอเท็มเซตที่คาดว่าจะเป็นไอเท็มเซตในฐานข้อมูลปรับปรุงเพียงค่าเดียว ซึ่งทำให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติไม่จำเป็นต้องคำนวณค่าความน่าจะเป็นให้แก่ไอเท็มเซตทุกตัวเช่นเดียวกับขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์โดยอาศัยหลักความน่าจะเป็น

แม้ว่าการทดสอบประสิทธิภาพของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติจะให้ผลลัพธ์ทางด้านเวลาที่ใช้ในการประมวลผลที่รวดเร็วกว่าขั้นตอนวิธีอื่นๆ ที่ถูกนำมาทดสอบ อย่างไรก็ตามของขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติสามารถประมวลผลได้เฉพาะการเพิ่มข้อมูลชุดใหม่เข้ามาในฐานข้อมูลเดิมเท่านั้น และไม่สามารถประมวลผลได้ในกรณีที่มีการลบหรือแก้ไขข้อมูลเดิมได้ ดังนั้นหากมีการนำเทคนิคต่างๆ เข้ามาช่วยในการประมวลผลกรณีที่มีการลบหรือแก้ไขข้อมูลเดิมจะทำให้ขั้นตอนวิธีการเพิ่มขยายกฎความสัมพันธ์ด้วยการประมาณค่าแบบการแจกแจงปกติมีประสิทธิภาพมากยิ่งขึ้น

เอกสารอ้างอิง

[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] Toivonen H, Sampling Large Database for Association Rules, **In proccedings of the 22th International Conference on Very Large Database (VLDB'96)**, September 1996, pp.134 – 145.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] Zhang S., Zhang C. and Yan X., "Post-mining: maintenance of association rules by weighting", **Information System**, 2003. pp.691 – 707.

[10] Dudek D., Zgrzywa A., "The Incremental Method for Discovery of Association Rules", Springer berlin/Heidelberg, Volume 30, 2005, pp. 153-160.

[11] 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.

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

[13] Lee, C.H., Lin, C.R. and Chen, M., "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining." **ACM**,2000.

[14] Chang, C.-H. and Yang, S.-H.: "Enhancing SWF for Incremental Association Mining by ไอเท็มเซต Maintenance"; **Proceeding of 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining,** April 2003.

[15] C.I. Ezeife and Y. Su. Mining Incremental Association Rules with Generalized FP-Tree. **Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence** pp.147-160, May 2002.

[16] 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.

[17] Feller W. **An Introduction to Probability Theory and its applications,** Third Edition, Volume 1, New York, 1968.

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

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

[20] Han J., Kamber M., Pei J., "**Data Mining: Concepts and Techniques**", 3rd Edition. Morgan Kaufmann Publishers, Waltham, USA, 2012.

[21] Grinstead, Charles M., J. Lauries Snell., "Introduction to Probability", 2nd Edition, American of Mathematical Society, 1997.

ภาคผนวก

ภาคผนวก ก

การพิสูจน์การประมาณค่าความน่าจะเป็นของการแจกแจงทวินามด้วยการแจกแจงปกติ

การพิสูจน์การประมาณค่าความน่าจะเป็นของการแจกแจงทวินามด้วยการการแจกแจงปกติ

As n grows large, for k in the neighborhood of np it can be approximated

$$\binom{n}{k} p^k q^{n-k} \approx \frac{1}{\sqrt{2\pi npq}} e^{-\frac{(k-np)^2}{2npq}}, \qquad p+q=1, \quad p,q>0$$

In the sense that the ratio of the left-hand side to the right hand side converges to 1 as $n \to \infty$

บทพิสูจน์

According to Stirling's formula, it can be replaced the factorial of a large number n, with the approximation:

$$n! \cong n^n e^{-n} \sqrt{2\pi n} \qquad \text{as } n \to \infty$$

Thus,

$$\begin{aligned}
\binom{n}{k} p^k q^{n-k} &= \frac{n!}{k!(n-k)!} p^k q^{n-k} \\
&= \frac{n^n e^{-n} \sqrt{2\pi n}}{\left(k^k e^{-k} \sqrt{2\pi k}\right)\left((n-k)^{n-k} e^{-(n-k)} \sqrt{2\pi(n-k)}\right)} p^k q^{n-k} \\
&= \left(\frac{\sqrt{2\pi n}}{\sqrt{2\pi k}\sqrt{2\pi(n-k)}}\right) \cdot \left(\frac{e^{-n}}{e^{-k} e^{-(n-k)}}\right) \cdot \left(\frac{n^n}{k^k (n-k)^{n-k}} p^k q^{n-k}\right) \\
&= \sqrt{\frac{n}{2\pi k(n-k)}} .1. \left(n^n \left(\frac{p}{k}\right)^k \left(\frac{q}{n-k}\right)^{(n-k)}\right) \\
&= \sqrt{\frac{n}{2\pi k(n-k)}} \cdot \left(n^{n-k} n^k \left(\frac{p}{k}\right)^k \left(\frac{q}{n-k}\right)^{(n-k)}\right) \\
&= \sqrt{\frac{n}{2\pi k(n-k)}} \cdot \left(\left(\frac{np}{k}\right)^k \left(\frac{nq}{n-k}\right)^{(n-k)}\right) \\
&= \sqrt{\frac{n}{2\pi k(n-k)}} \left(\frac{k}{np}\right)^{-k} \left(\frac{n-k}{nq}\right)^{-(n-k)}
\end{aligned}$$

$$= \sqrt{\frac{n}{2\pi k(n-k)}}\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1-x\sqrt{\frac{p}{nq}}\right)^{-(n-k)} \quad , \quad x := \frac{k-np}{\sqrt{npq}}$$

$$= \sqrt{\frac{n}{2\pi k(n-k)}.\frac{n^{-2}}{n^{-2}}}\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1-x\sqrt{\frac{p}{nq}}\right)^{-(n-k)}$$

$$= \sqrt{\frac{n^{-1}}{2\pi k(n-k)n^{-2}}}\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1-x\sqrt{\frac{p}{nq}}\right)^{-(n-k)}$$

$$= \sqrt{\frac{n^{-1}}{2\pi \frac{k}{n}.\frac{(n-k)}{n}}}\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1+x\sqrt{\frac{p}{nq}}\right)^{-(n-k)}$$

$$= \sqrt{\frac{n^{-1}}{2\pi \frac{k}{n}.\left(1-\frac{k}{n}\right)}}\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1+x\sqrt{\frac{p}{nq}}\right)^{-(n-k)}$$

$$\cong \sqrt{\frac{n^{-1}}{2\pi p.(1-p)}}\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1+x\sqrt{\frac{p}{nq}}\right)^{-(n-k)} \quad , \text{ as } k \to np \text{ we get } \frac{k}{n} = p$$

$$= \sqrt{\frac{1}{2\pi npq}}\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1+x\sqrt{\frac{p}{nq}}\right)^{-(n-k)} \quad , \quad p+q=1$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{\ln\left[\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\left(1-x\sqrt{\frac{p}{nq}}\right)^{-(n-k)}\right]\right\} \quad , \quad e^{\ln(y)} = y$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{\ln\left[\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\right]+\ln\left[\left(1-x\sqrt{\frac{p}{nq}}\right)^{-(n-k)}\right]\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{\ln\left[\left(1+x\sqrt{\frac{q}{np}}\right)^{-k}\right]+\ln\left[\left(1-x\sqrt{\frac{p}{nq}}\right)^{-(n-k)}\right]\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-k.\ln\left[1+x\sqrt{\frac{q}{np}}\right]-(n-k)\ln\left[1-x\sqrt{\frac{p}{nq}}\right]\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-(np+x\sqrt{npq})\ln\left[1+x\sqrt{\frac{q}{np}}\right]-\left(nq-x\sqrt{npq}\right)\ln\left[1-x\sqrt{\frac{p}{nq}}\right]\right\}$$

The last line follows from the definition of x. Now using the Taylor series expansion of the function $\ln(1\pm x)$, it arrives at:

$$\frac{1}{\sqrt{2\pi npq}}\exp\left\{-\left(np+x\sqrt{npq}\right)\left(x\sqrt{\frac{q}{np}}-\frac{x^2q}{2np}+\ldots\right)-\left(nq-x\sqrt{npq}\right)\left(-x\sqrt{\frac{p}{nq}}-\frac{x^2p}{2nq}-\ldots\right)\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-\left(x\sqrt{npq}-\frac{1}{2}x^2q+x^2q+\ldots\right)-\left(-x\sqrt{npq}-\frac{1}{2}x^2p+x^2p+\ldots\right)\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-\left(x\sqrt{npq}+\frac{1}{2}x^2q+\ldots\right)-\left(-x\sqrt{npq}+\frac{1}{2}x^2p+\ldots\right)\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-x\sqrt{npq}-\frac{1}{2}x^2q+x\sqrt{npq}-\frac{1}{2}x^2p-\ldots\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-\frac{1}{2}x^2(q+p)-\ldots\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-\frac{1}{2}x^2-\ldots\right\}$$

$$\cong \frac{1}{\sqrt{2\pi npq}}\exp\left\{-\frac{1}{2}x^2\right\} \qquad \text{as } n\to\infty \text{ we get } x\to 0$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-\frac{1}{2}\left(\frac{k-np}{\sqrt{npq}}\right)^2\right\}$$

$$= \frac{1}{\sqrt{2\pi npq}}\exp\left\{-\frac{(k-np)^2}{2npq}\right\}$$

Thus,

$$\binom{n}{k}p^kq^{n-k} \cong \frac{1}{\sqrt{2\pi npq}}e^{-\frac{(k-np)^2}{2npq}}.$$

ภาคผนวก ข

ผลงานวิจัยที่เกี่ยวข้องกับวิทยานิพนธ์และได้รับการตีพิมพ์

1. Araya Ariya, Worapoj Kreesuradej (2012). **Batch Fast Update Algorithm for Incremental Association Rule Discovery**. The 17th International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12). B-Con Plaza, Beppu, Oita, Japan, January 19 – 20, 2012, pp. 799 – 802.

2. Araya Ariya, Worapoj Kreesuradej (2013). **Probability-Based Incremental Association Rule Discovery Using The Normal Approximation.** Proceedings of the 2013 IEEE 14th International Conference on Information Reuse and Integration (IEEE IRI 2013). San Francisco, California, USA, August 14 – 16, 2013, pp. 432 – 439.

3. Araya Ariya, Worapoj Kreesuradej (2015). **An Enhanced Incremental Association Rule Discovery with a lower minimum support.** The 20th International Symposium on Artificial Life and Robotics 2015 (AROB 20th '15). B-Con Plaza, Beppu, Oita, Japan, January 21 – 23, 2015, pp. 192 – 197.

4. Araya Ariya, Worapoj Kreesuradej. **An Enhanced Incremental Association Rule Discovery with a lower minimum support.** (ได้รับการตอบรับและรอการตีพิมพ์ในวารสาร Artificial Life and Robotics หมายเลข DOI: 10.1007/s10015-016-0288-3)

PROCEEDINGS OF THE SEVENTEENTH INTERNATIONAL SYMPOSIUM ON ARTIFICIAL LIFE AND ROBOTICS

(AROB 17th '12)

Jan. 19-21, 2012

B-Con Plaza, Beppu, Oita, JAPAN

Editors: Masanori Sugisaka and Hiroshi Tanaka

Publisher: ALife Robotics Co., Ltd.
Publication Date: Jan. 10, 2012
ISBN 978-4-9902880-6-8

Proceedings of the Seventeenth International Symposium on

ARTIFICIAL LIFE AND ROBOTICS

(AROB 17th '12)

January 19- 21, 2012
B-Con Plaza, Beppu, Oita, Japan

Editors: Masanori Sugisaka and Hiroshi Tanaka

Batch fast update algorithm for incremental association rule discovery

Araya Ariya[1] , Worapoj Kreesuradej[2]

King Mongkut's Institute of Technology Ladkrabang, Thailand

[1]araya_aa@hotmail.com , [2]worapoj@it.kmitl.ac.th

Abstract: When new transactions are inserted into an original database, the existing rules may be change. An incremental association rule mining is an approach to deal with such problem. This paper proposes an algorithm for mining incremental association rules, called batch fast update (BFUP). The proposed algorithm improves the performance of FUP algorithm by reducing a number of scanning times of an original database. The experimental results show that an execution time of BFUP is much faster than that of FUP.

Keywords: Incremental association rules mining, Association rule mining, Data mining

1. INTRODUCTION

An association rule mining, one of the important tasks in data mining, is a well-known research topic that many researchers propose a large number of algorithms for solving association rule discovering problems. This problem was first presented by Agrawal et al [1] which analyzes the behavior of customer purchasing to help business make a decision. The results show co-occurrence buying items called frequent patterns, which can generate interesting rules. From that research, the problem statement of an association rule mining is 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 contain X if and only if $X \subseteq T$. Let X and Y are an item where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y \neq 0$. Each set of items , itemsets, is called a frequent itemset if and only if its support is greater than or equal to support threshold s%. It calculates from a number of transactions in DB that contain $X \cup Y$. An association rule can be shown in $X \Rightarrow Y$ form. 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 specified by user.

After an association rule mining was revealed, it motivated many researchers to extend this research area in a lot of issues. An incremental association rule mining is the one of an association rule mining issue which maintains association rules when new transactions are appended to an original database.

One research issue of an incremental association rule mining is reducing running time of the algorithms by minimizing the number of times to scan an original database. This paper also works on this issue. An algorithm for mining incremental association rules, called batch fast update (BFUP), is proposed. This algorithm has only one original database scanning.

The paper is organized as follows. The literatures of an association rule mining and an incremental association rule mining are reviewed in section 2. The problem statement of an incremental mining on association rule in dynamic database and FUP algorithm are detailed in section 3. The proposed algorithm and its experiment are presented in section 4 and 5 respectively. The paper conclusion and future work are briefed in section 6.

2. RELATED WORK

Mining association rules was first proposed by Agrawal et al [1] which finds a correlation between itemsets in a transaction database. The algorithm has 2 steps: finding frequent itemsets (sometimes they are called large itemsets) and generating rules. Subsequent years, Apriori [2], the most popular algorithm, was proposed to discover frequent itemsets.

When a database, called a dynamic database, is inserted new transactions, frequent itemsets can be changed after inserting new transactions into the dynamic database. Therefore, an association rule discovery algorithm for a dynamic database has to maintain frequent itemsets when new transactions are inserted into the dynamic database.

One approach to find the new frequent itemsets is to rerun Apriori algorithm for the whole transactions of the dynamic database. This approach is not efficient because all the computation done initially at finding out the old large itemsets are wasted and all large itemsets have to be computed again from scratch [3].

Cheung et al [3] proposed fast update algorithm, FUP, to solve a rules maintenance problem by using the previous knowledge to find frequent itemsets in updated database. The concept of FUP is re-using frequent itemsets of previous mining to update with frequent itemsets of an incremental database. Although FUP can decrease a number of candidate itemsets for scanning original database, it still needs to scan an original database k times when new frequent itemsets are found. This can degrade the performance of FUP algorithm.

In our observation, the advantage of FUP are re-using frequent itemsets of a previous mining to prune itemsets which cannot be a frequent itemset in updated database and reducing candidate itemsets to scan in an original database.

However, the disadvantage of FUP is the algorithm needs to scan an original database equal to a size of k frequent itemsets, e.g., if maximum size of k frequent itemsets is k=5, FUP needs to scan an original database for 5 times.

From this problem, this paper proposes an incremental association rule mining algorithm to improve the performance of FUP algorithm by reducing a number of scanning times of an original database.

3. INCREMENTAL ASSOCIATION RULES MINING

3.1 Problem statement

In a dynamic database, new transactions are appended to a database; accordingly, the previous valid rules may be invalid. The problem statement for an incremental association rule is defined as follows.

Let DB is an original database. An increment database db is the new transactions which are inserted into DB. Updated database UD is the combining between original database and increment database, i.e, $UD = DB \cup db$. A number of transactions of an original database, an increment database and an updated database are |DB|, |db| and |UD| = |DB|+|db| respectively.

Before updating activity, L is the frequent itemsets in DB if and only if X.support $\geq$ s×|DB|. After updating activity, L' is the frequent itemsets in updated database if and only if X.support $\geq$ s×|UD|.

According to Tsai et al [4], when news transactions are insert into an original database, an itemset, i.e. X, can be categorized into 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

Form these cases of itemsets are mentioned above, it is easy to discovered updated frequent itemsets for the itemset of case 1 and 2 because their count in DB and UD are known, therefore, an updating activity is a trivial task. The itemset in case 4 is unimportant because it cannot change an association rule. The most serious case is the 3rd because it needs to rescan an original database for updating its count. Thus, discovering itemsets in case 3 is the most important problem in an incremental association rule mining. In section 3.2, the method to solve that problem is reviewed and section 4, batch fast update algorithm is presented how to improve a performance of FUP.

3.2 FUP algorithm

Apriori [2] is successful for finding frequent itemsets in a database. However, it is unsuitable for mining in dynamic database. Cheung et al [3] have been proposed an incremental algorithm which has a good performance for mining association rules in dynamic database. The concept of FUP is reviewed briefly in this section.

The operation of FUP has 2 phases: 1-iteration and k-iteration where $k \geq 2$. In the first phase, an increment database is scanned for finding candidate 1-itemsets C_1 and its count. After that loser and winner itemsets are found. Finding loser and winner itemsets, C_1 are divided into 2 types: a member and not a member of previous frequent itemsets in an original database. The first type is updated its count and pruned loser itemsets if its updated count is less than s*|UD|. The second type is scanned to an original database if and only if it is a frequent itemset (winner) in an increment database, i.e., its count is greater than or equal to s*|db|. Both types which satisfy by them threshold can be frequent 1-itemsets in updated database L_1'(winner).

The second phase has 3 steps: filtering out loser itemsets, generating candidate k-itemsets $C_{k\geq 2}$ and finding new frequent itemsets. Firstly, FUP filters out losers from L_k (L_k in DB). Given $Y \in L_k$ and $X \in L_{k-1} - L_{k-1}'$, Y is a loser iff $X \in Y$. For the remaining L_k, they are scanned to an increment database and updated their count. Then they are checked for finding a winner or loser itemset similar to the first phase.

Secondly, C_k is generated by using Apriori-gen. Any C_k is pruned if and only if $Y \in C_k$ where Y is the loser from L_k. This step is a key to reduce a number of C_k before scanning an original database. Finally, new frequent itemsets L_k' are found with the same method as the first phase.

4. BATCH FAST UPDATE ALGORITHM

In this section, an algorithm for mining incremental association rules, batch fast update (BFUP) is presented. The proposed algorithm is assumed that two thresholds, minimum support s% and minimum confidence c%, are static. This algorithm needs only one original database pass and infrequent itemsets are not required. The notation used in this section is defined in Table 1.

Table 1. The notation for Batch fast update algorithm

notation	meaning
DB	original database
db	increment database
UD	updated database
s	minimum support
L_k^{DB}	frequent k-itemset in DB
L_k^{UD}	frequent k-itemset in UD
C_k	candidate k-itemset
|DB|	a number of transactions in DB
|db|	a number of transactions in db
|UD|	a number of transactions in UD
X.count	a support of an itemset
Temp_scanDB	itemsets which are scanned in DB

The algorithm has 2 phases: an increment updating phase and a re-scanning original database phase. The first phase is shown in figure 1. At each iteration, an increment

database is scanned to find candidate itemsets, i.e., C_k, and their support counts. Basically, candidate itemsets are divided into 2 types: a member and not a member of previous frequent itemsets of an original database. Candidate itemsets of the first type becomes updated frequent itemsets, i.e., L_k^{UD}, if and only if their updated support count is greater than or equal to s*|UD|. Candidate itemsets of the second type are kept in temp_scanDB for rescanning in an original database if and only if their count plus |DB|-1 is greater than s*|UD|. On the other hand, Candidate itemsets of the second type are pruned if and only if their count plus |DB|-1 is less than s*|UD|.

For generating candidate itemset C_k, Apriori-gen is applied. Apriori generates C_k with $L_{k-1}*L_{k-1}$, whereas BFUP generates C_k with L_k^{UD} * $temp\_scanDB_k$.

The second phase of BFUP algorithm is shown in figure 2. After an increment updating phase is ended, all itemsets in temp_scanDB are re-scanned in an original database and updated their support count. Then, all itemsets in temp_scanDB are checked to find updated frequent itemsets. Let $X \in temp\_scanDB$, X can be an updated frequent itemset, i.e., L_k^{UD}, if and only if $X.count \geq s*|UD|$.

Algorithm 1 An increment updating phase ()

```
Input : db, s, L_k^DB, C_1^DB, |DB|
Output: L_k^UD
1   |UD|=|DB|+|db|
2   k = 1
3   scan db for all X
4   for all X∈L_1^DB or X∈C_1^DB
5     update X.count
6     if X.count ≥ s*|UD|
7       X → L_k^UD
8   for all X ∉ L_1^DB or X∉C_1^DB
9     if X.count+|DB|-1 ≥ s*|UD|
10      X → temp_scanDB
11  k=2
12  while (L_{k-1}^UD ∪ temp_scanDB_k) > 1
13    C_k = L_{k-1}^UD * temp_scanDB_k
14    // using Apriori_gen()
15    scan db for all C_k
16    for all X∈ C_k do
17      for all X ∈ L_k^DB
18        update X.count
19        if X.count ≥  s*|UD|
20          X → L_k^UD
21      for all X ∉ L_k^DB
22        if X.count+|DB|-1 ≥ s*|UD|
23          X → temp_scanDB
24    k++
25  end loop
26  rescan_original()
27  L_k^UD = L_k^UD ∪ tempL
28  return L_k^UD
```

Fig. 1. An increment updating phase

Algorithm 2 a re-scanning original database phase ()

```
Input DB, temp_scanDB,s,|UD|
Output tempL
1   if temp_scanDB ≠ ∅
2       scan DB for all X ∈ temp_scanDB
3       update X.count
4       if X.count ≥ s*|UD|
5           X → tempL
6   endif
7   return tempL
```

Fig. 2. A re-scanning original database phase

5. EXPERIMENT

The proposed algorithm in this paper aims to improve the performance of FUP. To evaluate the performance of batch fast update (BFUP) algorithm, this algorithm is implemented and tested on a PC with a 2.93 GHz Intel Core i7 and 3 GB main memory. The experiment is tested with 2 synthetic datasets which are generated by using technique in Agrawal [1]. The first dataset is T10I4D100K which has 100,000 transactions. The second dataset is T10I4D50K which has 50,000 transactions. Both datasets are appended by an increment database that has 1,000 transactions.

For the first original database, i.e., T10I4D100K, the experiment is conducted with 0.1%, 0.3% and 0.5% minimum support thresholds. The average of an execution time is shown in table 2 and figure 3 respectively. For comparison, the results are compared with FUP.

Table 2. Average of Execution time for I10T4D100K

min sup	algorithm	execution time (sec.)	a number of frequent itemset	maximum frequent itemset (size of k)
0.1%	FUP	175890.2901	17,127	L_{10}
	BFUP	43580.3014		
0.3%	FUP	19645.1002	1,991	L_7
	BFUP	11678.5801		
0.5%	FUP	9771.1612	862	L_2
	BFUP	9105.8170		

For the second original database, i.e., T10I4D50K, the experiment is conducted with 0.2%, 0.3% and 0.4% minimum support thresholds. The average of an execution time is shown in table 3 and figure 4 respectively.

From the results of the both datasets, they are shown that an execution time of BFUP is much faster than that of FUP. Furthermore, we observe that the more size k is increasing, the execution time between FUP and BFUP is more different.

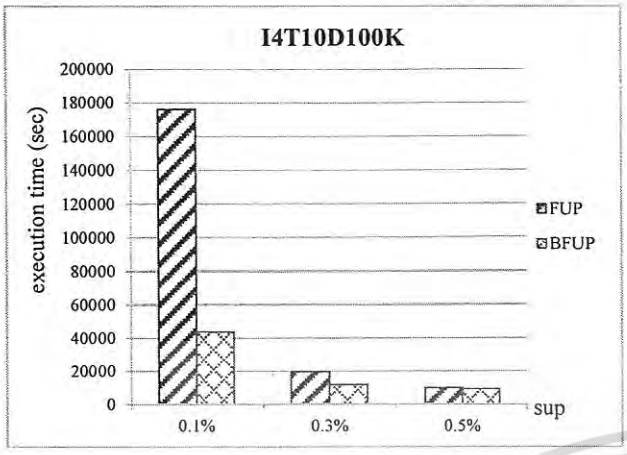


Fig. 3. Execution time comparison for I10T4D100K

Table 3. Average of Execution time for I10T4D50K

min sup	algorithm	execution time (sec.)	a number of frequent itemset	maximum frequent itemset (size of k)
0.2%	FUP	36012.0041	5,307	L_{10}
	BFUP	15402.0908		
0.3%	FUP	17695.8894	1,995	L_7
	BFUP	12906.0013		
0.4%	FUP	12363.9417	1,051	L_3
	BFUP	10866.7162		

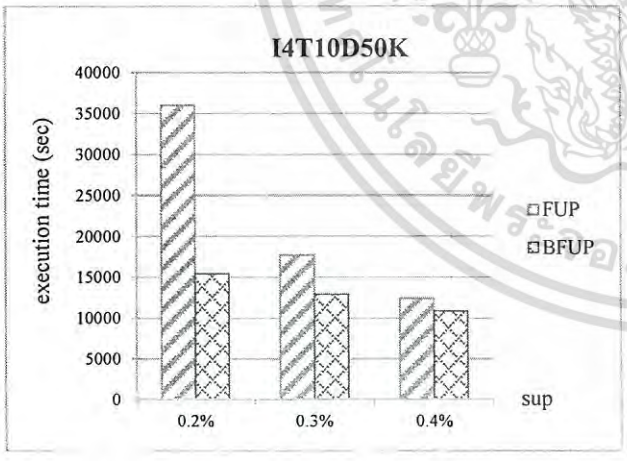


Fig. 4. Execution time comparison for I10T4D50K

6. CONCLUSION

An incremental association rules mining algorithm called batch fast update (BFUP), is proposed. The concept of this algorithm is based from Apriori and FUP algorithm. Although batch fast update algorithm has an execution time better than that of FUP, a large number of temp_scanDB are kept. In the future research, the algorithm for reducing temp_scanDB will be proposed.

7. REFERENCES

[1] Agrawal R, Imielinski T, Swami A (1993), 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] Agrawal R, Srikant R (1994), 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] Cheung D.W., Han J, Ng V.T., Wong C.Y., Maintenance of Discovered Association rules in Large Databases: An incremental updating technique, In 12th IEEE International Conference on Data Engineering, pp 106-114, 1996.

[4] Tsai Paulry S.M., Lee Chih-Chong, Chen Abree L.P. (2005), 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 archieve, 1999.

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

Committee: Lotfi Zadeh, Stuart Rubin, Shu-Ching Chen, Mei-Ling Shyu, Min-Yuh Day, Tanvir Ahmed, Taghi M. Khoshgoftaar, Lin Yang, Reda Alhajj, Gordon K. Lee, Suresh Vadhva, Li Tan, Wei-Bang Chen, Eric Gregoire, Wen-Lian Hsu, Xingquan (Hill) Zhu, Thouraya Bouabana-Tebibel, June R. Massoud, Du Zhang, Nathalie Baracaldo

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 $Prob_{pt}$. $Prob_{pt}$ is a constant threshold specified by users. $Prob_{pt}$ 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 $Prob_{pt}$ 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"

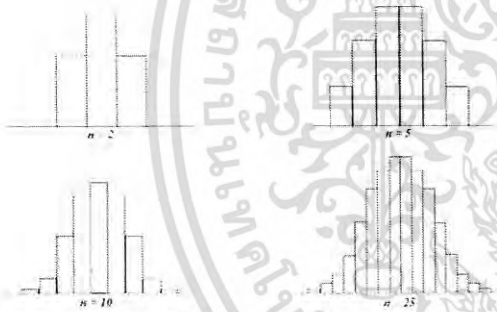


Figure 1. Binomial distribution with p = ½ and n = 2, 5, 10 and 25

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 $prob_{ptt}$. 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
$Prob_{pl}$	probability value threshold defined by user

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 | $\delta_x^{DB} <$ s×|DB|}
5 calculate ptt_x //using equation (14)
6 calculate probability of expected itemset X ($ProbEF_X$) //using equation (12)
7 ρ^{DB} = min(δ_x^{DB} |$ProbEF_x \geq Prob_{pl}$)
8 EF_k^{DB} = {X | s×|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 ≥ 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, $Prob_{pl}$, 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×|UD|}
4 for {X | $\delta_x^{UD} <$ s×|UD|}
5 calculate ptt_x //using equation (14)
6 calculate probability of expected itemset X ($ProbEF_X$) //using equation (12)
7 ρ^{UD} = min(δ_x^{DB} |$ProbEF_x \geq Prob_{pl}$)
8 EF_k^{UD} = {X | s×|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×|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≥*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.

PROCEEDINGS OF THE TWENTIETH INTERNATIONAL SYMPOSIUM ON ARTIFICIAL LIFE AND ROBOTICS

(AROB 20th 2015)

AROB 20th ANNIVERSARY

January 21 – 23, 2015

B-Con Plaza, Beppu, Oita, JAPAN

Publication Date: January 21, 2015
ISBN 978-4-9907582-1-9
ISSN 2185-3797

International Society of Artificial Life and Robotics

助成　日本万国博覧会記念基金
Supported by the Japan World Exposition 1970 Commemorative Fund.
この助成金は、日本万国博覧会の収益を基にしています。
公益財団法人　関西・大阪21世紀協会

Proceedings of the Twentieth International Symposium on

ARTIFICIAL LIFE AND ROBOTICS

(AROB 20th 2015)

AROB 20th ANNIVERSARY

January 21- 23, 2015
B-Con Plaza, Beppu, Oita, Japan

International Society of Artificial Life and Robotics

Supported by the Japan World Exposition 1970 Commemorative Fund

GS19-3 *Distributed Optimization based on Networked Multi-agent Systems and Its Application to Negotiation-based Real-time Pricing*

Kazunori Sakurama, Masashi Miura (Tottori University, Japan)

GS19-4 *Graphical and Scalable Multi-Agent Simulator for Real-time Pricing in Electric Power Grid*

Masashi Miura, Yuta Tokunaga, Kazunori Sakurama (Tottori University, Japan)

GS19-5 *Hardware Implementation of Lottery Controllers for Real-time Pricing*

Taichi Kitao, Ichiro Maruta, Shun-ichi Azuma (Kyoto University, Japan)

GS19-6 *Strategy Analysis of Soccer Teams from Kick Records*

Tomoharu Nakashima, Jordan Henrio, Satoshi Mifune (Osaka Prefecture University, Japan)

GS19-7 *Simulated Human Feelings Based on Desire Driven Process*

Roungsan Chaisricharoen (Mae Fah Luang University, Thailand)

Room D

GS10 Data mining

Chair: Hidekazu Yanagimoto (Osaka Prefecture University, Japan)

GS10-1 *Finding division points for a time series corpus based on the sequential probability ratio test*

Hiroshi Kobayashi, Ryosuke Saga (Osaka Prefecture University, Japan)

GS10-2 *Developing Mobile Wallet Services acceptance model: A proposed model*

Lumyai Cotsopa, Singha Chaveesuk (King Mongkut's Institute of Technology Ladkrabang, Thailand)

GS10-3 *An Enhanced Incremental Association Rule Discovery with a lower minimum support*

Araya Ariya, Worapoj Kreesuradej (King Mongkut's Institute of Technology Ladkrabang, Thailand)

GS10-4 *A Training Support System of Brush Coating Skill with Haptic Device for Technical Education at Primary and Secondary Schools*

Shimpei Matsumoto, Masaru Teranishi, Hidetoshi Takeno (Hiroshima Institute of Technology, Japan)

GS10-5 *A proposition of judging method among the five stages of open data in local government*

Tadanori Hisanaga, Takayasu Fuchida, Ryutaro Sueyoshi, Kyoshiro Hirata (Kagoshima University, Japan)

GS10-6 *Comparative analysis of genetic based approach and Apriori algorithm for mining maximal frequent item sets (Withdrawal)*

Mir Md Jahangir Kabir, Shuxiang Xu, Byeong Ho Kang, Zongyuan Zhao

GS10-7 *LDA-Based Path Model Construction Process for Structure Equation Modeling*

Ryosuke Saga, Rikuto Kunimoto (Osaka Prefecture University, Japan)

An Enhanced Incremental Association Rule Discovery with a lower minimum support

Araya Ariya[1], Worapoj Kreesuradej[2]

[1,2]King Mongkut's Institute of Technology Ladkrabang, Thailand
(Tel: 66-02-723-4900, Fax: 66-02-723-4910)

[1]araya_aa@hotmail.com, [2]worapoj@it.kmitl.ac.th

Abstract: In the real world of data, a new set of data has been being inserted in to the existing database. Thus, the rule maintenance of association rule discovery in large databases is an important problem. Every time the new data set is appended to an original database, the old rule may probably be valid or invalid. This paper is the extension work of Mining Dynamic Databases using Probability-based Incremental Association Rule Discovery Algorithm. The idea is applying the normal approximation to the binomial for calculating the lower minimum support for collecting the expected frequent itemsets. This proposed idea can reduce a process of calculating probability value for all itemsets that unnecessary. In addition, the confidence interval is also applied to ensure that the collecting of expected frequent itemsets is properly kept.

Keywords: Data Mining, Incremental Association Rule Discovery, Expected Frequent Itemset, Bernoulli Trials, Normal Approximation to the Binomial

1 INTRODUCTION

When the new set of transactions is appended to the original database, the old rule may be obsolete and the new rule may be promote. A basic and simple method for maintaining rule is to rescan entire database with Apriori algorithm [2]. However, this method is time consuming and inefficiency because entire database is rescanned for many times depend on the maximum size of itemsets. For instance, if the maximum size of k itemset is 7, the original database is rescanned 7 times. To avoid times to rescan database, there are several algorithms are proposed such as Fast Update algorithm (FUP) [3], Negative Border (NBd) [4] Probability-based incremental association rule discovery [5] and so on. This research is the extension work of the Probability-based incremental association rule discovery.

The other parts of this paper are organized as follows. The related work of an association rule mining are reviewed in section 2. The problem statement of the incremental association rule mining and the probability-based algorithm are detailed in section 3. The normal approximation to the binomial concept for calculating the lower minimum support for collecting the expected frequent itemsets is proposed in section 4. The experimental results and the conclusion are presented in section 5 and 6 respectively.

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. 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 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) [3] 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.

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.

To deal with the collecting of massive border itemsets problem, Tsai et al. [6], Hong et al. [7], 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. 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 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 probability-based algorithm [5], we observed that the numerical difficulty in computing probabilities occurs when the algorithm deals with a large database. Furthermore, the algorithm needs to calculate the probability value for all itemsets in each k iterations. To manipulate with both problems, the idea for calculating the lower minimum support for collecting the expected frequent itemsets is introduced in this paper. The normal approximation to the binomial is the main idea to deal with the numerical problem. In addition, the confidence interval is also applied to ensure that the collecting of expected frequent itemsets is properly kept. The detail of our algorithm is proposed in section 4.

3 INCREMENTAL ASSOCIATION RULE DISCOVERY

3.1 Problem statement of the incremental association rule mining

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
δ_x^{DB}, δ_x^{UD}	a support count of itemset X in DB and UD

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. [6], 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, we focus on 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. 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:

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

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:

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

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:

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

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 AN ENHANCED INCREMENTAL ASSOCIATION RULE DISCOVERY WITH THE LOWER MININUM SUPPORT

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

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

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 [8].

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*"

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

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,*

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

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

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

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

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

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:[10] "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"

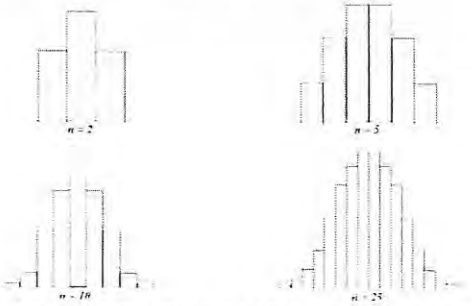


Fig. 1. Binomial distribution with p = ½ and n = 2, 5, 10 and 25

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* [10]. 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 probability density function of X is defined as

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

Applying to the proposed idea, the equation 9 and 10 are adapted to calculate the probability of occurrence itemsets. In addition, the continuity correction is also applied. 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

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

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

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

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.

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.

As mentioned to the probability-based algorithm, it needs to calculate the probability value for all itemsets in k iterations. For instance, there are 40, 100 and 250 itemsets in candidate 1-itemset, candidate 2-itemset and candidatae 3-itemset respectively. The probability-based algorithm must calculate the probability value for all 390 itemsets for finding itemset which can possibly be a frequent itemset. That degrades the algorithm efficiency. In this paper, we proposed the idea to calculate the lower minimum support, i.e. ρ^{DB} . It is initially calculated for only one time. After that, it uses for checking the support count of itemsets. Any infrequent itemset which its support count is less than $s \times |DB|$ and greater than or equal to $\rho^{DB} \times |DB|$ will be promoted to a set of expected frequent itemset. The lower minimum support , ρ^{DB} , can be obtained from the following equation:

$$P(X \geq k) = 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = \Pr ob_{PL} \quad (15)$$

where

$$\mu = n \cdot ptt_X = n(\rho^{DB} + Z_{\alpha/2}\sqrt{\frac{\rho^{DB}(1-\rho^{DB})}{n}})$$

and

$$\sigma = \sqrt{n\left(\rho^{DB} + Z_{\alpha/2}\sqrt{\frac{\rho^{DB}(1-\rho^{DB})}{n}}\right)\left(1-\left(\rho^{DB} + Z_{\alpha/2}\sqrt{\frac{\rho^{DB}(1-\rho^{DB})}{n}}\right)\right)}$$

Since eq. 15 is a nonlinear implicit function, a numerical analysis technique such as Newton–Raphson method is required to find the lower minimum support, i.e. ρ^{DB} .

Then, Any infrequent itemset which its support count is less than $s \times |DB|$ and greater than or equal $\rho^{DB} \times |DB|$ will be promoted to a set of expected frequent itemset. 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 subsection.

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 the proposed 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
$Prob_{pl}$	probability value threshold defined by user

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 calculate ρ^{DB} //using eq.(15)
3 scan DB for all X∈ C_1^{DB} and obtain δ_x^{DB}
4 F_k^{DB} ={X | $\delta_x^{DB} \geq$ s×|DB|}
5 EF_k^{DB} = {X | s×|DB| > $\delta_x^{DB} \geq \rho^{DB}$}
6 k =2
7 while $F_k^{DB} \neq \emptyset$
8 C_k = ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$) * ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$)
9 repeat line 2-5
10 k++
11 end while loop
12 Return F_k^{DB}, EF_k^{DB}, C_1^{DB}, ρ^{DB}

For the first iteration (k = 1) of original database phase as shown in algorithm 1, ρ^{DB} is first calculated. This ρ^{DB} indicates the possible minimum support count of expected frequent itemset. Next, itemsets are scanned to an original database and obtained their support count. Then, frequent itemsets and expected frequent itemset are found as line 4 and 5.

For the second iteration and beyond (k ≥ 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 8. After the candidate k-itemsets are obtained, the other steps is performed as the first iteration as line 9 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 12. These out puts are used in the next phase.

Algorithm2: Incremental Mining Phase
Input: DB, db, $Prob_{PL}$, 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 calculate ρ^{UD} //using eq.(15)
3 scan db and updating support count to obtain δ_x^{UD}
4 F_k^{UD} ={X | $\delta_x^{UD} \geq$ s×|UD|}
5 EF_k^{UD} = {X | s×|UD| > $\delta_x^{UD} \geq \rho^{UD}$}
6 end
7 k = 2
8 while $F_{k-1}^{UD} \neq \emptyset$
9 Generating Candidate itemset ( ) // call algorithm 3
10 repeat line 2-5
11 for X∈ C_k^{new} // C_k^{new} is obtained from algorithm 3
12 TempscanDB={X| (δ_x^{db} + (ρ^{DB} -1)) ≥ s×|UD|}
13 end
14 k++
15 end while loop
16 if Temp_scanDB ≠ ∅
17 Rescanning Original Database to obtain new F_k^{UD} and EF_k^{UD}
18 endif
19 clear Temp_scanDB
20 Return F_k^{UD}, EF_k^{UD}, C_1^{UD}, ρ^{UD}

The incremental mining phase, as shown in algorithm 2, there are 2 main tasks in this phase: $k=1$ iteration and $k \geq 2$ iteration. For the iteration of $k=1$, ρ^{UD} is first calculated in line 2. An increment database, db, is scanned and updated the support count of itemsets as shown in line 3. Then frequent 1-itemsets of updated database, F_k^{UD}, and the expected frequent 1-itemset, EF_k^{UD}, are found as line 4 and 5 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 9 and 17, respectively.

The generating candidate itemset is detailed in algorithm 3. 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 12 of algorithm 2 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∈ C_2^{db} |X∉ ($F_2^{DB} \cup EF_2^{DB}$)}
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 | $\delta_x^{UD} \geq$ s* |db|}
8 C_k^{db} = $F_{k-1}^{db} * F_{k-1}^{db}$
9 C_k^{new} = {X∈ C_k^{db} |X∉ ($F_k^{DB} \cup EF_k^{DB}$)}
10 end if
11 Return C_k^{db}, C_k^{new}

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}-I), where ρ^{DB} obtained from the original mining phase, is greater than or equal to s*|UD|. This process is shown in line 12 of algorithm 2 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 16-18 of algorithm 2. 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 EXPERIMENT

To evaluate the efficiency of our algorithm, we 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 40,000 transactions. A number of transactions of increment database appended to an original database is 10,000 transactions.

The experiment is conducted with 0.01% minimum support threshold and $Prob_{pf}$ = 0.2 and the confidence interval is 10%. The average of execution time compared with Apriori, FUP and probability-based algorithm are demonstrated in table 3.

Table 3. Average of Execution time for adding 10,000 transactions

Algorithm	Apriori	FUP	Probability -based	Prob-based using Normal Approximation
Execution time (sec)	449.9603	119.5068	103.7613	95.1432
a number of frequent itemset	2,121	2,121	2,121	2,121
a number of collected expected frequent itemsets	-	-	73	149
maximum frequent itemset (size of k)	F_6	F_6	F_6	F_6
a number of Temp_scanDB	-	-	41	5

The experiment results from table 3 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.

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 concept of the normal approximation to the binomial for calculating the lower minimum support for collecting the expected frequent itemsets. This proposed idea can reduce a process of calculating probability value for all itemsets that unnecessary. The experiment results show the better time when compare with other algorithms.

REFERENCES

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

[2] Agrawal R, Srikant R (1994), 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] Cheng D W, Han J, Ng V T, Wong C Y, maintenance of Discovered Association Rules in Large Databases: An incremental updating technique, In 12th IEEE International conference on Data Engineering, 1996, pp. 106-114.

[4] Toivonen H, 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] Amornchewin R, Kreesuradej W, 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] Tsai Paulry S M, lee Chin-Chong, Chen Arbee L P, an Efficient Approach for incremental assoicaiotn 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.

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

[8] Robert V. H, Elliot A.T, Probability and Statistical Inference, 8th edi, pearson Education, Inc, New jersey, 2010, pp. 78-86.

[9] Larry J.K, Exploring Statistics: A Modern introduction to Data Analysis and Inference, 2nd edi., Integre Technical Publising company, Incl; G^S Typesetters, Inc, 1998, pp.265-275.

[10] John E.F, Benjamin M.P, Statistics: A first course, 8th edi., Pearson Education, Inc, New Jersey, 2004, pp.256-260.

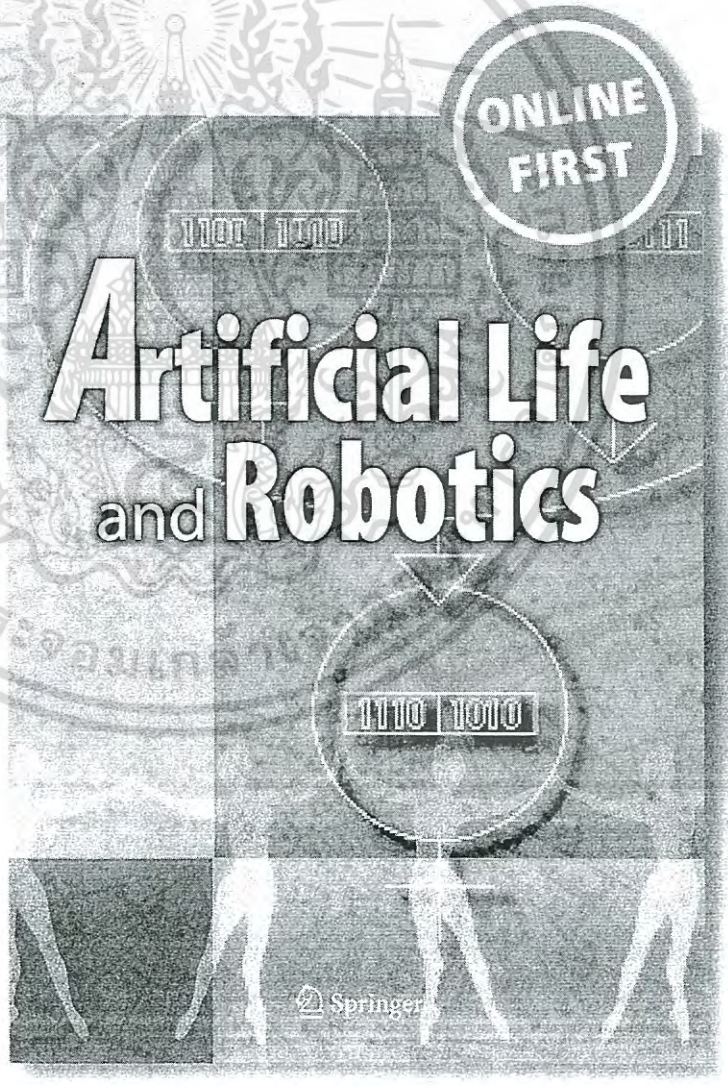
An enhanced incremental association rule discovery with a lower minimum support

Araya Ariya & Worapoj Kreesuradej

Artificial Life and Robotics

ISSN 1433-5298

Artif Life Robotics
DOI 10.1007/s10015-016-0288-3



Artif Life Robotics
DOI 10.1007/s10015-016-0288-3

ORIGINAL ARTICLE

An enhanced incremental association rule discovery with a lower minimum support

Araya Ariya[1] · **Worapoj Kreesuradej**[1]

Received: 8 April 2015 / Accepted: 22 June 2016

Abstract In the real world of data, a new set of data has been being inserted into the existing database. Thus, the rule maintenance of association rule discovery in large databases is an important problem. Every time the new data set is appended to an original database, the old rule may probably be valid or invalid. This paper proposed the approach to calculate the lower minimum support for collecting the expected frequent itemsets. The concept idea is applying the normal approximation to the binomial theory. This proposed idea can reduce a process of calculating probability value for all itemsets that are unnecessary. In addition, the confidence interval is also applied to ensure that the collection of expected frequent itemsets is properly kept.

Keywords Data mining · Incremental association rule discovery · Expected frequent itemset · Bernoulli trials · Normal approximation to the binomial

1 Introduction

When the new set of transactions is appended to the original database, the old rule may be obsolete and the new rule may be promote. A basic and simple method for maintaining rule is to rescan entire database with Apriori algorithm [2]. However, this method is time consuming and inefficient, because entire database is rescanned for more times depending on the maximum size of itemsets. For instance, if the maximum size of k itemset is 7, the original database is rescanned 7 times. To avoid times to rescan database, there are several algorithms that are proposed such as negative border (NBd) [4], pre-large [12], probability-based incremental association rule discovery [5] and so on. This research is the extension work of the probability-based incremental association rule discovery. The idea is applying the normal approximation to the binomial for calculating the lower minimum support for collecting the expected frequent itemsets. This proposed idea can reduce a process of calculating probability value for all itemsets that are unnecessary.

The other parts of this paper are organized as follows. The related works of the association rule mining are reviewed in Sect. 2. The problem statement of the incremental association rule mining and the probability-based algorithm are detailed in Sect. 3. The normal approximation to the binomial concept for estimating the lower minimum support threshold that uses to collect the expected frequent itemsets is proposed in Sect. 4. The experimental results and conclusion are presented in Sects. 5 and 6, respectively.

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. A year later, Apriori [2],

This work was presented in part at the 20th International Symposium on Artificial Life and Robotics, Beppu, Oita, January 21–23, 2015.

✉ Araya Ariya
araya_aa@hotmail.com

Worapoj Kreesuradej
worapoj@it.kmitl.ac.th

[1] Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand

Published online: 19 July 2016

the most popular algorithm of association rule mining, was proposed. After Apriori was revealed, there are many researchers that 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 set of transactions is inserted into 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 so 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.

Generally, the incremental association rule algorithm assumes that both old transaction and new transaction are equally important. According to Teng and Chen [6], the incremental association rule approach is divided into three types. There are partition-based, pattern growth and Apriori-based technique.

In partition-based technique, database is divided into n partitions. It works from the oldest partition to the newest partition. There are two thresholds assigned to find frequent itemsets. A local threshold uses to prune frequent itemsets in each partition while a global threshold uses to prune frequent itemsets for all partitions. Here are the examples of algorithm in this technique: Sliding-window filtering (SWF) [7] and FI_SWF and CI_SWF [8].

Pattern growth technique uses a tree structure to generate itemsets and to discover frequent itemsets. There are many algorithms in this technique such as DB-tree [9], TIARM [10] and Pre-FUIT [11]. DB-tree is based on FP-tree. TIARM is based on INC-Tree that extends from FP-tree. Pre-FUIT is the modification between IT-tree structure and pre-large algorithm [12].

Apriori based technique adopts Apriori concept to maintain association rule. Many proposed algorithms in this technique try to reduce times to scan original database and the number of collected infrequent itemset to be a frequent itemset. Our proposed algorithm is based on this technique. These algorithms in this technique are FUP [3], negative border [4], pre-large [12], probability-based algorithm [5] and so on.

Fast Update algorithm (FUP) [3] was proposed to maintain association rules in dynamic databases. It works using frequent itemsets from previous mining in the original database compared 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 updated 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.

Negative border algorithm (NBd) [4] was proposed to reduce the times of original databases rescanning by collecting both frequent itemsets and border itemsets. Border itemset is not a frequent itemset but its proper subsets are frequent itemset. This algorithm is successful for reducing the rescanning times but a large number of border itemsets have to be collected. Thus, this negative border consumes a large amount of memory.

To deal with the collecting of massive border itemsets problem, Tsai et al. [13], Hong et al. [12], and Amornchewin and Kreesuradej [5] presented algorithms that can reduce not only a number of collecting border itemsets but also a rescanning times. These algorithms keep both frequent itemsets and expected frequent itemsets. However, each researcher gives quite different method to select expected frequent itemsets. Tsai et al. determined a new threshold, called tolerance degree. It collaborates with the support threshold to select expected frequent itemsets. Hong et al. also assigned the two new thresholds; an upper support and a lower support threshold, to limit expect frequent itemsets. Amornchewin and Kreesuradej applied the principle of Bernoulli trials and defined a new threshold, Prob_{pl}, to predict expected frequent itemsets. Our proposed algorithm is based on this direction.

For the probability-based algorithm [5], we observed that the numerical difficulty in computing probabilities occurs when the algorithm works with the large database. Furthermore, the algorithm needs to calculate the probability value for all itemsets in every k iteration. To manipulate with both problems, the idea for calculating the lower minimum support for collecting the expected frequent itemsets is introduced in this paper. The normal approximation to the binomial is the main approach to solve that numerical problem. In addition, the confidence interval is also applied to ensure that the collecting of expected frequent itemsets is properly kept. The detail of our algorithm is proposed in Sect. 4.

3 Incremental association rule discovery

3.1 Problem statement of the incremental association rule mining

Let DB is the original database, which is a collection of old transactions, and db is the increment database, which is a collection of new transactions. Then, UD is the updated database, which is the database after merging DB and db together, i.e. $\text{UD} = \text{DB} \cup \text{db}$. The number of transactions

Table 1 The notation of 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-itemsets of DB and UD
$\delta_x^{DB}, \delta_x^{UD}$	A support count of itemset X in DB and UD

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.

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

As mentioned to Tsai et al. [13], when the original database and the 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 DB and UD.

From all cases as mentioned above, we seriously focus on the third case because it needs to rescan the original database to obtain the support count of itemsets in the updating task. Rescanning the original database requires a lot of I/O operations that wastes necessary time.

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 to predict expected frequent itemsets using the principle of Bernoulli trials. The expected frequent itemsets are collected during finding the frequent itemsets in the original database. In this subsection, the necessary concept of probability-based algorithm is briefly described.

For probability-based algorithm, the process of inserting m transactions into the 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:

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

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 in the original database.

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

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

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

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

Probability-based algorithm assumes that the statistics of old transactions obtained from previous mining can be utilized for approximating that of new transactions. Therefore, the probability-based 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 Eq. 3, i.e.p, can be approximated as the following equation:

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

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

Here, the expected frequent itemset is the itemset that is not the frequent itemset but its probability to be a frequent itemset is greater than or equal to Prob_{pl}. Prob_{pl} is a constant threshold specified by a user; it indicates the minimum confidence level that the expected frequent itemset will be the frequent itemset after inserting new transaction into the original database. The higher Prob_{pl} is set, the lesser expected frequent itemsets are kept.

For instance, given $\text{Prob}_{\text{pl}} = 0.1$ and the set of 1-infrequent itemset is B, E, F and G. According to Eq. 3, the probability values of B, E, F and G are 0.15, 0.08, 0.2 and 0.07, respectively. Hence, B and F are collected to the set of expected frequent itemset because their probability values are greater than or equal to Prob_{pl}.

According to Eq. 3, the Binomial term, i.e., $\binom{n+m}{x}$, is calculated by $\frac{(n+m)!}{x!(n+m-x)!}$. This term may suffer from a numerical problem when the factorial is computed with a large value. This is the case because the factorial of a large value cannot fit to the size of the integer variable. For a long integer data type, which occupies 64 bits, can only store 20!. The computer must return a null value, when the output of the Binomial term is more than 20!. For example, let n and m be the size of the original database and the increment database, respectively. Given $n = 500$ and $m = 100$. The minimum support count after inserting db into DB, i.e.β, is 80. Then the Binomial term calculated by Eq. 3 returns $\frac{(500+100)!}{80!(500+100-80)!}$ = null. Consequently, the output of $P(x \geq \beta)_{\text{item}}$ from Eq. 3 must be a null value either.

In the association rule mining, it is common to deal with computing massive transactions. Therefore, the long integer type does not have enough memory to store the factorial result. As a result, some probabilities of itemsets cannot be computed directly by Eq. 3. In the probability-based algorithm, these probabilities are approximated by a linear extrapolation to avoid that problem. This gives some errors of the probability values. In addition, the probability-based algorithm needs to compute the probability value for all itemsets so as to find expected frequent itemsets. This reduces the efficiency of the algorithm.

To deal with these problems, we proposed the idea to use the normal approximation to binomial to solve both problems as mentioned above. For the first problem, our proposed approach can solve the numerical problem. The second problem, a new minimum support threshold, called the lower minimum support threshold, is derived from the normal approximation to the binomial concept. With this new threshold, the expected frequent itemset can be found without computing the probability value for all itemsets.

4 An enhanced incremental association rule discovery with the lower minimum support

According to the probability-based algorithm, there are two main problems of this algorithm: the factorial numerical problem and multiple times for calculating the probability value. This section, the concept idea and approach to deal with both problems are illustrated in three subsections. First, the normal approximation to the binomial theory is explained. Second, the approach to estimate the lower minimum support is shown. Finally, the algorithm work is detailed.

4.1 Normal approximation to the binomial

The Bernoulli trial 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

$$f(x) = p^x(1-p)^{1-x}, \quad \begin{matrix} x = 0 \text{ (Failure)} \\ x = 1 \text{ (Success)}. \end{matrix} \tag{5}$$

Generally, the 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 [15].

Definition 1 [14] "Let X be the number of observed success in n Bernoulli trials, then the possible values of X are $0, 1, 2, \ldots, n$. If x successes occur, where $x = 0, 1, 2, \ldots, n$, then $n - x$ failures occurs The number of ways of selecting x positions for the x success in the n trials is"

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

Definition 2 [14] "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,

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

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

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

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

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

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

Referred to the probability-based algorithm, the expected frequent itemset is predicted by applying Eq. 9. In practice, the binomial probability has a numerical problem when factorial

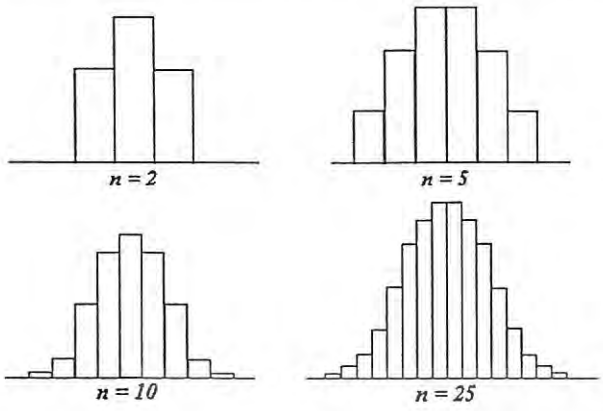


Fig. 1 Binomial distribution with $p = ½$ and $n = 2$, 5, 10 and 25

is computed with the large n. That is the factorial value of the large n cannot fit to the size of the integer variable.

Definition 3 [16] "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".

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 $n(1-p) > 5$ [10].

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

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

The normal can approximate to the binomial well when $np > 5$ and $n(1-p) > 5$. However, our proposed algorithm focuses on infrequent itemsets that can potentially be the frequent itemset. These infrequent itemsets satisfy $np > 5$ and $n(1-p) > 5$. Generally, the initial original database size is conducted at least 10,000 transactions. Let n and p be the original database size and the minimum support threshold. Given $n = 10,000$ and $p = 0.001$. Thus $np = 10,000 \times 0.001 = 10$ and $n(1-p) = 10,000 \times 0.999 = 9990$. In fact, mining data in real world is operated with massive data. Accordingly, np and $n(1-p)$ are absolutely more than 5. Thus, the normal approximation to the binomial can be applied to our algorithm.

According to Eqs. 9 and 10, these two equations are adapted to calculate the probability of occurrence itemset. In addition, the continuity correction is also modulated. Using the continuity correction, the number of successes, i.e. β, is decreased by 0.5. Accordingly, the probability success k times can be derived as the following equation

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

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

$$P(X \ge \beta) = 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx, \tag{12}$$

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.

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 the statistical confidence interval.

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

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

According to Eq. 13, the output has two thresholds: 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

$$\text{ptt}_X = p + Z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}}, \tag{14}$$

where $Z_{\alpha/2}$ is the confidence interval determined by user. Here, ptt_X in Eq. 14 is used in Eq. 12 instead of p.

4.2 Estimating the lower minimum support threshold

According to the previous section, the probability of the itemset, i.e., $P(X \ge \beta)$, to be the frequent itemset in the updated database can be derived as the following equation:

$$P(X \ge \beta) = 1 - \int_{-0.5}^{k-0.5} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = \text{Prob}_{\text{PI}}, \tag{15}$$

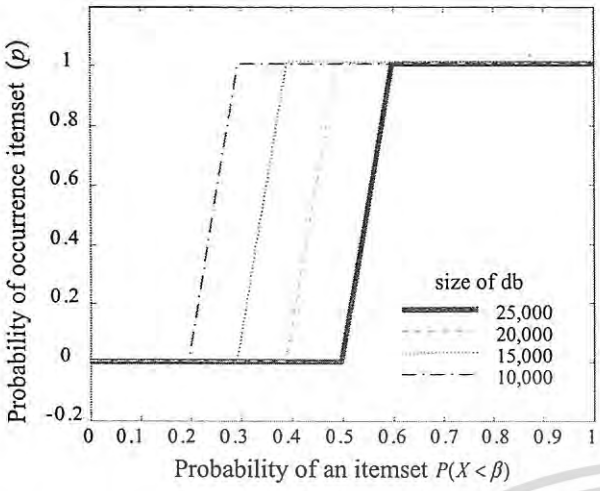


Fig. 2 The relationship between the probability of an itemset, i.e., $P(X < \beta)$, and the probability of occurrence itemset i.e., ρ

where

$$\mu = n \cdot \mathrm{ptt}_X = n\left(\rho^{\mathrm{DB}} + Z_{\alpha/2}\sqrt{\frac{\rho^{\mathrm{DB}}(1-\rho^{\mathrm{DB}})}{n}}\right),$$

and

$$\sigma = \sqrt{n\left(\rho^{\mathrm{DB}} + Z_{\alpha/2}\sqrt{\frac{\rho^{\mathrm{DB}}(1-\rho^{\mathrm{DB}})}{n}}\right)\left(1 - \left(\rho^{\mathrm{DB}} + Z_{\alpha/2}\sqrt{\frac{\rho^{\mathrm{DB}}(1-\rho^{\mathrm{DB}})}{n}}\right)\right)}.$$

For the probability-based algorithm, it needs to calculate the probability value for all itemsets in k iterations. For instance, there are 40, 100 and 250 itemsets in candidate 1-itemset, candidate 2-itemset and candidate 3-itemset, respectively. The probability-based algorithm has to calculate the probability value for all 390 itemsets for finding expected frequent itemsets. That degrades the algorithm efficiency.

With reference to Eq. 15, given the user specifies the probability of an itemset to be the frequent itemset in the updated database, i.e., $P(X \geq \beta) = \mathrm{Prob}_{\mathrm{pl}}$. The probability of occurrence for the itemset, i.e. p, can be found using the numerical method such as Newton–Raphson method. This is the case because Eq. 15 is a non-linear implicit function.

The probability of occurrence for the itemset, i.e., p, indicates that the itemset is the expected frequent itemset because its probability to be a frequent itemset in the updated database is equal to $\mathrm{Prob}_{\mathrm{pl}}$. In addition, the relationship between $P(X \geq \beta)$ and p in Eq. 15 is monotonically increasing as shown in Fig. 2.

Therefore, if $p_{x_1} \geq p_{x_2}$ then $P(X_1 \geq \beta) \geq P(X2 \geq \beta)$. Thus, if the user specifies the probability of an itemset to be a frequent itemset in the updated database, i.e. $P(X \geq \beta) = \mathrm{Prob}_{\mathrm{pl}}$, then the computed probability of occurrence for the itemset can be found using the numerical methods, i.e. $p = \rho^{\mathrm{DB}}$. Next, all infrequent itemsets are classified as expected frequent itemset if their probability of occurrence itemsets or the support factor, i.e., p, are greater than or equal to ρ^{DB} but less than the minimum support threshold. This is the case because the relationship between $P(X < \beta)$ and p in Eq. 15 is monotonically increasing.

Here, the computed probability of occurrence for an itemset, i.e., ρ^{DB}, is called the lower minimum support threshold because this value is always less than a true minimum support threshold, i.e., $s \times |\mathrm{DB}|$. For this reason, any infrequent itemset that its support count is less than $s \times |\mathrm{DB}|$ but greater than or equal to ρ^{DB} will be promoted to a set of expected frequent itemset. This can avoid calculating the probability value for all itemsets in every iteration. Therefore, it can improve the algorithm efficiency.

Like Tsai et al. [13], Hong et al. [12] and, Amornchewin and Kreesuradej [5], both frequent itemsets and expected frequent itemsets are kept to reduce the times to rescann the original database. Our proposed algorithm is presented in the next subsection.

4.3 An incremental association rule discovery with a lower minimum support algorithm

The proposed algorithm, an incremental association rule discovery with a lower minimum support algorithm, is based on the probability-based incremental association rule discovery algorithm. While the probability-based algorithm uses the binomial trial, our proposed algorithm uses the normal approximation to the binomial. There are 2 main phases: original mining phase and incremental mining phase. The notation used in this 2 main phases is defined in Table 2.

For the first iteration ($k = 1$) of the original database phase as shown in algorithm 1,ρ^{DB} is first calculated. This ρ^{DB} indicates the possible minimum support count of expected frequent itemset. Next, itemsets are scanned to the original database to obtain their support count. Then, frequent itemsets and expected frequent itemsets are found as line 4 and 5.

Table 2 The notation of the proposed 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
$\delta_x^{DB}, \delta_x^{db}, \delta_x^{UD}$	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
$Prob_{pl}$	Probability value threshold defined by user

For the second iteration and beyond ($k \geq 2$), the apriori_gen concept, which proposed in Agrawal [2], is used to generate candidate k-itemset. Unlike apriori_gen, the candidate k-itemset of the proposed algorithm is generated by $(F_{k-1}^{DB} \cup EF_{k-1}^{DB}) \times (F_{k-1}^{DB} \cup EF_{k-1}^{DB})$ as line 8. After the candidate k-itemset is obtained, the other steps are performed as the first iteration as line 9 and $C_{k\geq 2}$ does not collect. These steps of $k \geq 2$ 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 12. These outputs are used in the next phase.

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 calculate ρ^{DB} //using eq.(15)
3 scan DB for all X∈ C_1^{DB} and obtain δ_x^{DB}
4 F_k^{DB} ={X | $\delta_x^{DB} \geq s\times|DB|$}
5 EF_k^{DB} = {X | $s\times|DB| > \delta_x^{DB} \geq \rho^{DB}$}
6 k =2
7 while $F_k^{DB} \neq \emptyset$
8 C_k = ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$) * ($F_{k-1}^{DB} \cup EF_{k-1}^{DB}$)
9 repeat line 2-5
10 k++
11 end while loop
12 Return F_k^{DB}, EF_k^{DB}, C_1^{DB}, ρ^{DB}

The incremental mining phase, as shown in algorithm 2, has 2 main tasks: $k = 1$ iteration and $k \geq 2$ iteration. For the iteration $k = 1$, ρ^{UD} is first calculated in line 2. The increment database, db, is scanned and updated the support count of itemsets as shown in line 3. Then frequent 1-itemsets of the updated database, F_k^{UD}, and the expected frequent 1-itemset, EF_k^{UD}, are found as line 4 and 5, respectively.

For the iteration $k \geq 2$, 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 9 and 17, respectively.

The generating candidate itemset is shown in algorithm 3. Candidate 2-itemset generating and candidate $k \geq 2$ itemset generating are different. For candidate 2-itemset, it is generated by $F_1^{UD} \times F_1^{UD}$ as line 2. Candidate $k \geq 2$ itemset is generated by $F_{k-1}^{db} \times F_{k-1}^{db}$ as line 8 where F_{k-1}^{db} is obtained from itemset that its support count is greater than or equal to $s \times |db|$. To ensure that no candidate itemsets are lost, itemset that 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. These 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 12 of algorithm 2 to extract Temp_scanDB .

Algorithm2: Incremental Mining Phase
Input: DB, db, $Prob_{PL}$, 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 calculate ρ^{UD} //using eq.(15)
3 scan db and updating support count to obtain δ_x^{UD}
4 F_k^{UD} ={X | $\delta_x^{UD} \geq s\times|UD|$}
5 EF_k^{UD} = {X | $s\times|UD| > \delta_x^{UD} \geq \rho^{UD}$}
6 end
7 k = 2
8 while $F_{k-1}^{UD} \neq \emptyset$
9 Generating Candidate itemset ( ) // call algorithm 3
10 repeat line 2-5
11 for X∈ C_k^{new} // C_k^{new} is obtained from algorithm3
12 TempscanDB={X|(δ_x^{db} +(ρ^{DB} -1))≥ $s\times|UD|$}
13 end
14 k++
15 end while loop
16 if Temp_scanDB ≠ ∅
17 Rescanning Original Database to obtain new F_k^{UD} and EF_k^{UD}
18 endif
19 clear Temp_scanDB
20 Return F_k^{UD}, EF_k^{UD}, C_1^{UD}, ρ^{UD}

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 $\rho^{DB} - 1$, where ρ^{DB} obtained from the original mining phase, is greater than or equal to $s \times |UD|$. This process is shown in line 12 of algorithm 2 for pruning C_k^{new}, which impossibly be a frequent itemset or an expected frequent itemset of the updated database.

Table 3 Execution time for adding 1000 transactions for T15I10D100K dataset

Sup (%)	Algorithm	Execution time (s)	Maximum frequent itemset (size of k)	Number of		
				Frequent itemset	Expected frequent itemset	TempscanDB
3	Apriori	2194.3887	12	20,265	–	–
	FUP	216.2582	12	20,265	–	–
	Negative border	186.5962	12	20,265	28,383	–
	Pre-large	301.3411	12	20,265	10,745	5530
	Prob-based	137.1251	12	20,265	725	6
	Proposed algorithm	126.6239	12	20,265	1131	–
5	Apriori	853.2071	12	9402	–	–
	FUP	48.8817	12	9402	–	–
	Negative border	39.1972	12	9402	10,102	–
	Pre-large	77.9665	12	9402	1157	3516
	Prob-based	39.0021	12	9402	88	10
	Proposed algorithm	37.9976	12	9402	131	–
7	Apriori	399.3395	6	3629	–	–
	FUP	20.8542	6	3629	–	–
	Negative border	16.4778	6	3629	6119	–
	Pre-large	59.2013	6	3629	2354	3301
	Prob-based	15.1268	6	3629	487	14
	Proposed algorithm	13.6357	6	3629	737	2
10	Apriori	125.6599	4	727	–	–
	FUP	3.8167	4	727	–	–
	Negative border	4.7960	4	727	2571	–
	Pre-large	19.4071	4	727	212	3114
	Prob-based	3.1992	4	727	6	18
	Proposed algorithm	2.8975	4	727	8	3

For the final step of an incremental mining phase, Temp_scanDB is rescanned to the original database to update the support count and find new frequent itemset and expected frequent itemset of the updated database as shown in line 16–18 of algorithm 2. 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 find absolutely.

5 Experiment

To evaluate the efficiency of our algorithm, we 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 and a real dataset.

For the synthetic dataset, it is generated from synthetic technique proposed by Agrawal [1]. The synthetic dataset is T15I10D100K with the original database of 100,000 transactions. A number of transactions of increment database appended to an original database are 1, 5, 10 and 25 %, respectively. Thus, the increment sizes are 1, 5, 10 and 25 K transactions. The experiments are conducted with four minimum support thresholds: 3, 5, 7 and 10 %. The confidence interval is 10 % and $Prob_{pl} = 0.1$.

The real dataset is Tafeng groceries shopping dataset [17]. The original database is also set to 100,000 transactions. An increment database size is 1 % or 1000 transactions. The experiment is conducted with 2 minimum support thresholds: 1 and 3 %.

The execution time is compared to six algorithms: Apriori, FUP, negative border, pre-large, probability-based and the proposed algorithm. The experimental results of synthetic dataset are demonstrated in Tables 3, 4, 5, 6. Table 7 shows the result for Tafeng dataset. Like other works [1–13], this paper does not mention about generating rules because if all frequent itemsets of the algorithms and those obtained from Apriori algorithm are the same, discovered association rules are the same too.

Table 4 Execution time for adding 5000 transactions for T15I10D100K dataset

Sup (%)	Algorithm	Execution time (s)	Maximum frequent itemset (size of k)	Number of		
				Frequent itemset	Expected frequent itemset	TempscanDB
3	Apriori	2304.9454	12	21,743	–	–
	FUP	250.6498	12	21,743	–	–
	Negative border	238.0776	12	21,743	28,383	–
	Pre-large	334.7520	12	21,743	10,745	5521
	Prob-based	231.4976	12	21,743	719	6
	Proposed algorithm	211.8430	12	21,743	1109	–
5	Apriori	808.5289	12	8899	–	–
	FUP	60.4293	12	8899	–	–
	Negative border	63.1051	12	8899	10,102	–
	Pre-large	92.3061	12	8899	1157	3402
	Prob-based	59.1415	12	8899	81	7
	Proposed algorithm	57.2944	12	8899	128	–
7	Apriori	344.2176	6	2818	–	–
	FUP	25.0324	6	2818	–	–
	Negative border	27.4875	6	2818	6119	–
	Pre-large	70.4088	6	2818	2354	3265
	Prob-based	24.5601	6	2818	481	16
	Proposed algorithm	23.9756	6	2818	730	2
10	Apriori	121.8540	4	684	–	–
	FUP	8.2637	4	684	–	–
	Negative border	9.9996	4	684	2571	–
	Pre-large	26.2128	4	684	212	3231
	Prob-based	8.0117	4	684	6	14
	Proposed algorithm	7.6303	4	684	8	4

To verify the output accuracy, we compare with Apriori algorithm. At the fifth column of all result tables illustrate that the number of frequent itemset of Apriori and the proposed algorithm are the same. The experiment results of the proposed algorithm are the same as Apriori. In other words, no frequent itemsets are lost.

The experiment results in Tables 3, 4, 5, 6, 7 shows that the execution time of the proposed algorithm is better than all compared algorithms. Apriori and FUP need k times to rescan the original database, while the proposed algorithm needs once. Consequently, the proposed algorithm is faster than these two algorithms. For Negative border, Pre-large and Prob-based algorithm, the proposed algorithm is faster than these algorithms because of lesser expected frequent itemsets and Temp_scanDB.

Focusing on the number of collected expected frequent itemset, the sixth column of all result tables, the proposed algorithm has expected frequent itemsets less than negative border and pre-large. This means that the proposed algorithm keeps lesser itemsets but sufficient to maintain frequent itemsets in the updated database.

Even though the proposed algorithm has expected frequent itemsets more than Prob-based algorithm, the execution time of the proposed is less than Prob-based algorithm. There are two reasons why the proposed algorithm has a better execution time. First, Prob-based calculates the probability for all itemsets in all k iterations, while the proposed algorithm calculates once. Second, Prob-based collects less expected frequent itemset. Therefore, they do not cover some expected frequent itemset. This affect to the Temp_scanDB, which is more than the proposed algorithm. Because of both reasons, the proposed algorithm can reduce the execution time.

6 Conclusion

In this paper, we proposed an incremental association rule discovery with a lower minimum support algorithm for collecting the expected frequent itemsets. This proposed idea can reduce the process of calculating the probability value for all itemsets that are unnecessary. The experiment results show the better time when compared with other algorithms.

Table 5 Execution time for adding 10,000 transactions for T15I10D100K dataset

Sup (%)	Algorithm	Execution time (s)	Maximum frequent itemset (size of k)	Number of		
				Frequent itemset	Expected frequent itemset	TempscanDB
3	Apriori	2194.4013	12	20,268	–	–
	FUP	297.2837	12	20,268	–	–
	Negative border	331.5162	12	20,268	28,383	–
	Pre-large	397.2434	12	20,268	10,745	5504
	Prob-based	281.1498	12	20,268	714	14
	Proposed algorithm	264.0748	12	20,268	1069	1
5	Apriori	771.4682	12	8424	–	–
	FUP	80.6114	12	8424	–	–
	Negative border	98.2398	12	8424	10,102	–
	Pre-large	113.3607	12	8424	1157	3420
	Prob-based	75.9631	12	8424	79	13
	Proposed algorithm	72.8610	12	8424	126	1
7	Apriori	312.3503	5	2517	–	–
	FUP	36.2310	5	2517	–	–
	Negative border	46.0290	5	2517	6119	–
	Pre-large	87.9960	5	2517	2354	3251
	Prob-based	34.9677	5	2517	477	22
	Proposed algorithm	31.7676	5	2517	725	2
10	Apriori	117.5186	4	622	–	–
	FUP	11.3947	4	622	–	–
	Negative border	17.1617	4	622	2571	–
	Pre-large	39.9790	4	622	212	3235
	Prob-based	10.8825	4	622	6	30
	Proposed algorithm	9.2315	4	622	8	4

Table 6 Execution time for adding 25,000 transactions for T15I10D100K dataset

Sup (%)	Algorithm	Execution time (s)	Maximum frequent itemset (size of k)	Number of		
				Frequent itemset	Expected frequent itemset	TempscanDB
3	Apriori	2032.5613	12	15,954	–	–
	FUP	415.4250	12	15,954	–	–
	Negative border	607.9784	12	15,954	28,383	–
	Pre-large	651.6698	12	15,954	10,745	5701
	Prob-based	388.5814	12	15,954	702	20
	Proposed algorithm	369.6878	12	15,954	1001	2
5	Apriori	699.0446	11	6646	–	–
	FUP	140.9498	11	6646	–	–
	Negative border	212.3472	11	6646	10,102	–
	Pre-large	224.3151	11	6646	1157	3616
	Prob-based	135.2121	11	6646	74	24
	Proposed algorithm	130.3802	11	6646	121	2
7	Apriori	202.7285	4	1091	–	–
	FUP	48.713	4	1091	–	–
	Negative border	101.9907	4	1091	6119	–
	Pre-large	105.7476	4	1091	2354	3314
	Prob-based	44.6443	4	1091	474	35
	Proposed algorithm	40.0661	4	1091	712	2

Table 6 continued

Sup (%)	Algorithm	Execution time (s)	Maximum frequent itemset (size of k)	Number of		
				Frequent itemset	Expected frequent itemset	TempscanDB
10	Apriori	100.6985	3	346	–	–
	FUP	18.5784	3	346	–	–
	Negative border	38.0645	3	346	2571	–
	Pre-large	55.6431	3	346	212	3248
	Prob-based	17.1964	3	346	6	51
	Proposed algorithm	16.3051	3	346	8	4

Table 7 Execution time for adding 1000 transactions for Tafeng dataset

Sup (%)	Algorithm	Execution time (s)	Maximum frequent itemset (size of k)	Number of		
				Frequent itemset	Expected frequent itemset	TempscanDB
3	Apriori	71.6480	6	1523	–	–
	FUP	15.4360	6	1523	–	–
	Negative border	15.7790	6	1523	1721	–
	Pre-large	24.7865	6	1523	75	1036
	Prob-based	14.8546	6	1523	29	2
	Proposed algorithm	11.9936	6	1523	35	–
5	Apriori	19.0668	5	235	–	–
	FUP	4.5855	5	235	–	–
	Negative border	5.1739	5	235	1218	–
	Pre-large	6.0871	5	235	34	522
	Prob-based	3.6300	5	235	5	2
	Proposed algorithm	2.0553	5	235	7	–
7	Apriori	11.9572	4	83	–	–
	FUP	4.5237	4	83	–	–
	Negative border	4.7326	4	83	1164	–
	Pre-large	4.9505	4	83	33	483
	Prob-based	1.9558	4	83	2	–
	Proposed algorithm	1.1335	4	83	2	–
10	Apriori	8.4296	2	28	–	–
	FUP	2.6286	2	28	–	–
	Negative border	3.1750	2	28	1132	–
	Pre-large	3.8836	2	28	12	434
	Prob-based	1.1435	2	28	2	–
	Proposed algorithm	0.9276	2	28	2	–

Nevertheless, the common limitation of the proposed algorithm is the fixed size of the increment database. This means that the proposed algorithm needs to know the exact 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. However, the proposed algorithm runs as a batch processing. Thus, we can partition a new database into several increments that have the assumed size. Then, each increment can be inserted separately into an original database. The future research is looking for the approach to deal with the fixed increment database size problem.

References

1. Agrawal R, Imielinski T, Swanmi A (1993) A mining association rules between sets of items in large databases. In: Proceeding of the ACM SIGMOD Int'l Conference on Management of Data (ACM SIGMOD'93), Washington, USA, May 1993, pp 207–216
2. Agrawal R, Srikant R (1994) Fast algorithm for mining association rules. In: Proceedings 20th International Conference on Very large Databases (VLDB'94), Santiago, Chile, September 12–15, 1994, pp 487–499
3. Cheng DW, Han J, Ng VT, Wong CY Maintenance of Discovered association rules in large databases: an incremental updating technique. In: 12th IEEE International Conference on Data Engineering, 1996, pp 106–114
4. Toivonen H, 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. Amornchewin R, Kreesuradej W (2009) Mining dynamic databases using probability-based incremental association rule discovery algorithm. J Univers Comput Sci 15(12):2409–2428
6. Teng WG, Chen MS (2005) Incremental mining on association rules, vol 180. Springer, Berlin, pp 125–162
7. Lee CH, Lin CR, Chen MS (2001) Sliding-window filtering: an efficient algorithm for incremental mining. In: Proceeding of the ACM 10th International Conference on Information and Knowledge Management, November 2001
8. Chang CH, Yang SH Enhancing SWF for incremental association mining by itemset maintenance. In: Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining, April 2003
9. Ezeife EI, Su Y Mining incremental association rules with generalized FP-tree. In: Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence, May 2002, pp 147–160
10. Pradeepini G, Jyothi S Tree-based incremental association rule mining without candidate itemset generation. In: 2nd International Conference on Trendz in Information Sciences and Computing (TISC2010), December 17–19, 2010, pp 78–81
11. TP Le, TP Hong, B Vo, B Le An Efficient incremental mining approach based on IT-tree. In: 2012 IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future, February 27–March 1, 2012
12. Hong TP, Wang CY, Tao YH (2001) A new incremental data mining algorithm using pre-large itemsets. J Intell Data Anal 5(2):111–129
13. Tsai PSM, Lee CC, Chen ALP An efficient approach for incremental association rule mining. In: Proceedings of the third Pacific-Asia Conference on methodologies for Knowledge discovery and Data Mining, Lecture notes in computer Science, Vol. 1574 archive, 1999
14. Robert VH, Elliot AT (2010) Probability and statistical inference, 8th edn. Pearson Education Inc, New Jersey, pp 78–86
15. Larry JK (1998) Exploring statistics: a modern introduction to data analysis and inference, 2nd edn. Cengage Learning, US, pp 265–275
16. John EF, Benjamin MP (2004) Statistics: a first course, 8th edn. Pearson Education Inc, New Jersey, pp 256–260
17. Tafeng groceries shopping dataset. http://recsyswiki.com/wiki/Grocery_shopping_datasets. Accessed 15 May 2016

ภาคผนวก ค

ประวัติผู้เขียน

ประวัติผู้เขียน

ชื่อ – นามสกุล นางสาวอารยา อริยา

วันเดือนปีเกิด 12 เมษายน 2520

ที่อยู่ 83 หมู่ 1 ต.ริมกก อ.เมือง จ.เชียงราย 57100

ประวัติการศึกษา

2550 วิทยาศาสตรมหาบัณฑิต (เทคโนโลยีสารสนเทศ) มหาวิทยาลัยนเรศวร จังหวัดพิษณุโลก

2548 วิทยาศาสตรมหาบัณฑิต (เศรษฐศาสตร์เกษตร) มหาวิทยาลัยแม่โจ้ จังหวัดเชียงใหม่

2542 บริหารธุรกิจบัณฑิต (คอมพิวเตอร์ธุรกิจ) วิทยาลัยโยนก จังหวัดลำปาง

ประวัติการทำงาน

2543 – 2546 อาจารย์สอนคอมพิวเตอร์ ศูนย์คอมพิวเตอร์ท็อตส์ จังหวัดเชียงใหม่

2546 – 2548 อาจารย์ประจำสาขาวิชาคอมพิวเตอร์ธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยราชภัฏสวนดุสิต วิทยาเขตพิษณุโลก

2548 – 2549 อาจารย์ประจำสาขาวิชาเศรษฐศาสตร์ธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยราชภัฏนครสวรรค์

2549 – ปัจจุบัน อาจารย์ประจำสาขาวิชาเศรษฐศาสตร์ธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยราชภัฏลำปาง

ผลงานวิจัย

- Araya Ariya, Worapoj Kreesuradej (2015). **An Enhanced Incremental Association Rule Discovery with a lower minimum support**. The 20th International Symposium on Artificial Life and Robotics 2015 (AROB 20th '15). B-Con Plaza, Beppu, Oita, Japan, January 21 – 23, 2015, pp. 192 – 197.

- Araya Ariya, Worapoj Kreesuradej (2013). **Probability-Based Incremental Association Rule Discovery Using The Normal Approximation**. Proceedings of the 2013 IEEE 14th International Conference on Information Reuse and Integration (IEEE IRI 2013). San Francisco, California, USA, August 14 – 16, 2013, pp. 432 – 439.

- Araya Ariya, Worapoj Kreesuradej (2012). **Batch Fast Update Algorithm for Incremental Association Rule Discovery**. The 17th International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12). B-Con Plaza, Beppu, Oita, Japan, January 19 – 20, 2012, pp. 799 – 802.

- Araya Ariya, Worapoj Kreesuradej. **An Enhanced Incremental Association Rule Discovery with a lower minimum support**. (ได้รับการตอบรับและรอการตีพิมพ์ในวารสาร Artificial Life and Robotics หมายเลข DOI: 10.1007/s10015-016-0288-3)