บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในยุคเศรษฐกิจดิจิทัล ระบบแนะนำสินค้า (Recommender Systems) ได้กลายเป็นองค์ประกอบหลักที่ขาดไม่ได้สำหรับธุรกิจแพลตฟอร์มในทุกอุตสาหกรรม ทั้งแพลตฟอร์มตลาดขายของออนไลน์ (E-Commerce) ไปจนถึงการบริการสตรีมมิ่ง (Streaming) เช่น ภาพยนต์และเพลง (Lu et al., 2015) ระบบเหล่านี้มีบทบาทสำคัญในการช่วยผู้ใช้รับมือกับการที่ผู้ใช้ต้องเห็นข้อมูลมากเกินไป (Information Overload) (Roy et al., 2022) โดยนำเสนอเฉพาะสิ่งที่ผู้ใช้อาจจะสนใจเท่านั้น

ผลลัพธ์จากการใช้ระบบแนะนำสินค้า มีการพิสูจน์กันอย่างกว้างขวางว่าสามารถช่วยสร้างประโยชน์ได้ทั้งต่อผู้ใช้และองค์กร ประโยชน์หลักที่เห็นได้ชัดคือการเพิ่มยอดขาย (Sale Revenue) และการมีส่วนร่วมของผู้ใช้ (Engagement) ระบบเหล่านี้ทำงานโดยการประเมินพฤติกรรมและความสนใจของผู้ใช้ในอดีตเพื่อนำเสนอคำแนะนำที่เป็นส่วนตัว (Personalized Recommendation) (Rao et al., 2024)

ระบบแนะนำสินค้ามีหลายรูปแบบ โดยสามารถจำแนกได้เป็น 4 ประเภท (Aggarwal, 2016) คือ

1. Knowledge-Based: เป็นเทคนิคที่ใช้ข้อกำหนดของผู้ใช้ร่วมกับความรู้ในโดเมน (Domain Knowledge) เช่น กฎเกณฑ์ต่างๆ ในการแนะนำสินค้า โดยไม่ต้องอาศัยข้อมูลประวัติการให้คะแนน
2. Content-Based Filtering (CBF): เป็นเทคนิคที่แนะนำสินค้าโดยอาศัยการวิเคราะห์ข้อมูลคุณลักษณะ (Features) ของตัวสินค้าเอง เทคนิคนี้จะพยายามค้นหาสินค้าที่มีคุณลักษณะคล้ายคลึงกับสินค้าที่ผู้ใช้เคยชื่นชอบหรือมีปฏิสัมพันธ์ด้วยในอดีต (เช่น แนะนำภาพยนต์แนว Sci-Fi เพราะผู้ใช้เคยดูภาพยนต์แนว Sci-Fi หลายเรื่อง) แต่จำเป็นต้องใช้ข้อมูลคุณลักษณะ (Feature) ของสินค้าและผู้ใช้เพิ่มเติม
3. Collaborative Filtering (CF): เป็นเทคนิคที่แนะนำสินค้าโดยอาศัยการให้คะแนน (Ratings) หรือการโต้ตอบ (Interaction) ของผู้ใช้กลุ่มใหญ่ เทคนิคนี้ใช้สมมติฐานที่ว่าใช้ที่มีความชอบคล้ายคลึงกันในอดีต (เช่น ให้คะแนนภาพยนต์เรื่องต่างๆ คล้ายกัน) จะมีความชอบคล้ายคลึงกันในอนาคต

4. Hybrid Methods: คือการนำแนวคิดของ CF และ CBF (หรือแนวคิดอื่น) มาผสมผสานกัน เพื่อดึงจุดแข็งของแต่ละฝ่ายมาใช้งานและกลบจุดอ่อนของกันและกัน ซึ่งโดยทั่วไปจะช่วยเพิ่มประสิทธิภาพและความแม่นยำของระบบโดยรวม

ในบรรดาเทคนิคการแนะนำที่หลากหลาย CF โดยเฉพาะอย่างยิ่ง Matrix Factorization (MF) ได้รับการยอมรับอย่างกว้างขวางว่าเป็นหนึ่งในโมเดลที่มีประสิทธิภาพสูง เนื่องจากความสามารถในการเรียนรู้ปัจจัยแฝง (Latent Factors) ของผู้ใช้และสินค้า (Aggarwal, 2016) ต่อมาได้มีการพัฒนาโมเดลโดยการเพิ่มความเอนเอียง (Bias) เข้าไปในโมเดล (Bias MF) ซึ่งเพิ่มความแม่นยำในการทำนายได้ดียิ่งขึ้นไปอีก โมเดล Bias MF ได้รับความสนใจและโด่งดังอย่างมากจากการแข่งขัน Netflix Prize (Koren et al., 2009) แต่ในการประยุกต์ใช้งานจริง โมเดลกลุ่ม MF และ Bias MF ต้องเผชิญกับข้อจำกัดสำคัญ 2 ประการที่ส่งผลต่อความแม่นยำ ดังนี้

1. ปัญหาความเบาบางของข้อมูล (Data Sparsity): คือ สภาวะที่ระบบมีข้อมูลการให้คะแนน (Rating) หรือการโต้ตอบ (Interaction) น้อยมาก เมื่อเทียบกับปริมาณผู้ใช้และสินค้าทั้งหมด อาทิ ชุดข้อมูล MovieLens ซึ่งเป็นชุดข้อมูลมาตรฐานในงานวิจัยหลากหลายงาน พบว่ามีการให้คะแนนเกิดขึ้นจริงเพียงแค่ 1% (Roy et al., 2022) ส่วนที่เหลือคือช่องว่างที่โมเดลไม่รู้ข้อมูล
2. ปัญหาการเริ่มต้นระบบ (Cold Start) ปัญหา Cold Start เกิดจากการที่มีผู้ใช้ใหม่เข้ามา แต่ยังไม่เคยให้คะแนนสินค้าใดๆ (User Cold Start) หรือมีสินค้าใหม่เข้ามา แต่ยังไม่มีใครเคยกดให้คะแนน (Item Cold Start)

เนื่องจากโมเดล MF และ Bias MF จำเป็นต้องใช้คะแนนที่เกิดขึ้นจริง (Existing Ratings) ในการเรียนรู้ เมื่อเกิดปัญหา Cold Start หรือ Sparsity จึงส่งผลให้ความแม่นยำแย่ลง ดังนั้นเพื่อแก้ไขปัญหา Cold Start แนวทางที่นักวิจัยนิยมใช้กันมากที่สุดคือการพัฒนาโมเดลแบบผสม (Hybrid Methods) ซึ่งมักจะใช้วิธีการดึงเอาข้อมูลคุณลักษณะ (Feature) ของสินค้าและผู้ใช้เข้ามาช่วย อาทิ เพศ อายุ หรือชนิดของภาพยนต์ แต่ก็ยังมีข้อจำกัดอีกประการหนึ่ง คือ ข้อมูลคุณลักษณะ (Feature) อาจจะไม่ได้มีในทุกชุดข้อมูล

จากปัญหาดังกล่าว จึงเกิดเป็นช่องว่างของงานวิจัย (Research Gap) คือ การพัฒนาระบบแนะนำสินค้าและบริการ ที่มีความทนทาน (Robust) ต่อปัญหา Sparsity ได้มากขึ้น และสามารถรับมือกับปัญหา User Cold Start และ Item Cold Start ได้อย่างมีประสิทธิภาพ ในสถานการณ์ที่ไม่มีข้อมูลคุณลักษณะ (Feature) ใดเพิ่มเติมเลยทั้งคุณลักษณะของผู้ใช้และคุณลักษณะของสินค้า

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

จากช่องว่างของงานวิจัย (Research Gap) วิทยานิพนธ์ฉบับนี้จึงมีวัตถุประสงค์ คือ

1. ออกแบบและพัฒนาโมเดลการแยกตัวประกอบเมทริกซ์โดยใช้สัญญาณการจัดกลุ่มและการหากฎความสัมพันธ์ (Clustering Association Rule Mining Signal Matrix Factorization – CARMS MF) ที่สามารถสกัดสัญญาณของสินค้าชิ้นถัดไป (Signal) จากข้อมูล โดยใช้การจัดกลุ่ม (Clustering) และการหากฎความสัมพันธ์ (Association Rule Mining)
2. สัญญาณ (Signal) ที่สกัดได้จากการจัดกลุ่มและกฎความสัมพันธ์ สามารถนำไปปรับปรุงและต่อยอดโมเดลพื้นฐาน คือ MF (กลายเป็น CARMS MF) และ Bias MF (กลายเป็น CARMS Bias MF) ผ่านการปรับปรุงฟังก์ชันเป้าหมาย (Objective Function)
3. ประสิทธิภาพของโมเดล CARMS MF และ CARMS Bias MF เปรียบเทียบกับโมเดลพื้นฐาน (MF และ Bias MF) จะมีความแม่นยำมากขึ้นโดยเฉพาะกับกลุ่มลูกค้าใหม่ (Cold Start User) ที่จะต้องมีความแม่นยำที่เพิ่มขึ้นอย่างเห็นได้ชัด
4. ไม่ใช้ข้อมูลคุณลักษณะ (Feature) ใดเพิ่มเติมเลยทั้งคุณสมบัติของผู้ใช้อละคุณสมบัติของสินค้า ซึ่งแตกต่างจากโมเดลแบบผสม (Hybrid) ทั่วไป ที่เน้นการใช้ข้อมูลคุณลักษณะเข้ามาเสริม

1.3 สมมุติฐานของการศึกษา

สมมติฐานของสถาปัตยกรรมโมเดลใหม่ (CARMS MF) ประกอบไปด้วย

1. โมเดล CARMS MF และ CARMS Bias MF ที่ผนวกสัญญาณแบบ CARMS จะให้ค่าความแม่นยำในการทำนายในภาพรวมสูงกว่า โมเดล MF และ Bias MF
2. โมเดล CARMS MF และ CARMS Bias MF จะสามารถลดผลกระทบจากปัญหา Use Cold Start และ Item Cold Start ได้ดีกว่า โมเดล MF และ Bias MF ที่ไม่ได้ใช้สัญญาณเหล่านี้ โดยดูจากความแม่นยำในกลุ่มของผู้ใช้ใหม่ (User Cold Start) ที่เพิ่มขึ้นเป็นอย่างมาก

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

แนวคิดหลักของวิทยานิพนธ์ฉบับนี้ คือนำเสนอโมเดลการแยกตัวประกอบเมทริกซ์โดยใช้สัญญาณการจัดกลุ่มและการหากฎความสัมพันธ์ (Clustering Association Rule Mining Signal Matrix Factorization – CARMS MF) โดย CARMS MF เป็นโมเดลแบบผสม (Hybrid Method) ที่ใช้พื้นฐานของ MF หรือ Bias MF แต่จะมีความใหม่และแตกต่าง (Novelty) คือการเพิ่มสัญญาณของสินค้าชิ้นถัดไป หรือสัญญาณของกลุ่มสินค้าชนิดถัดไปเพื่อให้โมเดลได้เรียนรู้สัญญาณนี้ไปพร้อมกับเมทริกซ์ผู้ใช้และสินค้า (User-Item Matrix) โดยมีขั้นตอนในการคำนวณ 3 ขั้นตอน ดังนี้

1. ใช้ข้อมูลจาก User-Item Matrix ในการจัดกลุ่มผู้ใช้ (User Clustering) และจัดกลุ่มสินค้า (Item Clustering)
2. ปรับปรุงธุรกรรม (Transaction) ของผู้ใช้แต่ละราย โดยใส่กลุ่มผู้ใช้และกลุ่มของสินค้าที่ได้จากการจัดกลุ่มผู้ใช้และกลุ่มสินค้า
3. ใช้การหากฎความสัมพันธ์ (Association Rule Mining) ร่วมกับ Transaction ที่มีการปรับปรุง (Augmented Transaction) เพื่อหากฎความสัมพันธ์มีหลากหลายมากขึ้น
4. ปรับปรุงสมการเป้าหมาย (Objective Function) ให้โมเดลสามารถเรียนรู้ปัจจัยแฝง (Latent Factor) ของผู้ใช้และของสินค้าด้วยวิธีการแยกองค์ประกอบเมทริกซ์ร่วม (Co-Factorization)

1.5 ขอบเขตของงานวิจัย

ขอบเขตของวิทยานิพนธ์และรวมถึงการวิจัยและการทดสอบผล มีดังนี้

1. ใช้ชุดข้อมูลสาธารณะ (Public Datasets) คือ MovieLen ซึ่งเป็นชุดข้อมูลที่ใช้กันอย่างแพร่หลาย จากงานศึกษาพบว่างานวิจัยที่เกี่ยวข้องกับระบบแนะนำสินค้าและบริการราว 20% จะใช้ชุดข้อมูล MovieLen เป็นตัวเปรียบเทียบประสิทธิภาพ (Roy et al., 2022)
2. การประเมินผลจะเปรียบเทียบโมเดล CARMS MF และ CARMS Bias MF กับ MF และ Bias MF
3. การวัดผลจะใช้ Normalized Discounted Cumulative Gain (NDCG) เนื่องจากระบบแนะนำสินค้าจะสนใจการเรียงลำดับมากกว่าการทำนายคะแนนที่ถูกต้อง และจากงานวิจัยชี้ให้เห็นว่ามีงานวิจัยเพียง 10% ที่ใช้การวัดผลด้านคะแนน อาทิ Root Mean Square Error (RMSE) ในการวัดผลระบบแนะนำสินค้า (Roy et al., 2022)

1.6 ขั้นตอนของการศึกษา

วิทยานิพนธ์ฉบับนี้แบ่งออกเป็น 5 บท ดังนี้

1. บทที่ 1 บทนำ: นำเสนอความเป็นมา วัตถุประสงค์ สมมติฐาน แนวคิดหลักของ CARMS MF รวมไปถึงขอบเขตของการวิจัยและการวัดผลของโมเดลที่จะนำเสนอ
2. บทที่ 2 ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง: ทบทวนทฤษฎีที่เกี่ยวข้อง คือ การจัดกลุ่มข้อมูล (Clustering) การหากฎความสัมพันธ์ (Association Rule Mining) และโมเดลพื้นฐาน คือ MF และ Bias MF ปัญหาความเบาบางของข้อมูล (Sparsity) ปัญหาผู้ใช้และสินค้าใหม่ (Cold Start) งานวิจัยที่มีผู้วิจัยแล้วในอดีต รวมถึงช่องว่างในการวิจัย (Research Gap)
3. บทที่ 3 วิธีวิจัย: อธิบายสถาปัตยกรรมของโมเดล CARMS MF โดยแบ่งเป็นการจัดกลุ่มข้อมูล (Clustering) การปรับปรุงรายการธุรกรรม (Augmented Transaction) การสร้างกฎความสัมพันธ์ (Association Rule Mining) การสร้างเมทริกซ์สัญญาณสินค้าชิ้นถัดไป และการปรับปรุงสมการเป้าหมาย (Objective Function) ด้วยเทคนิคการแยกตัวประกอบเมทริกซ์ร่วม (Co-Factorization) รวมไปถึงการนำเสนอคุณลักษณะของชุดข้อมูลที่จะใช้ทดสอบ
4. บทที่ 4 ผลการวิจัยและอภิปรายผล: นำเสนอผลลัพธ์ที่ได้จากการเปรียบเทียบ CARMS MF และ CARMS Bias MF กับโมเดลพื้นฐาน คือ MF และ Bias MF ในชุดข้อมูลที่แตกต่างกัน พร้อมทั้งสรุปอภิปรายผล
5. บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ: สรุปใจความสำคัญของงานวิจัย ข้อจำกัด และแนวทางในการต่อยอดงานวิจัยในอนาคต

บทที่ 2

ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ทฤษฎีระบบแนะนำสินค้าและบริการ

ระบบแนะนำสินค้า (Recommendation System) มีหลากหลายแบบด้วยกัน แต่สามารถแบ่งตามวิธีทำงานได้ 4 ประเภทใหญ่ ตามการจัดกลุ่มของ Aggarwal (2016) ดังนี้

1. Knowledge-Based: เป็นระบบที่แนะนำสินค้าหรือบริการให้กับผู้ใช้โดยอิงจากเงื่อนไข ร่วมกับความรู้ในสายนั้นๆ (Domain Knowledge) โดยไม่จำเป็นต้องอาศัยข้อมูลประวัติการให้คะแนนของผู้ใช้ ระบบจะทำงานโดยมุ่งเน้นที่การค้นหาสินค้าที่ตรงตามข้อกำหนดที่ผู้ใช้ป้อนเข้ามาหรือโดยการอนุมาน (Inferring) จากความรู้ที่มี ตัวอย่างเช่น แนะนำเสื้อผ้าผู้หญิงให้กับผู้ใช้ที่เป็นเพศหญิง
2. Content-Based Filtering (CBF) เป็นหนึ่งในประเภทพื้นฐานของระบบแนะนำสินค้าที่ทำงานโดยอาศัยการจับคู่ระหว่างคุณลักษณะ (Feature) ของสินค้า เช่น ผู้กำกับ นักแสดง หรือหมวดหมู่ของภาพยนต์ (Genre) กับการให้คะแนนของผู้ใช้ อาทิ หากผู้ใช้ A เคยให้คะแนนสูงกับภาพยนตร์ในหมวดหมู่ Sci-Fi ที่มีผู้กำกับคนหนึ่งๆ ระบบจะแนะนำภาพยนต์มีคุณลักษณะคล้ายคลึงกัน โดยกระบวนการทำงานของ CBF แบ่งออกเป็น 2 ขั้นตอน ได้แก่
    - การสร้างโปรไฟล์สินค้า (Item Profiling): สินค้าแต่ละชิ้นจะถูกแปลงเป็นเวกเตอร์ของคุณลักษณะ (Feature Vector) ซึ่งอาจใช้เทคนิคต่างๆ เช่น TF-IDF (Term Frequency-Inverse Document Frequency) หากคุณลักษณะเป็นข้อความ (เช่น คำบรรยายสินค้า)
    - การสร้างโปรไฟล์ผู้ใช้ (User Profiling): โปรไฟล์ผู้ใช้ p_u จะถูกสร้างโดยการรวม (Aggregation) เวกเตอร์คุณลักษณะของสินค้าที่ผู้ใช้ u เคยชอบเข้าด้วยกัน อาจใช้การหาค่าเฉลี่ยแบบถ่วงน้ำหนัก (Weighted Average) ในการรวมคะแนน ตามสมการที่ 2.1

$$p_u = \sum_{i \in R_u^+} w_i \cdot F_i \qquad 2.1$$

- การคำนวณคะแนนการแนะนำ (Recommendation Score Calculation): หลังจากได้โปรไฟล์สินค้าใหม่ F_{new} และโปรไฟล์ผู้ใช้ p_uแล้ว ระบบจะคำนวณ ความคล้ายคลึง (Similarity) ระหว่างสองเวกเตอร์นี้ ตามสมการที่ 2.2

$$\text{Score}(\text{new}, u) = \text{Similarity}(F_{\text{new}}, p_u) = \frac{F_{\text{new}} \cdot p_u}{\| F_{\text{new}} \| \cdot \| p_u \|} \qquad 2.2$$

ข้อดีของวิธีนี้คือสามารถแก้ปัญหา Item Cold Start ได้เป็นอย่างดี แต่ไม่สามารถแก้ปัญหา User Cold Start และไม่สามารถแนะนำสิ่งที่หลากหลาย (Overspecialization) ให้กับผู้ใช้ได้

3. Collaborative Filtering (CF): เป็นเทคนิคที่ใช้กันอย่างแพร่หลายในระบบแนะนำสินค้า โดยมีหลักการพื้นฐานคือ ผู้ใช้ที่มีพฤติกรรมคล้ายคลึงกันในอดีตจะมีแนวโน้มที่จะชอบสิ่งที่คล้ายคลึงกันในอนาคต (Homophily) ระบบนี้จะทำงานโดย ไม่จำเป็นต้องใช้ความรู้เกี่ยวกับคุณลักษณะ (Feature) ของสินค้าเลย ระบบแบบ CF ใช้ประโยชน์จากเมทริกซ์ผู้ใช้และสินค้า (User-Item Matrix) ซึ่งมักจะมีค่าว่างไปเป็นจำนวนมาก (สินค้าที่ผู้ใช้ยังไม่ได้ให้คะแนน) เป้าหมายของ CF คือการเติมค่าที่หายไปเพื่อทำนายความชอบของผู้ใช้ต่อสินค้าที่ยังไม่ให้คะแนน โดย CF แบ่งออกเป็น 2 ประเภท คือ

    - Memory-Based: ใช้วิธีคำนวณความคล้ายคลึง (เช่น Pearson Correlation หรือ Cosine Similarity) ระหว่างผู้ใช้ (User-based) หรือระหว่างสินค้า (Item-based) โดยตรงจาก User-Item Matrix ทั้งหมดเพื่อทำการทำนาย แม้จะตีความได้ง่ายแต่มีปัญหาด้านความสามารถในการขยายขนาด (Scalability) เมื่อข้อมูลมีขนาดใหญ่
    - Model-Based: ใช้วิธีการสร้างโมเดลทางสถิติหรือการเรียนรู้ของเครื่อง (Machine Learning) เช่น MF หรือ Bias MF เพื่อเรียนรู้ Pattern ที่ซับซ้อนจากข้อมูล โมเดลประเภทนี้มีจุดเด่นด้านความแม่นยำที่มักจะสูงกว่า Memory-Based อย่างมีนัยสำคัญ โดยเฉพาะในข้อมูลที่เบาบาง (Sparse) นอกจากนี้ยังมีข้อได้เปรียบด้าน Scalability ที่สามารถทำนายผลได้ไวกว่า

ข้อดีของ CF คือ มีความแม่นยำสูง โดยมักมีความแม่นยำสูงกว่า Memory-Based อย่างมีนัยสำคัญ แต่ก็ยังเผชิญกับข้อจำกัดหลายประการ โดยเฉพาะปัญหา Cold Start ทั้ง User Cold Start และ Item Cold Start ซึ่งกระทบต่อความแม่นยำของโมเดล

4. Hybrid Method: คือการรวมเอาเทคนิคการแนะนำตั้งแต่สองวิธีขึ้นไปมาทำงานร่วมกัน โดยมีเป้าหมายหลักเพื่อจัดการกับข้อจำกัดของเทคนิคเดียว (เช่น ปัญหา Cold Start) ซึ่งโดยทั่วไปจะสามารถเพิ่มประสิทธิภาพและความแม่นยำในการแนะนำที่สูงขึ้นได้ การผสมผสานสามารถทำได้หลายรูปแบบ เช่น การเลือกใช้โมเดลที่เหมาะสมตามสถานการณ์ปัจจุบัน (Switching Hybridization) ตัวอย่างเช่น การใช้ Content-Based Filtering สำหรับผู้ใช้ใหม่เพื่อแก้ปัญหา Cold Start ก่อนที่จะสลับไปใช้ Collaborative Filtering เมื่อมีข้อมูลเพียงพอ หรือการรวมถึงการรวมผลลัพธ์ (Output) ของระบบต่างๆ เข้าด้วยกัน, เช่น การใช้ค่าเฉลี่ยถ่วงน้ำหนัก (Weighted Hybridization) เพื่อคำนวณเป็นคำแนะนำสุดท้ายเพียงค่าเดียว

2.1.2 ทฤษฎีการแยกตัวประกอบเมทริกซ์ (Matrix Factorization)

แนวคิดหลักคือการย่อยสลาย (Decompose) เมทริกซ์ผู้ใช้และสินค้า (User-Item Matrix) (Y) ซึ่งมีขนาดเท่ากับ ($m \times n$) ให้กลายเป็นเมทริกซ์ (Matrix) ขนาดเล็ก 2 ตัว (Koren et al., 2009) ซึ่งประกอบด้วย

1. User Matrix (P): ขนาด $m \times k$ (m แทนจำนวนผู้ใช้ และ k แทนจำนวนปัจจัยแฝง โดย p_u คือเวกเตอร์ปัจจัยแฝงของผู้ใช้ u
2. Item Matrix (Q): ขนาด $n \times k$ (n แทนจำนวนสินค้า และ k แทนจำนวนปัจจัยแฝง) โดย q_i คือเวกเตอร์ปัจจัยแฝงของสินค้า i

การทำนายคะแนน $\hat{y}_{ui}$ ของผู้ใช้ u ต่อสินค้า i คำนวณจากผล Dot product ของเวกเตอร์แฝง (Latent Vector) สองตัว โดย k คือ Hyperparameter ที่มีค่าน้อยกว่า m และ n ตามสมการที่ 2.3

$$y = q_i^T p_u \qquad 2.3$$

โมเดลจะเรียนรู้ P และ Q โดยการพยายามลดค่าความคลาดเคลื่อน (Error) บนชุดข้อมูลที่ผู้ใช้ลงคะแนนแล้ว (Observed Rating) ($\mathcal{K}$) พร้อมตัวกำกับ (Regularization) (λ) เพื่อป้องกันปัญหา Overfitting ผ่านฟังก์ชันเป้าหมาย (Objective Function) ตามสมการที่ 2.4

$$\min_{p,q} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2) \qquad 2.4$$

2.1.3 ทฤษฎีการแยกตัวประกอบเมทริกซ์ที่เพิ่มความเอนเอียง (Bias Matrix Factorization)

ในการแข่งขัน Netflix Prize หนึ่งในบทเรียนสำคัญที่นำไปสู่การพัฒนาระบบแนะนำสินค้าอย่างก้าวกระโดด คือการค้นพบว่าโมเดล MF แบบดั้งเดิมนั้นมีข้อจำกัดในการอธิบายพฤติกรรมการให้คะแนนที่ซับซ้อนของมนุษย์ (Koren et al., 2009) งานวิจัยชี้ให้เห็นว่าปัจจัยสำคัญที่ช่วยเพิ่มความแม่นยำของโมเดล MF คือการเพิ่มองค์ประกอบของ ความเอนเอียง (Bias) เข้าไปในสมการ

แนวคิดนี้ตั้งอยู่บนข้อสังเกตที่ว่า คะแนนที่ผู้ใช้ให้ (Ratings) ส่วนใหญ่ไม่ได้เกิดจากปฏิสัมพันธ์แฝง (Latent) ระหว่างรสนิยมเฉพาะตัวของผู้ใช้กับคุณลักษณะเฉพาะของสินค้าเพียงอย่างเดียว แต่คะแนนเหล่านั้นได้รับผลกระทบโดยอิทธิพลที่คงที่และสังเกตได้ง่ายกว่า คือ ความเอนเอียง (Bias) ที่เกิดจากตัวผู้ใช้หรือตัวสินค้า หากไม่จัดการอิทธิพลเหล่านี้ให้ถูกต้อง โมเดลจะเรียนรู้ได้ยากขึ้น ดังนั้นโมเดล Bias MF จึงพยายามแยกแยะและอธิบายอิทธิพลเหล่านี้ก่อน ประกอบด้วย 3 ส่วนหลัก

- Global Bias (ความเอนเอียงรวม): คือค่าเฉลี่ยของคะแนนในชุดข้อมูล ทำหน้าที่เป็นค่าพื้นฐาน (Baseline) ที่สุดของระบบ
- User Bias (ความเอนเอียงของผู้ใช้): ผู้ใช้บางคนมีแนวโน้มที่จะให้คะแนนสูงกว่าคนอื่น (เช่น ให้ 4-5 ดาวตลอด) ในขณะที่ผู้ใช้บางคนอาจเป็นนักวิจารณ์ที่เข้มงวดและมักจะให้คะแนนต่ำกว่าค่าเฉลี่ย
- Item Bias (ความเอนเอียงของสินค้า): โดยสินค้าบางชิ้น (เช่น ภาพยนตร์บางเรื่อง) ได้รับการยอมรับในวงกว้างว่าดี จึงมักจะได้รับคะแนนสูงกว่าค่าเฉลี่ย ในทางกลับกัน บางเรื่องก็อาจจะถูกมองว่าแย่กว่าเรื่องอื่น

Bias MF จึงพยายามระบุอิทธิพลของความเอนเอียงเหล่านี้ก่อน ดังนั้นสมการประมาณคะแนน ($\hat{y}$) ของ Bias MF จึงถูกปรับไปตามสมการที่ 2.5

$$y = \mu + b_i + b_u + q_i^T p_u \qquad 2.5$$

และสมการเป้าหมาย (Objective Function) จะถูกปรับไปเป็นตามสมการที่ 2.6

$$\min_{p,q,\mu,b_i,b_u} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - \mu + b_i + b_u + q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2) \qquad 2.6$$

2.1.4 การเรียนรู้ (Training) ของการแยกตัวประกอบเมทริกซ์ (Matrix Factorization)

การเรียนรู้ P และ Q มักทำผ่าน 2 วิธีหลัก (Koren et al., 2009) คือ

1. Stochastic Gradient Descent (SGD): ปรับ (Update) ค่าพารามิเตอร์ (Parameter) ทีละจุดข้อมูล โดยคำนวณค่าความคลาดเคลื่อน (Error) จาก $e_{ui} = y_{ui} - \hat{y}_{ui}$ และ Update ค่า p_u และ q_i ตามทิศทางตรงข้าม Gradient เช่น $p_u \leftarrow p_u + \eta(e_{ui}q_i - \lambda p_u)$ ซึ่งสามาถพัฒนาไปเป็น Mini-Batch Gradient Descent ซึ่งเสถียรกว่า SGD (เพราะใช้ข้อมูลหลายจุดมาเฉลี่ย Gradient) และเร็วกว่า Full-Batch (เพราะไม่ต้องรอคำนวณข้อมูลทั้งหมด) โดยมากจะตั้งขนาดของกลุ่มการเรียนรู้ (Batch Size) เท่ากับ 128, 256 หรือ 512 ทำให้โดยรวมแล้วโมเดลเรียนรู้ได้เสถียรขึ้นถ้าเทียบกับ Stochastic Gradient Descent และยังคงเรียนรู้ได้ไวกว่าถ้าเทียบกับ Full-Batch Gradient Descent
2. Alternating Least Squares (ALS): ใช้วิธีตรึง (Fix) P ไว้ แล้วแก้สมการหา Q ที่ดีที่สุด จากนั้นตรึง Q แล้วแก้สมการหา P ทำสลับกันไป วิธีการนี้เหมาะกับข้อมูลที่เป็นความชอบแฝง (Implicit Data) มากกว่าความชอบโดยตรง (Explicit Data) จุดเด่นคือเมื่อตรึงปัจจัยฝั่งหนึ่งไว้ ปัญหาจะกลายเป็นโจทย์ Least Squares มาตรฐานที่แก้สมการได้โดยตรง ทำให้ในแต่ละก้าวของการสลับ (Alternating Step) สามารถคำนวณแบบขนาน (Parallelize) เพื่อหาค่าปัจจัยแฝงทั้งหมด (เช่น p_u ของผู้ใช้ทุกคน) ได้พร้อมกันอย่างมีประสิทธิภาพ โดยเฉพาะกับข้อมูลที่เบาบาง (Sparse)

2.1.5 ทฤษฎีการจัดกลุ่ม (Clustering)

Clustering เป็นแนวทางหนึ่งของการเรียนรู้แบบไม่มีผู้สอน (Unsupervised) เป้าหมายคือการจัดกลุ่มของข้อมูลโดยข้อมูลที่อยู่ในกลุ่ม (Cluster) เดียวกันจะมีความคล้ายคลึงกัน (Similarity) และข้อมูลที่อยู่ต่างกลุ่มกันจะมีความแตกต่างกัน (Dissimilarity)

K-Means เป็นหนึ่งในอัลกอริทึม (Algorithm) ในการจัดกลุ่มที่ได้รับความนิยมมากที่สุด (Sinaga & Yang, 2020) และจัดเป็นวิธีการจัดกลุ่มโดยใช้ระยะทาง (Distance Base) โดยK-Means จะพยายามแบ่งข้อมูลทั้งหมดออกเป็นกลุ่มย่อยตามจำนวนกลุ่ม (k) โดยทำงานแบบวนซ้ำเพื่อลดค่าในฟังก์ชันเป้าหมาย Within-Cluster-Sum-of-Squares (WCSS) ตามสมการที่ 2.7

$$J(z, A) = \sum_{i=1}^{n}\sum_{k=1}^{c} z_{ik} \parallel y_i - \mu_k \parallel^2 \qquad 2.7$$

ขั้นตอนในการคำนวณของ K-Means จะมีทั้งสิ้น 4 ขั้นตอน ดังนี้

1. Initialization (การเริ่มต้น): เลือกจุดศูนย์กลาง (Centroids) จำนวน k จุด โดยแต่ละจุดแทนด้วย μ_k
2. Assignment Step (การกำหนดสมาชิก)
   - คำนวณระยะห่างระหว่างจุดข้อมูล y_i แต่ละจุดกับ Centroid μ_k ทุกจุด (โดยทั่วไปจะใช้ฟังก์ชันระยะทางแบบ Euclidean Distance)
   - กำหนดให้จุดข้อมูล y_i เป็นสมาชิกของ k ที่มีระยะทางใกล้ที่สุด (มีค่า $\|y_i - \mu_k\|^2$ น้อยที่สุด)
3. Update Step (การ Update จุด Centroid)
   - คำนวณตำแหน่ง Centroid μ_k ใหม่ โดยการหาค่าเฉลี่ย (Mean) ของจุดข้อมูล y_i ทั้งหมดที่ถูกกำหนดให้อยู่ใน Cluster k นั้น
4. Repeat (ทำซ้ำ)
   - ทำซ้ำขั้นตอนที่ 2 (Assignment) และ 3 (Update) ไปจนกระทั่งจุด Centroid ไม่มีการเปลี่ยนแปลงตำแหน่ง หรือเปลี่ยนแปลงน้อยมาก (Convergence)

2.1.6 การวิเคราะห์ตะกร้าสินค้า (Market Basket Analysis)

Market Basket Analysis (MBA) หรือการวิเคราะห์ตะกร้าสินค้า คือเทคนิคที่ใช้ค้นหากฎความสัมพันธ์ในข้อมูลขนาดใหญ่ (Big Data) เพื่อทำความเข้าใจพฤติกรรมลูกค้า ตัวอย่างเช่น การหาว่าลูกค้ามักซื้อสินค้าอะไรคู่กัน เพื่อนำไปจัดโปรโมชั่นขายพ่วง (Bundle)

เทคนิคที่นิยมใช้ทำ MBA คือ Association Rule Mining (ARM) ซึ่งเหมาะสำหรับการวิเคราะห์ธุรกรรม (Transaction) จำนวนมาก เพื่อค้นหาชุดสินค้าที่ถูกซื้อบ่อย โดยจะมีอัลกอริทึม (Algorithm) ที่นิยมใช้ ดังนี้

1. Apriori: เป็นขั้นตอนวิธีที่ใช้ในการนำ ARM มาใช้งานในยุคแรกเริ่ม โดยอาศัยหลักการสร้างชุดสินค้าท้าชิง (Candidate Item Sets) ในแต่ละรอบ อย่างไรก็ตามวิธีการนี้อาจก่อให้เกิดปัญหาคอขวด (Bottleneck) ในการคำนวณเมื่อต้องประมวลผลฐานข้อมูลขนาดใหญ่

2. FP-Growth (Frequent Pattern-Growth): เป็นขั้นตอนวิธีที่ได้รับการพัฒนาขึ้นเพื่อปรับปรุงประสิทธิภาพของ Apriori โดยใช้โครงสร้างข้อมูลแบบต้นไม้ (FP-Tree) ที่ช่วยบีบอัดฐานข้อมูลธุรกรรม จุดเด่นของ FP-Growth คือสามารถสร้างชุดสินค้าที่เกิดบ่อยได้ โดยไม่ต้องสร้างชุดสินค้าผู้ท้าชิงทำให้โดยทั่วไปมีเวลาในประมวลผลไวกว่า Apriori

ผลลัพธ์ที่ได้จาก ARM ไม่ว่าจะเป็น Algorithms ใดก็ตามคือกฎ (Rule) ซึ่งถูกนิยามในรูปแบบ $X \rightarrow Y$ โดยที่ X และ Y คือSetย่อยของสินค้าทั้งหมด $(X \subset I, U \subset I)$ และไม่มีสินค้าชิ้นใดร่วมกัน $(X \cap I = \emptyset)$ หากแปลเงื่อนไขเหล่านี้กับข้อมูลในโลกแห่งความเป็นจริง เช่น ห้างสรรพสินค้า (Supermarket) กฎ $X \rightarrow Y$ หมายความว่าการซื้อ X (Antecedent หรือ ส่วนนำ) ซึ่งอาจเป็นสินค้าชิ้นเดียวหรือสินค้าหลายชิ้น สื่อโดยนัยถึงการซื้อ Y (Consequent หรือ ส่วนตาม) ซึ่งอาจเป็นสินค้าชิ้นเดียวหรือสินค้าหลายชิ้นก็ได้เช่นกัน

หลังจากที่ได้กฎแล้ว จำเป็นต้องมีกระบวนการประเมินคุณภาพเพื่อวัดความน่าสนใจและความสำคัญของกฎ (ซึ่งอาจมีการเลือกบางกฎที่มีความสำคัญน้อยออกในขั้นตอนนี้) โดยตัวชี้วัดคุณภาพที่รู้จักกันดีที่สุดคือ Support และ Confidence และ Lift

1. Support (ค่าสนับสนุน): คือสัดส่วนของธุรกรรมในฐานข้อมูล ที่มีทั้ง A และ B เกิดขึ้นพร้อมกัน (Itemset) หรือก็คือความน่าจะเป็นที่จะพบ A และ B พร้อมกันในธุรกรรมเดียว หน้าที่ของ Support จะถูกใช้เพื่อบ่งชี้ความถี่ โดย Itemset ที่มีค่า Support ต่ำอาจมีความสำคัญน้อยเนื่องจากเป็นสิ่งที่เกิดขึ้นไม่บ่อยนัก จากสมการที่ 2.8 มีความหมายคือ ความน่าจะเป็นที่ Itemset คือ A และ B เกิดขึ้นพร้อมกันในธุรกรรมเดียวกัน

$$supp(A \rightarrow B) = P(A \cap B) \qquad 2.8$$

2. Confidence (ค่าความเชื่อมั่น): หรือความแข็งแกร่งของกฎ (Strength of the rule) เป็นการวัดความน่าจะเป็นแบบมีเงื่อนไข (Conditional probability) ที่บ่งชี้ว่าหากสินค้าหรือชุดของสินค้ามีการเกิดขึ้นอยู่แล้ว (Antecedent) จะมีสินค้าหรือชุดของสินค้าที่สนใจ เกิดขึ้นตามมาบ่อยเพียงใด (Consequent) จากสมการที่ 2.9 สามารถตีความได้ว่า ความน่าจะเป็นที่จะเจอ B เมื่อ (Given) A เกิดขึ้นแล้ว

$$conf(A \rightarrow B) = P(B \mid A) = \frac{P(A \cap B)}{P(A)} \qquad 2.9$$

3. Lift (ค่าแรงสนับสนุน): เป็นหนึ่งในตัววัดคุณภาพเพิ่มเติมที่ใช้ประเมินว่าการมีอยู่ของสินค้าหรือชุดของสินค้ามีปฏิสัมพันธ์กันในเชิงบวกหรือเชิงลบ หรือเป็นเพียงการเกิดขึ้นพร้อมกันโดยบังเอิญ (เนื่องจาก อาจเป็นสินค้ายอดนิยมที่เกิดขึ้นบ่อยโดยปกติอยู่แล้ว) สมการที่ 2.10 สามารถตีความได้ว่ามี A ทำให้โอกาสเจอ B เพิ่มขึ้นกี่เท่า เมื่อเทียบกับโอกาสปกติที่จะเจอ B อยู่แล้ว

$$lift(A \rightarrow B) = \frac{P(B \mid A)}{P(B)} = \frac{conf(A \rightarrow B)}{P(B)} \qquad 2.10$$

โดยสรุปแล้ว ARM จะมี Hyperparameter ที่ต้องพิจารณาทั้งสิ้น 3 ตัว เพื่อกรองเอาเงื่อนไขที่เหมาะสมมาใช้ต่อ เริ่มต้นจาก (1) Support ซึ่งใช้กรอง Itemset ที่เกิดน้อยเกินไป (Noise) ให้เหลือเฉพาะ Itemset ที่พบบ่อยและมีความสำคัญทางธุรกิจ จากนั้นใช้ (2) Confidence ใช้สร้างกฎที่น่าเชื่อถือว่าถ้าเกิด A แล้ว B จะตามมาบ่อยแค่ไหน และสุดท้ายจะใช้ (3) Lift ยืนยันว่ากฎจาก Confidence นั้นเป็นความสัมพันธ์ที่เกี่ยวข้องกันจริงๆ ไม่ใช่แค่เพราะ B เป็นสินค้ายอดนิยม

2.1.7 Root Mean Square Error (RMSE)

RMSE คือรากที่สองของค่าเฉลี่ยของความคลาดเคลื่อนกำลังสอง (Mean Squared Error - MSE) โดยมีเป้าหมายเพื่อวัดว่าค่า Rating ที่ระบบทำนาย ($\hat{y}_{ui}$) อยู่ใกล้เคียงกับค่า Rating ที่ผู้ใช้ให้จริง (y_{ui}) มากเพียงใด คำนวณจากผลรวมของค่าความคลาดเคลื่อน ($\hat{y}_{ui} - y_{ui}$) ยกกำลังสอง ตามสมการที่ 2.11

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,j) \in N} (\hat{y}_{uj} - y_{uj})^2}{n}} \qquad 2.11$$

จุดอ่อนของ RMSE สำหรับการใช้วัดผลระบบแนะนำสินค้าคือ การที่ RMSE เป็นการวัดความผิดพลาดของการทำนายคะแนน $\hat{y}_{ui}$ ซึ่งไม่ได้คำนึงถึงการเรียงลำดับของคำแนะนำ (Ranking)

2.1.8 Normalized Discounted Cumulative Gain (NDCG)

NDCG เป็นหนึ่งในกลุ่มตัวชี้วัดที่ใช้วัดการจัดอันดับ (Ranking) โดยเป้าหมายของ NDCG จะวัดคุณภาพของรายการแนะนำ โดยเน้นว่ารายการที่เกี่ยวข้องหรือมีประโยชน์ (Utility) สูง ควรถูกจัดไว้ในอันดับต้นๆ แนวคิดพื้นฐานมาจาก Discounted Cumulative Gain (DCG) ซึ่งจะบวกค่าความเกี่ยวข้อง

(Gain หรือ Utility, g_{ui}) ของแต่ละรายการ โดยมีส่วนหัก (Discount) กับรายการที่อยู่อันดับล่างๆ โดยทั่วไปใช้ $log_2(v_j + 1)$ เป็นตัวหาร เมื่อ v_j คืออันดับของรายการ ตามสมการที่ 2.12

$$\mathrm{DCG} = \sum_{j \in I_u, v_j \le L} \frac{g_{uj}}{\log_2(v_j + 1)} \qquad 2.12$$

เพื่อให้คะแนนเป็นมาตรฐาน (Normalize) เราต้องรู้ก่อนว่า "คะแนนที่ดีที่สุดที่เป็นไปได้" ในลิสต์นั้นคือเท่าไหร่ IDCG คือการคำนวณ DCG ของรายการชุดเดิม แต่เรียงลำดับจากความเกี่ยวข้องมากไปน้อยอย่างสมบูรณ์ (Perfect Sorting) ตามสมการที่ 2.13

$$IDCG_k = \sum_{i=1}^{|REL|} \frac{rel_i^{ideal}}{\log_2(i+1)} \qquad 2.13$$

สุดท้ายจะแปลงค่าให้อยู่ระหว่าง 0 ถึง 1 โดยการนำคะแนนที่โมเดลทำได้ (DCG) หารด้วยคะแนนที่ดีที่สุดที่เป็นไปได้ (IDCG) ตามสมการที่ 2.14

$$NDCG_k = \frac{DCG_k}{IDCG_k} \qquad 2.14$$

2.1.9 Cross-Validation

การวัดผลของระบบแนะนำสินค้าจำเป็นต้องมีการประเมินผลที่ถูกต้องและแม่นยำเพื่อให้มั่นใจว่าโมเดลจะสามารถทำนายความชอบของผู้ใช้ได้อย่างมีประสิทธิภาพ ในทางปฏิบัติ ข้อมูลการให้คะแนน (Rating) ทั้งหมดที่มีอยู่ไม่สามารถนำมาสร้างโมเดลและทดสอบในเวลาเดียวกันได้ เนื่องจากจะถือว่าเป็นการรั่วไหลของข้อมูล (Data Leakage) และจะทำให้ค่าความแม่นยำที่ได้สูงเกินความเป็นจริง (Overfitting) ดังนั้น การแบ่งข้อมูล (Data Segmentation) จึงเป็นขั้นตอนสำคัญในการทดลอง (Aggarwal, 2016)

การประเมินผลมักแบ่งข้อมูลออกเป็นส่วนๆ โดยส่วนหนึ่งใช้สำหรับสร้างโมเดล (Training) และอีกส่วนหนึ่งใช้สำหรับทดสอบ (Testing) โดยวิธีการแบ่งข้อมูลที่เป็นมาตรฐานมีดังนี้:

1. Hold-Out: เป็นวิธีพื้นฐานที่แบ่งข้อมูล Rating ออกเป็นสองส่วน โดยส่วนหนึ่งถูกซ่อนไว้ (Hidden) เพื่อใช้เป็นชุดทดสอบ และข้อมูลที่เหลือใช้สำหรับการฝึกฝนโมเดล (Training) ความแม่นยำจะวัดจากความสามารถของโมเดลในการทำนายค่าที่ถูกซ่อนไว้

2. Cross-Validation: เพื่อให้การวัดผลมีความน่าเชื่อถือและลดอคติจากการเลือกชุดข้อมูลทดสอบมากกว่าวิธี Hold-Out วิธี Cross-Validation จะแบ่งข้อมูลที่มีอยู่ออกเป็น q ส่วนเท่าๆ กัน ในแต่ละรอบการทดสอบ ข้อมูล 1 ส่วนจะถูกใช้เป็นชุดทดสอบ (Test Set) และอีก $q-1$ ส่วนที่เหลือจะถูกใช้เป็นชุดฝึกฝน (Training Set) กระบวนการนี้จะทำซ้ำ q ครั้ง โดยหมุนเวียนให้แต่ละส่วนเป็นชุดทดสอบจนครบ และนำค่าความแม่นยำเฉลี่ยมาเป็นผลลัพธ์สุดท้าย

ในงาน Recommender Systems การทำ Cross-Validation มีความแตกต่างจากงานการแบ่งข้อมูล (Classification) ทั่วไป ตรงที่การแบ่งข้อมูลจะทำในระดับรายการ (Entry-wise) ไม่ใช่ระดับแถว (Row-wise) กล่าวคือ ระบบจะสุ่มซ่อนค่า Rating บางค่าใน Matrix ไม่ใช่การซ่อนข้อมูลผู้ใช้ทั้งคน ซึ่งสะท้อนลักษณะของปัญหา Matrix Completion ที่ค่าความชอบของผู้ใช้อาจหายไปในช่องใดก็ได้

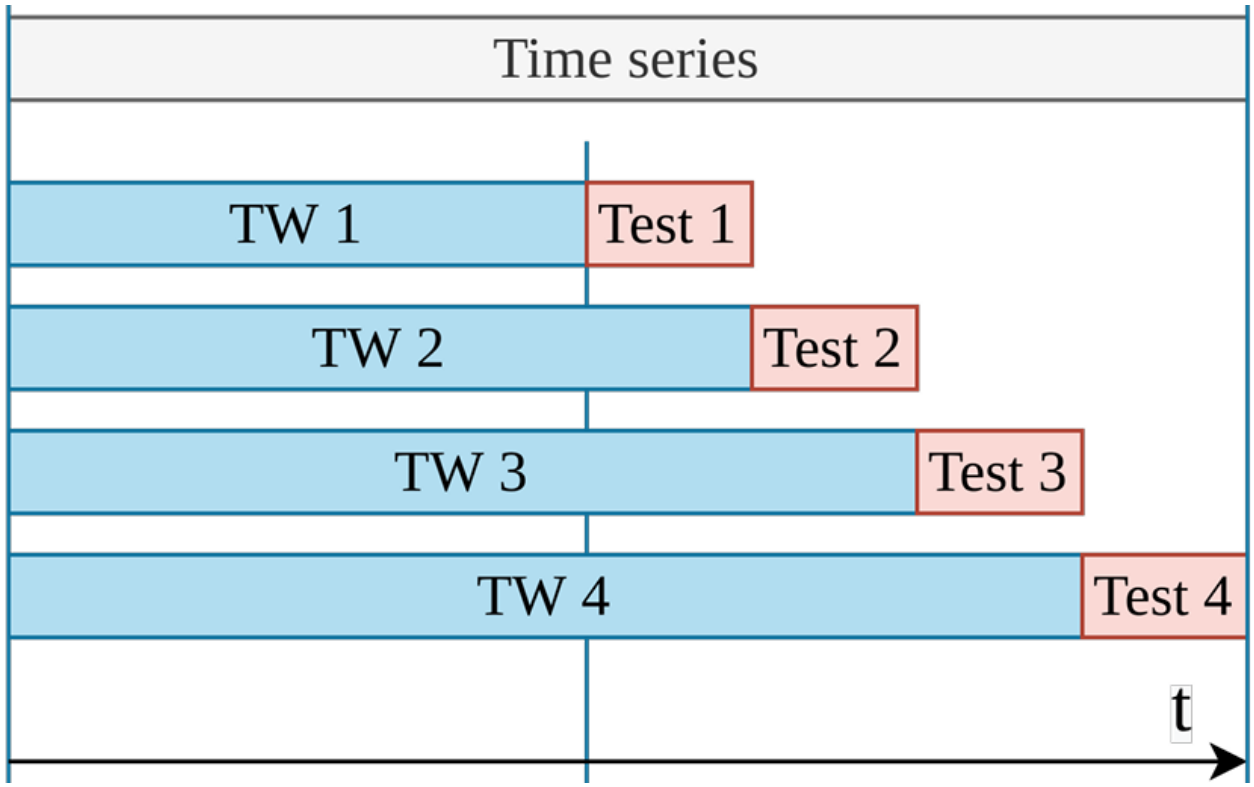
และแม้ว่าวิธี Cross-Validation แบบสุ่ม (Random Split) จะเป็นมาตรฐานในการทดสอบความแม่นยำของการเรียนรู้ของเครื่อง (Machine Learning) ทั่วไป แต่ Aggarwal, 2016 ได้เน้นย้ำประเด็นสำคัญว่าการสุ่มเลือกข้อมูลที่หายไป (Random Missing Data) อาจไม่สะท้อนความเป็นจริงเสมอไป เนื่องจากการให้ Rating ของผู้ใช้มักมีลำดับเวลาเข้ามาเกี่ยวข้อง

Aggarwal, 2016 ระบุว่าการแข่งขัน Netflix Prize เป็นกรณีศึกษาที่ชัดเจนที่สุดของการแก้ไขปัญหานี้ โดยไม่ได้ใช้วิธีสุ่มแบ่งข้อมูลแบบดั้งเดิม แต่ใช้วิธีการแบ่งตามลำดับเวลาซึ่งเปรียบเสมือนการจำลองสถานการณ์จริงแบบ Expanding Window

- Netflix แบ่งชุดข้อมูลออกเป็น Training Set ซึ่งประกอบด้วยข้อมูล Rating ในอดีต (ประมาณ 96% ของข้อมูล) และ Qualifying Set ซึ่งเป็นข้อมูล Rating ที่เกิดขึ้นหลังจากช่วงเวลาของ Training Set
- แนวทางนี้คือการบังคับให้โมเดลเรียนรู้จากอดีตเพื่อทำนายอนาคต โดยห้ามไม่ให้ข้อมูลในอนาคตรั่วไหลกลับไปสู่กระบวนการฝึกฝน (No Look-ahead Bias)

การออกแบบการทดลองเช่นนี้ของ Netflix ได้กลายเป็นมาตรฐาน (Standard Benchmark) ที่แสดงให้เห็นว่า สำหรับระบบแนะนำแล้ว การวัดผลที่แม่นยำที่สุดต้องเกิดจากการทดสอบกับข้อมูลที่ใหม่กว่าข้อมูลที่ใช้สอนเสมอ

รูปที่ 2.1: การแบ่งข้อมูลด้วยวิธี Expanding Window



ที่มา: Markudova et al. (2021)

2.1.10 Hyperparameter Tuning

การปรับจูนไฮเปอร์พารามิเตอร์ (Hyperparameter Tuning) เป็นขั้นตอนสำคัญในการสร้างโมเดลระบบแนะนำที่มีประสิทธิภาพ เนื่องจากพารามิเตอร์เหล่านี้ส่งผลโดยตรงต่อความสามารถในการทำนายของโมเดล (Aggarwal, 2016) ค่าพารามิเตอร์บางตัวไม่ได้ถูกเรียนรู้จากข้อมูลโดยตรง แต่ต้องถูกกำหนดโดยผู้สร้างโมเดล อาทิ Regularization (λ) ใน MF หากเลือกค่าที่ไม่เหมาะสม อาจทำให้โมเดลมีความแม่นยำต่ำ

การค้นหาแบบตาราง (Grid Search) เป็นวิธีที่นิยมที่สุดในการปรับจูนไฮเปอร์พารามิเตอร์ (Hyperparameter Tuning) โดยหลักการทำงานของ Grid Search มีขั้นตอนดังนี้:

1. กำหนดชุดค่าพารามิเตอร์ต่างๆ ที่เป็นไปได้ขึ้นมา (เช่น λ เท่ากับ 0.01 และ 0.05 ตามลำดับ)
2. ทำการรันหรือสร้างโมเดลหลายๆ ตัว โดยแต่ละตัวจะใช้ค่าพารามิเตอร์ที่แตกต่างกัน

3. แทนที่จะทดสอบกับชุดข้อมูลเพียงชุดเดียว Aggarwal, 2016 ระบุว่า ห้ามใช้ชุดข้อมูลทดสอบ (Test Set) ในการปรับจูนพารามิเตอร์โดยเด็ดขาด เพราะจะทำให้ผลลัพธ์การวัดผลมีความลำเอียง แต่ควรแบ่งข้อมูลส่วนหนึ่งออกมาเป็น Validation Set หรือใช้เทคนิค Cross-Validation ดังที่กล่าวไปในส่วน 2.3.9 ร่วมกันเพื่อใช้ในการปรับค่าเหล่านี้แทน เพื่อลดความลำเอียงของข้อมูล

4. ทำการคำนวณ ค่าเฉลี่ยของประสิทธิภาพ (Average Performance) จากทุกรอบการทดสอบ (Folds) โมเดลที่ให้ค่าเฉลี่ยดีที่สุด (เช่น ค่าความผิดพลาดเฉลี่ยต่ำที่สุด หรือค่าความแม่นยำเฉลี่ยสูงที่สุด) จะถือว่าเป็นโมเดลที่ใช้พารามิเตอร์ที่เหมาะสมที่สุด และจะถูกนำไปเทรนใหม่ (Retrain) กับข้อมูลทั้งหมดเพื่อใช้งานจริง

การทำ Hyperparameter Tuning ด้วยแนวคิดแบบ Grid Search เป็นวิธีมาตรฐานที่ช่วยให้มั่นใจว่าโมเดลระบบแนะนำจะทำงานได้เต็มประสิทธิภาพ โดยหัวใจสำคัญคือการแยกกระบวนการนี้ออกจากขั้นตอนการทดสอบจริง (Testing) อย่างเด็ดขาด เพื่อให้ผลการวัดความแม่นยำมีความน่าเชื่อถือและสะท้อนการใช้งานจริงมากที่สุด

2.2 วิจัยที่เกี่ยวข้อง

2.2.1 การเพิ่มความเอนเอียงในการแยกตัวประกอบเมทริกซ์ (Bias MF)

เทคนิค Matrix Factorization (MF) คือหนึ่งในโมเดลที่ประสบความสำเร็จในการสร้างระบบแนะนำสินค้า โดยเฉพาะอย่างยิ่งหลังจากการแข่งขัน Netflix Prize เทคนิคนี้ได้รับความนิยมเนื่องจากแม่นยำและสามารถขยายระบบได้ (Scalability) ถือเป็นพื้นฐานสำคัญสำหรับระบบแนะนำสินค้า

โมเดล Matrix Factorization พื้นฐาน จะประมาณการคะแนน ($\hat{y}_{ui}$) โดยอาศัยการคำนวณ Dot Product ระหว่างเวกเตอร์แฝง (Latent Vector) ของผู้ใช้ (p_u) และสินค้า (q_i) ดังสมการที่ 2.15

$$y = q_i^T p_u \qquad 2.15$$

อย่างไรก็ตาม Koren et al. (2009) ได้ชี้ให้เห็นว่าความผันผวนของคะแนนที่สังเกตได้ส่วนใหญ่ไม่ได้เกิดจากปฏิสัมพันธ์ (Interaction) ระหว่างผู้ใช้และสินค้าเพียงอย่างเดียว แต่เกิดจากอคติ (Bias) ที่เกี่ยวข้องกับตัวผู้ใช้หรือตัวสินค้า อาทิ ผู้ใช้บางคนมีแนวโน้มที่จะให้คะแนนสูงกว่าคนอื่นหรือสินค้าบางชิ้นมีแนวโน้มที่จะได้รับคะแนนสูงกว่าชิ้นอื่นโดยเฉลี่ย

ด้วยเหตุนี้ การพยายามอธิบายคะแนนทั้งหมดด้วย $q_i^T p_u$ เพียงอย่างเดียวจึงอาจไม่เหมาะสม ผู้วิจัยจึงได้เสนอโมเดลที่เพิ่ม Bias เข้าไป เพื่อแยกองค์ประกอบของคะแนน ดังสมการที่ 2.16

$$y = \mu + b_i + b_u + q_i^T p_u \qquad 2.16$$

และสมการเป้าหมายจะถูกปรับไปตามสมการที่ 2.17

$$\min_{p,q,\mu,b_i,b_u} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - \mu + b_i + b_u + q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2) \qquad 2.17$$

การเพิ่ม Bias เข้าไปใน MF แบบดั้งเดิมช่วยเพิ่มความแม่นยำในการทำนายได้อย่างมีนัยสำคัญ ทั้งนี้แม้ Bias MF ยังคงเผชิญกับปัญหา Cold Start อยู่ซึ่ง Koren et al. (2009) แนะนำให้ใช้ร่วมกับการเพิ่มคุณลักษณะ (Feature) เข้าไปเพิ่มเติม สิ่งที่สามารถนำไปต่อยอดได้คือการนำโมเดลพื้นฐาน MF และ Bias MF มาปรับปรุงให้เป็นโมเดล CARMS MF แต่จะไม่ได้เพิ่มเพิ่มคุณลักษณะ (Feature) เข้าไปเพิ่มเติมตามที่ Koren et al. (2009) เสนอ

2.2.2 การใช้เทคนิคการจัดกลุ่ม

งานวิจัยของ Mirbakhsh & Ling (2015) นำเสนอโมเดล Cluster-based Matrix Factorization (CBMF) ซึ่งเป็นการนำเทคนิคการจัดกลุ่ม (Clustering) มาประยุกต์ใช้เพื่อเพิ่มประสิทธิภาพให้กับระบบแนะนำสินค้า แนวคิดหลักของ paper นี้คือการใช้ Clustering ในการหากลุ่มของผู้ใช้ (User Clustering) และกลุ่มของสินค้า (Item Clustering) จากนั้นปรับปรุง Matrix Factorization (MF) โดยให้โมเดลเรียนรู้ความสนใจในระดับกลุ่มผู้และกลุ่มสินค้า สมการทำนายคะแนนเป็นไปดังตามสมการที่ 2.18

$$\hat{y}_{ui} = T_{\alpha}(q_i, q_{C_i})^T T_{\alpha}(p_u, p_{C_u}) + T_{\beta}(b_u, b_{C_u}) + T_{\beta}(b_i, b_{C_i}) \qquad 2.18$$

ซึ่งผลลัพธ์คือความแม่นยำสูงขึ้น แต่สูงขึ้นเพียงเล็กน้อยเท่านั้น RMSE ลดลงน้อยมาก (จาก 0.9025 เหลือ 0.9020) ความซับซ้อนของโมเดลที่เพิ่มสูงขึ้นมาก เนื่องจากมี Hyperparameter เพิ่มขึ้นในจำนวนที่มาก และงานดังกล่าวละเลยการประเมินผลแบบ Top-N เช่น NDCG@K ซึ่งสะท้อนความเป็นจริงมากกว่าในงานที่เกี่ยวข้องกับระบบแนะนำสินค้า (Roy et al., 2022)

สิ่งที่สามารถนำไปต่อยอดได้จากงานดังกล่าวได้คือ เราสามารถใช้ Clustering ในการหากลุ่มของผู้ใช้และสินค้าได้โดยไม่ต้องใช้ข้อมูลคุณลักษณะ (Feature) เพิ่มเติมเลย แต่ใช้เมทริกซ์ผู้ใช้และสินค้า (User-Item Matrix) ในการดึงคุณลักษณะออกมาแทน

2.2.3 การใช้เทคนิคปัจจัยร่วม (Co-Factor) เพื่อเพิ่มประสิทธิภาพ

งานวิจัยของ Liang et al., (2016) นำเสนอโมเดล Co-Factor Matrix Factorization (Co-Factor MF) มุ่งเน้นการเพิ่มความแม่นยำของระบบ โดยมีแนวคิดที่ว่าโมเดล MF แบบมาตรฐาน (เช่น WMF) มักจะมีปัญหากับสินค้าที่ไม่ได้เป็นที่นิยมมากนัก (Rare Item) เนื่องจากมีข้อมูลผู้ใช้น้อยทำให้โมเดลเรียนรู้ปัจจัยแฝง (Latent Factors) ในกลุ่มสินค้าดังกล่าวไม่ดี

งานวิจัยเสนอให้เรียนรู้ปัจจัยแฝงร่วม (Co-Factor) โดยการเรียนรู้ของโมเดลจะมีส่วนประกอบที่ต้องเรียนรู้ 2 ส่วน ประกอบด้วย

1. User-Item Matrix: บอกว่าผู้ใช้คนไหนให้คะแนนหรือคลิกสินค้าชิ้นไหน เป็น User-Item Matrix ปกติที่ใช้กันทั่วไปในระบบแนะนำสินค้าแบบ Collaborative Filtering (CF)
2. Item-Item Co-occurrence Matrix: บอกว่ามีผู้ใช้กี่คนที่ใช้สินค้า A และสินค้า B ทั้งคู่

แนวคิดนี้ได้รับแรงบันดาลใจมาจากโมเดล Word2vec ในงานทางด้านภาษาศาสตร์ (NLP) ที่มองว่าคำที่มักเกิดในบริบทใกล้เคียงกันจะมีความหมายคล้ายกัน ในที่นี้ก็คือ สินค้าที่ถูกซื้อพร้อมกันโดยผู้ใช้หลายๆ คน ก็ควรจะมีความคล้ายคลึงกัน โดยสมการ Item-Item Co-occurrence Matrix ที่จะให้โมเดลเรียนรู้เพิ่ม จะถูกคำนวณมาก่อน (Pre-computed) จากสมการการเกิดร่วมกัน เรียกว่า Shifted Positive Pointwise Mutual Information (SPPMI) ตามสมการที่ 2.19

$$PMI(i,j) = \log \frac{\#(i,j) \cdot D}{\#(i) \cdot \#(j)} \qquad 2.19$$

เมื่อได้ค่า PMI แล้ว จากนั้นทำการแก้ไขฟังก์ชันเป้าหมาย (Objective Function) เพื่อให้โมเดลเรียนรู้ทั้ง User-Item Matrix และ Item-Item Co-occurrence Matrix ดังสมการที่ 2.20 วิธีนี้จึงเปรียบเสมือนการใช้ข้อมูล Co-occurrence เป็นตัวกำกับ (Regularization) เพื่อบังคับให้ Item Factor ที่ได้มีคุณภาพและสมเหตุสมผลมากขึ้น

$$\mathcal{L}_{co} = \sum_{u,i} c_{ui}(y_{ui} - p_u^\top q_i)^2 + \sum_{m_{ij} \neq 0} (m_{ij} - q_i^\top \gamma_j - w_i - c_j)^2 \qquad 2.20$$

โมเดล Co-Factor MF มีการประยุกต์ใช้ Co-occurrence Matrix เพื่อเพิ่มความแม่นยำ โดยเฉพาะกับ Rare Item ทำให้ความแม่นยำของระบบที่ใช้ Co-Factor MF เพิ่มขึ้น โดยแนวคิดดังกล่าวมีความใกล้เคียงกับที่งานวิจัยจะเสนอ แต่มีจุดที่เราสามารถต่อยอดได้ 2 จุด คือ

1. การใช้ Co-occurrence Matrix โดยในธุรกรรม (Transaction) มีเพียงสินค้าอย่างเดียวนั้นจะไม่สามารถรับมือกับ User Cold Start และ Item Cold Start ได้ เพราะหากผู้ใช้หรือสินค้าใดไม่มีคะแนน (Ratings) ที่ผู้ใช้ให้เลย โมเดลก็อาจจะเรียนรู้ได้ไม่ดีนัก (เช่นเดียวกับ Bias MF) หากมีการปรับปรุง Transaction ให้มีกลุ่มของผู้ใช้ (เพื่อแก้ User Cold Start) และกลุ่มของสินค้า (เพื่อแก้ Item Cold Start) ก็จะทำให้รับมือกับปัญหา Cold Start ได้ดีขึ้นไปอีก
2. การให้โมเดลเรียนรู้แบบ Co-Factor ซึ่งจะทำให้โมเดลเก่งทั้งการทำนายตามเป้าหมายหลัก (First Objective) และเป้าหมายรอง (Second Objective) เป็นจุดที่น่าสนใจ และสามารถนำไปใช้เป็นวิธีการปรับปรุงฟังก์ชันเป้าหมาย (Objective Function) ของงาน CARMS MF ที่จะเสนอได้

2.2.4 การใช้กฎความสัมพันธ์ (Association Rule Mining) เพื่อเพิ่มประสิทธิภาพ

งานวิจัยของ Alsalama (2013) เสนอระบบแบบผสม (Hybrid Method) ที่ผสมผสานระหว่าง Association Rules Mining (ARM) กับแนวทาง Content-Based Filtering (CBF) เพื่อเพิ่มความสามารถในการทำนายผลของระบบแนะนำสินค้า

จุดเด่นของงานนี้คือการพยายามใช้ประโยชน์จากข้อมูลทั้งหมดที่ผู้ใช้ให้มาโดยระบุว่าระบบส่วนใหญ่จะเน้นไปที่สินค้าที่ได้คะแนนสูง (Favorite) เพื่อแนะนำของที่คล้ายกัน แต่งานวิจัยนี้ตั้งสมมติฐานว่าแม้แต่สินค้าที่ผู้ใช้ไม่ชอบ (Non-Favorite) ก็ยังสามารถให้ข้อมูลที่เป็นประโยชน์ได้ โมเดลนี้แบ่งการทำงานออกเป็น 2 ส่วนหลัก คือ

1. Rule Generation:
    - ใช้ Apriori กับทุกสินค้าที่ผู้ใช้แต่ละคนเคยใช้งานเพื่อค้นหากฎความสัมพันธ์ของสินค้าแต่ละชิ้น เช่น $item_1 \rightarrow item_3$ หมายความว่าคนที่ดู $item_1$ มักจะดู $item_3$ ด้วย
2. Item Distinction:
    - สำหรับผู้ใช้เป้าหมาย ระบบจะแบ่งสินค้าที่ผู้ใช้เคยให้คะแนนไว้ โดยแบ่งเป็น 2 กลุ่มคือ Favorite Items: สินค้าที่ผู้ใช้ให้คะแนนสูง (ผู้ใช้ให้คะแนนมากกว่าหรือเท่ากับ 3) และ Non-Favorite Items: สินค้าที่ผู้ใช้ให้คะแนนต่ำ (ผู้ใช้ให้คะแนนน้อยกว่า 3)
3. Recommendation:
    - Favorite: ระบบจะใช้ ARM ที่สร้างไว้ หากผู้ใช้ชอบ $item_1$ (ซึ่งอยู่ในกลุ่ม Favorite) และมีกฎว่า $item_1 \rightarrow item_3$ ระบบก็จะแนะนำ $item_3$ ให้กับผู้ใช้
    - Non-Favorite: ระบบจะใช้แนวทาง Content-Based โดยจะหาสินค้าที่มีเนื้อหา (เช่น มหวดหมู่ของภาพยนต์) คล้ายกับสินค้าที่ผู้ใช้ไม่ชอบ และแนะนำสินค้าเหล่านั้น (แนวคิดคือผู้ใช้อาจจะไม่ชอบหนัง Action เรื่องหนึ่ง แต่ก็อาจจะชอบหนัง Action อีกเรื่องหนึ่งก็ได้)

งานวิจัยนี้ชี้ให้เราเห็นว่าการใช้ ARM ในการพัฒนาระบบแนะนำสินค้าสามารถเพิ่มความแม่นยำได้ เนื่องจากการตัดสินใจเลือกสินค้าของผู้ใช้มีลำดับขั้นตอนเหมือนกับแนวคิดของ ARM ซึ่งเป็นจุดที่สามารถต่อยอดได้ โดย CARMS MF จะประยุกต์ใช้ ARM ในแนวทางที่สามารถแก้ไขปัญหา Cold Start ได้

บทที่ 3

วิธีวิจัย

3.1 แบบจำลองที่ใช้ในการวิจัย

แบบจำลองการแยกตัวประกอบเมทริกซ์โดยใช้สัญญาณการจัดกลุ่มและการหากฎความสัมพันธ์ (Clustering Association Rule Mining Signal Matrix Factorization – CARMS MF) ถือเป็นแบบจำลองแบบผสม (Hybrid Method) ซึ่งจะใช้อัลกอริทึม (Algorithm) ทั้งการจัดกลุ่ม (Clustering) การหากฎความสัมพันธ์ (Association Rule Mining) และการแยกตัวประกอบเมทริกซ์ (Matrix Factorization) ร่วมกันเพื่อทำนายสินค้าที่ผู้ใช้จะเลือกในอนาคต โดยมีขั้นตอนในการคำนวณทั้งสิ้น 5 ขั้นตอน ประกอบด้วย

3.1.1 การจัดกลุ่มข้อมูล (Clustering)

จากที่ทบทวนในงานวิจัยของ (Liang et al., 2016) หากพิจารณาถึงปัญหาของระบบแนะนำสินค้าโดยฉพาะกับโมเดล MF และ Bias MF คือ User Cold Start และ Item Cold Start หากเราสามารถปรับปรุงธุรกรรม (Transaction) ก่อนจะหากฎความสัมพันธ์ เราจะมีกฎความสัมพันธ์ที่สามารถแนะนำสินค้าให้กับ Cold Start User หรือแนะนำสินค้าที่เป็น Item Cold Start ได้

เพื่อปรับปรุงธุรกรรม สามารถใช้เมทริกซ์ผู้ใช้และสินค้า (Y) แบ่งกลุ่มของผู้ใช้และกลุ่มของสินค้าโดยใช้ K-Means (Mirbakhsh & Ling, 2015)

แบ่งผู้ใช้ออกเป็น K_u กลุ่ม โดยใช้ Feature จากแถวของ Y

$$\mathcal{C}^U = \{C_1^U, ..., C_{K_u}^U\} = \arg\min_{\mathcal{C}} \sum_{k=1}^{K_u} \sum_{u \in C_k^U} || \, y_u - \mu_k^U \, ||^2 \qquad 3.1$$

โดยที่

- y_u คือเวกเตอร์ (Vector) การให้คะแนนของผู้ใช้ u เปรียบได้กับคุณลักษณะ (Feature) ของผู้ใช้
- μ_k^U คือจุดศูนย์กลาง (Centroid) ของกลุ่มผู้ใช้ (User Cluster) ที่ k
- ผลลัพธ์คือSet (Set) ของกลุ่มผู้ใช้ $C^U = (C_1^U, C_2^U, \dots C_{K_u}^U)$

เช่นเดียวกับผู้ใช้ แบ่งสินค้าออกเป็น K_i กลุ่ม โดยใช้ Feature จากแถวของ Y^T

$$\mathcal{C}^I = \{C_1^I, ..., C_{K_i}^I\} = \arg\min_{\mathcal{C}} \sum_{k=1}^{K_i} \sum_{i \in C_k^I} || \, y_i^T - \mu_k^I \, ||^2 \qquad 3.2$$

โดยที่

- y_i^T คือเวกเตอร์ (Vector) การให้คะแนนของผู้ใช้ u ในสินค้า i เปรียบได้กับคุณลักษณะ (Feature) ของสินค้า
- μ_k^I คือจุดศูนย์กลาง (Centroid) ของกลุ่มสินค้า (Item Cluster) ที่ k
- ผลลัพธ์คือSet (Set) ของกลุ่มสินค้า $C^I = (C_1^I, C_2^I, \dots C_{K_i}^I)$

โดยในขั้นตอนนี้จะต้องเลือกจำนวนกลุ่มผู้ใช้ (K_u)และจำนวนกลุ่มสินค้า (K_i) เป็น Hyperparameter ในสมการ CARMS MF และ CARMS Bias MF โดยผลลัพธ์จากสมการที่ 3.1 และ สมการที่ 3.2 จะได้กลุ่มของผู้ใช้เท่ากับ $C^U = (C_1^U, C_2^U, \dots C_{K_u}^U)$ และกลุ่มของสินค้าเท่ากับ $C^I = (C_1^I, C_2^I, \dots C_{K_i}^I)$ ซึ่งจะนำไปใช้ปรับปรุงธุรกรรมต่อไป

3.1.2 การสร้างรายการธุรกรรม

เนื่องจากข้อมูลตั้งต้นเป็นคะแนนความชอบ (Ratings) ซึ่งเป็นค่าต่อเนื่อง และอาจมีค่าไม่เท่ากันในแต่ละชุดข้อมูล เช่น ข้อมูล A มีค่าตั้งแต่ 1-5 แต่ข้อมูล B มีค่าความชอบตั้งแต่ 1-10 เพื่อระบุว่าผู้ใช้ชอบหรือไม่ชอบสินค้านั้นโดยสามารถใช้ได้กับทุกชุดข้อมูล จะใช้เกณฑ์เปอร์เซ็นไทล์ (Percentile Threshold) (Alsalama, 2013) ในการหาเกณฑ์ความชอบของผู้ใช้แต่ละราย แทนด้วย τ_u เป็น Hyperparameter ในสมการ CARMS MF และ CARMS Bias MF ตามสมการที่ 3.4 และสุดท้ายผู้วิจัยจึงทำการแปลงเป็นข้อมูลทวิภาค (Binary) ระบุสินค้าที่ผู้ใช้ชอบ แทนด้วย B ตามสมการที่ 3.5

$$\begin{aligned} min_u &= \min_{i \in I, y_{ui} > 0}(y_{ui}) \\ max_u &= \max_{i \in I, y_{ui} > 0}(y_{ui}) \end{aligned} \qquad 3.3$$

$$\tau_u = min_u + (max_u - min_u) \times \theta \qquad 3.4$$

$$B_{ui} = \begin{cases} 1, & \text{if } y_{ui} \geq \tau_u \\ 0, & \text{otherwise} \end{cases} \qquad 3.5$$

ทั้ง 3 กลุ่ม คือ (1) กลุ่มของผู้ใช้ (2) สินค้าที่ผู้ใช้ชอบ และ (3) กลุ่มของสินค้าที่ผู้ใช้ชอบ จะถูกนำมาทำการรวม (Union) จะได้ T'_u เป็นธุรกรรมของผู้ใช้ u ที่ปรับปรุงแล้ว (Augmented Transaction) ตามสมการที่ 3.6

$$T'_u = \{\text{user\_group}_{G_u(u)}\} \cup \{\text{item}_i \mid B_{ui} = 1\} \cup \{\text{item\_group}_{G_i(i)} \mid B_{ui} = 1\} \qquad 3.6$$

ซึ่งจะทำให้เงื่อนไขจาก ARM ไม่ได้มีเฉพาะเงื่อนไขของสินค้าอย่างเดียว แต่สามารถมีเงื่อนไขทั้งสินค้า กลุ่มของสินค้า และกลุ่มของผู้ใช้ ซึ่งจากสมมุติฐานของงานวิจัย เงื่อนไขที่มีกลุ่มของผู้ใช้หรือกลุ่มของสินค้าผสมอยู่ด้วย จะสามารถแก้ไขปัญหาทั้ง User Cold Start และ Item Cold Start ได้ โดยเฉพาะกับปัญหา User Cold Start เพราะผู้ใช้ที่ไม่เคยซื้อสินค้าใดเลย ก็ยังจะมีกลุ่มของผู้ใช้อยู่ และกฎที่เกิดจากกลุ่มของผู้ใช้ใหม่เหล่านี้ ก็จะเป็นกฎในลักณะของสินค้าชิ้นแรกๆ ที่ผู้ใช้ใหม่เลือกซื้อ

จากนั้นรวมธุรกรรมที่ปรับปรุงแล้ว (Augmented Transaction) ของผู้ใช้แต่ละคน (T'_u) เป็นฐานข้อมูลของธุรกรรมที่ปรับปรุงแล้ว (Augmented Transaction Database) ตามสมการที่ 3.7 และจะนำไปหากฎความสัมพันธ์ (ARM) ต่อไป

$$D = \{T_1', T_2', \ldots, T_N'\} \qquad 3.7$$

3.1.3 การสกัดสัญญาณของสินค้าชิ้นถัดไป

ในขั้นตอนนี้ เรานำฐานข้อมูลธุรกรรมที่สร้างขึ้นมาค้นหากฎความสัมพันธ์ที่ซ่อนอยู่ และคัดกรองเฉพาะกฎที่สามารถนำไปสร้างเป็นสัญญาณ (Signal) สำหรับการแนะนำสินค้าได้ โดยกระบวนการนี้แบ่งออกเป็น 3 ขั้นตอนย่อย ดังนี้

1. Frequent Itemset Generation (การหาชุดรายการที่ปรากฏบ่อย)

ใช้วิธี FP-Growth เพื่อค้นหาชุดของสินค้า กลุ่มของสินค้า หรือกลุ่มของผู้ใช้ ที่เกิดขึ้นบ่อย (Frequent Set) ในธุรกรรมที่ปรับปรุงแล้ว (Augmented Transaction) ขั้นต่ำที่กำหนด โดยเริ่มจากแบ่ง Subset ของ T'_u แทนแต่ละสมาชิกด้วย X โดยที่ $X \subseteq T'_u$ ตัวอย่างเช่น หาก $T'_u = \{Item_1, Item\_group_1, User\_group_1\}$ ดังนั้นจะมี Subset เท่ากับ 7 รูปแบบ ประกอบด้วย

- สมาชิกแบบเดี่ยว ทั้งหมด 3 Set
  $\{Item_1\}, \{Item\_group_1\}, \{User\_group_1\}$
- สมาชิกแบบคู่ ทั้งหมด 3 Set
  $\{Item_1, Item\_group_1\}, \{Item_1, User\_group_1\}, \{Item\_group_1, User\_group_1\}$
- สมาชิกแบบเต็ม ทั้งหมด 1 Set
  $\{Item_1, Item\_group_1, User\_group_1\}$

จากนั้นคิด Support ซึ่งเป็นค่าที่บอกว่า X เกิดขึ้นบ่อยแค่ไหนในฐานข้อมูลทั้งหมด โดยที่ D คือ จำนวนธุรกรรมทั้งหมด (หรือเท่ากับจำนวนผู้ใช้ทั้งหมด เนื่องจากกำหนดให้ผู้ใช้ 1 คน มี 1 ธุรกรรม) โดยค่าสนับสนุน (Support) ของ X ตามสมการที่ 3.8

$$supp(X) = \frac{|\{T_u \in D | \ X \subseteq T_u\}|}{|D|} \qquad 3.8$$

จากนั้นเราจะคัดเลือกเฉพาะชุดรายการที่ผ่านเกณฑ์ $min\_support$ ซึ่งจะเป็น Hyperparameter ที่ต้องเลือกใน CARMS MF และ CARMS Bias MF ตามสมการที่ 3.9

$$supp(X) \geq min\_support \qquad 3.9$$

2. Rule Generation (การสร้างกฎความสัมพันธ์)

จาก Frequent Set ที่ได้ เราจะสร้างกฎความสัมพันธ์ในรูปแบบ $A \rightarrow B$ (เหตุ → ผล) โดยที่ A คือ Antecedent (สิ่งที่ผู้ใช้มีหรือเป็น) B คือ Consequent (สิ่งที่น่าจะตามมา หรือสิ่งที่น่าจะสนใจ) โดยที่เหตุและผลต้องไม่ซ้ำกัน $A \cap B = \emptyset$

ตัวเศษหรือเหตุ (Antecedent) จะบอกถึงความบ่อยที่เจอความสัมพันธ์ทั้ง A และ B (ทั้งคู่) ตัวส่วนหรือผล (Consequent) คือค่าจะบอกว่าความบ่อยที่จะเจอตัวหน้า (A) ก่อนเท่าไหร่ ซึ่งคือความน่าจะเป็นแบบมีเงื่อนไข (Conditional Probability) ว่า ถ้าเกิด A แล้ว มีโอกาสที่จะเกิด A คู่กับ B เท่าไหร่ เทียบเท่ากับหลักการความน่าจะเป็นแบบมีเงื่อนไขตามหลักการของ Bayes ตาม 3.10

$$conf(A \rightarrow B) = P(B \mid A) = \frac{supp(A \cup B)}{supp(A)} \qquad 3.10$$

จากนั้นเราจะคัดเลือกเฉพาะชุดรายการที่ผ่านเกณฑ์ $min\_confidence$ ซึ่งจะเป็น Hyperparameter ที่ต้องเลือกใน CARMS MF และ CARMS Bias MF ตามสมการ 3.11

$$conf(A \rightarrow B) \geq min\_confidence \tag{3.11}$$

3.1.4 การสร้างเมทริกซ์สัญญาณสินค้าชิ้นถัดไป (Signal Matrix)

เพื่อให้กฎที่ได้นำไปใช้ประโยชน์ได้ เนื่องจากเรามีการปรับปรุงมาใช้ธุรกรรมที่ปรับปรุงแล้ว (Augmented Transaction) ทำให้ต้องกำหนดข้อจำกัด (Constraint) ให้กับส่วนผลลัพธ์ (Consequent) ในบางกฎ เช่น คนที่ชอบสินค้า i น่าจะเป็นคนกลุ่ม k ($Item_i \rightarrow user\_group_k$) ออกก่อนการสร้างสัญญาณ เนื่องจากกฎลักษณะนี้ไม่มีประโยชน์ในการแนะนำสินค้า

ผลลัพธ์สุดท้าย (R) จะได้ Set ของกฎทั้งหมดที่ผ่าน Confidence ขึ้นต่ำ และจะไม่มี $user\_group$ ใดๆ ปนอยู่เลย ตามสมการที่ 3.12

$$R = \{(A \rightarrow B) | \; conf(A \rightarrow B) \geq min\_confidence \wedge B \cap C^I = \varnothing\} \tag{3.12}$$

ค่าสัญญาณ S_{ui} จะถูกคำนวณโดยการสะสม (Cumulative) ค่า Confidence ของกฎที่เกี่ยวข้องทั้งหมด โดยที่ $\Pi(.)$ คือ Indicator Function ที่มีค่าเป็น 1 เมื่อเงื่อนไขเป็นจริง และ 0 เมื่อเป็นเท็จ หมายความว่า หากผู้ใช้ u อยู่ในธุรกรรม T'_u และสินค้า i หรือกลุ่มของสินค้า i อยู่ในธุรกรรม T'_u จะนำค่า Confidence ของมารวมกัน ตามสมการที่ 3.13

$$S_{ui} = \sum_{r \in \mathcal{R}} \mathbb{I}(user \in T_u) \cdot \mathbb{I}(i \in T_u \vee D_u \in T_u') \cdot Conf(A \rightarrow B) \tag{3.13}$$

และสุดท้ายเพื่อลดผลกระทบของค่าสัญญาณที่อาจสูงเกินไป (Outliers) และส่งผลต่อขนาดของ Gradient ใน Gradient Descent ผู้วิจัยจึงทำการปรับค่าด้วยฟังก์ชัน Logarithm ซึ่งมีสูตรตามสมการที่ 3.14 และผลลัพธ์สุดท้ายคือ $\hat{S}_{ui}$ ซึ่งเป็นตัวแทนของสัญญาณ CARMS

$$\hat{S}_{ui} = \ln(1 + S_{ui}) \tag{3.14}$$

3.1.5 การแก้ไขสมการเป้าหมายด้วยเทคนิคเป้าหมายร่วม (Co-Objective)

เมื่อได้สัญญาณ (Signals) ของสินค้าชิ้นถัดไปแล้ว ขั้นตอนต่อไปคือการแก้ไขฟังก์ชันเป้าหมาย (Objective Function) เพื่อปรับให้โมเดลสามารถเรียนรู้ได้ทั้งข้อมูลที่มาจาก (User-Item Matrix (Y) และ Signal Matrix () ที่สร้างจาก CARMS จะใช้วิธีการปรับปรุงสมการเป้าหมายด้วยวิธีที่เรียกว่า Co Factorization อ้างอิงจากงานของ Liang et al., 2016 หากนำสัญญาณไปใช้กับโมเดล MF ดั้งเดิม จะได้ Objective Function ตามสมการที่ 3.15

$$\min_{p,q,r} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - q_i^T p_u)^2 + \delta(s_{ui} - r_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2 + \| r_i \|^2) \qquad 3.15$$

นอกจากนั้น สัญญาณ CARMS ยังสามารถนำไปปรับใช้ได้กับโมเดล Bias MF ด้วย ซึ่งเมื่อนำไปใช้จะได้ Objective Function ของสมการ CARMS Bias MF ตามสมการที่ 3.16

$$\min_{p,q,r,\mu,b_i,b_u} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - q_i^T p_u)^2 + \delta(s_{ui} - r_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2 + \| r_i \|^2) \qquad 3.16$$

หากพิจารณาสมการที่ 3.9 และสมการที่ 3.10 จะเห็นว่ามีส่วนที่เพิ่มเข้ามา คือ $\delta(s_{ui} - r_i^T p_u)$ ส่วนดังกล่าวจะเป็นส่วนที่บังคับให้ p_u เรียนรู้ทั้ง User-Item Matrix (First Objective) และ Signal Matrix (Second Objective) ไปพร้อมๆ กันเพื่อทำให้พจน์ดังกล่าวมีค่าน้อยที่สุด เช่นเดียวกับพจน์แรกในสมการ

แต่เนื่องจากความสำคัญของพจน์ CARMS อาจมีความสำคัญไม่เท่ากันในแต่ละชุดข้อมูล อาทิ CARMS อาจให้ผลที่ดีกับชุดข้อมูลที่เกี่ยวข้องกับห้างสรรพสินค้า (Supermarket) แต่อาจไม่ได้มีผลมากกับชุดข้อมูลอื่น เช่น เพลง หรือภาพยนตร์ จึงต้องมีตัวปรับความสำคัญของพจน์ Signal (δ) ซึ่งเป็น Hyperparameter อีกตัวหนึ่งที่เพิ่มเข้ามาจากโมเดล MF และ Bias MF

3.1.6 การเรียนรู้ (Training)

เนื่องจากชุดข้อมูล (ซึ่งจะอธิบายต่อใน 3.2) เป็นข้อมูลที่มีการให้คะแนนแบบ Explicit Data ดังนั้นจึงเลือกใช้วิธีการแก้ไขปัญหา (Solver) โดยใช้ Stochastic Gradient Descent (SGD) (Koren et al., 2009) โดยจะเลือกใช้เป็น Mini-Bach Gradient Descent ซึ่งถือเป็น Gradient Descent ที่ดีที่สุด เนื่องจากรวมทั้งข้อดีของ Stochastic Gradient Descent (SGD) และ Full-Batch Gradient Descent เอาไว้ด้วยกัน โดยสมการที่จะปรับตาม Gradient จะเป็นไปตามสมการที่ 3.17 ถึง 3.19

$$p_u \leftarrow p_u + \eta(e_{ui} q_i - \lambda p_u) \qquad 3.17$$

$$q_i \leftarrow q_i + \eta(e_{ui} p_u - \lambda q_i) \qquad 3.18$$

$$r_i \leftarrow r_i + \eta(\delta e_{ui} p_u - \lambda r_i) \qquad 3.19$$

และในส่วนของโมเดล CARMS Bias MF จะมีการเพิ่มพารามิเตอร์ที่ต้องหาค่านอกเหนือจากสมการ CARMS MF โดยสมการที่จะปรับตาม Gradient จะเป็นไปตามสมการที่ 3.20 ถึง 3.25

$$p_u \leftarrow p_u + \eta(e_{ui} q_i - \lambda p_u) \qquad 3.20$$

$$q_i \leftarrow q_i + \eta(e_{ui} p_u - \lambda q_i) \qquad 3.21$$

$$r_i \leftarrow r_i + \eta(\gamma e_{ui} p_u - \lambda r_i) \qquad 3.22$$

$$b_u \leftarrow b_u + \eta(e_{ui}) \qquad 3.23$$

$$b_i \leftarrow b_i + \eta(e_{ui}) \qquad 3.24$$

$$\mu \leftarrow \mu + \eta(e_{ui}) \qquad 3.25$$

3.2 ข้อมูลที่ใช้

งานวิจัยนี้วัดประสิทธิภาพโมเดล CARMS-MF และ CARMS Bias MF ด้วยชุดข้อมูลมาตรฐาน MovieLens ซึ่งนิยมใช้เปรียบเทียบผลในงานศึกษาที่เกี่ยวข้องกับระบบแนะนำสินค้า (Recommendation System) โดยจากการทบทวนของ Roy et al., (2022) พบว่ามีงานวิจัยประมาณร้อยละ 20 ใช้ชุดข้อมูล MovieLen โดยแบ่งการทดสอบเป็น 3 ขนาด ได้แก่ MovieLens-Small, MovieLen-100K และ MovieLen-1M เพื่อให้ครอบคลุมปริมาณการโต้ตอบที่แตกต่างกัน รายละเอียดของข้อมูลแสดงตามตารางที่ 3.3

ตารางที่ 3.1: รายเอียดของชุดของข้อมูล MovieLen

ชุดข้อมูล	จำนวนผู้ใช้	จำนวนภาพยนต์	ความเบาบาง (Sparse)
MovieLens-Small	610	9,724	1.6%
MovieLen-100K	943	1,682	6.3%
MovieLen-1M	6,040	3,706	4.5%

บทที่ 4

ผลการวิจัยและอภิปรายผล

4.1 ผลลัพธ์ของแบบจำลอง CARMS MF

จากการทดสอบโมเดล CARMS MF เมื่อเทียบกับตัวเทียบ (Baseline) คือ MF พบว่าคะแนน NDCG@10 (All User) ซึ่งเป็นตัวชี้วัดหลักที่จะใช้ในงานศึกษาครั้งนี้ เพิ่มขึ้นทุกชุดข้อมูล โดย MovieLen-Small เพิ่มจาก 21% เป็น 29% (เพิ่มขึ้น 8%) MovieLen-100K เพิ่มจาก 21% เป็น 27% (เพิ่มขึ้น 6%) และ MovieLen-1M เพิ่มจาก 21% เป็น 29% (เพิ่มขึ้น 8%)

ค่าที่สนใจอีกตัวเลขหนึ่งคือ NDCG@10 จากลูกค้าใหม่ (Cold Start) โดย MovieLen-Small เพิ่มจาก 22% เป็น 32% (เพิ่มขึ้น 10%) MovieLen-100K เพิ่มจาก 25% เป็น 34% (เพิ่มขึ้น 9%) และ MovieLen-1M เพิ่มจาก 22% เป็น 35% (เพิ่มขึ้น 13%) ตัวเลขเป็นไปตามสมมุติฐานของงานศึกษาว่าการเพิ่มสัญญาณแบบ CARMS ให้โมเดลได้เรียนรู้เพิ่มเติม จะเป็นส่วนเสริมให้ความแม่นยำในกลุ่มของลูกค้าใหม่เพิ่มขึ้น

แต่หากพิจารณาทั้ง NDCG@10 จากลูกค้าปัจจุบัน (Not Cold Start) ไม่ได้เปลี่ยนไปมากนัก แสดงว่าการเพิ่มสัญญาณแบบ CARMS อาจเป็น Noise สำหรับลูกค้าปัจจุบัน

นอกจากนี้หากพิจารณาตัวชี้วัดอื่นๆ ประกอบด้วย NDCG@5 (สนใจสินค้าเฉพาะอันดับต้นๆ) และ NDCG@20 (สนใจสินค้าในปริมาณที่มากขึ้น) พบว่าสูงกว่าเมื่อเทียบกับ MF ทั้งหมด แต่จุดที่น่าสนใจคือควาแม่นยำที่เพิ่มขึ้นจะลดลงเรื่อยๆ ตามปริมาณของสินค้าที่พิจารณา (5, 10 และ 20) อาจตีความได้ว่าการเพิ่มสัญญาณแบบ CARMS อาจบอกได้แค่สินค้าที่ผู้ใช้อาจจะเลือกในชิ้นถัดไปสั้นๆ ไม่ได้บอก Preference ในระยะยาว ทั้งนี้โมเดลทั้ง 2 โมเดล คือ MF และ CARMS MF ยังมีความแม่นยำต่ำกว่า Bias MF และ CARMS Bias MF ทั้งหมด

4.2 ผลลัพธ์ของแบบจำลอง CARMS Bias MF

จากการทดสอบโมเดล CARMS Bias MF เมื่อเทียบกับตัวเทียบ (Baseline) คือ Bias MF พบว่าคะแนน NDCG@10 (All User) ซึ่งเป็นตัวชี้วัดหลักที่จะใช้ในงานศึกษาครั้งนี้ เพิ่มขึ้นทุกชุดข้อมูล โดย MovieLen-Small เพิ่มจาก 32% เป็น 27% (เพิ่มขึ้น 5%) MovieLen-100K เพิ่มจาก 30% เป็น 32% (เพิ่มขึ้น 2%) และ MovieLen-1M เพิ่มจาก 29% เป็น 32% (เพิ่มขึ้น 3%)

ค่าที่สนใจอีกตัวเลขหนึ่งคือ NDCG@10 จากลูกค้าใหม่ (Cold Start) โดย MovieLen-Small เพิ่มจาก 35% เป็น 42% (เพิ่มขึ้น 10%) MovieLen-100K เพิ่มจาก 40% เป็น 42% (เพิ่มขึ้น 2%) และ MovieLen-1M เพิ่มจาก 35% เป็น 39% (เพิ่มขึ้น 4%) ตัวเลขเป็นไปตามสมมุติฐานของงานศึกษาว่าการเพิ่มสัญญาณแบบ CARMS ให้โมเดลได้เรียนรู้เพิ่มเติม จะเป็นส่วนเสริมให้ความแม่นยำในกลุ่มของลูกค้าใหม่เพิ่มขึ้น

แต่หากพิจารณาทั้ง NDCG@10 จากลูกค้าปัจจุบัน (Not Cold Start) ไม่ได้เปลี่ยนไปมากนัก เช่นเดียวกับ CARM MF แสดงว่าการเพิ่มสัญญาณแบบ CARMS อาจเป็น Noise สำหรับลูกค้าปัจจุบัน

นอกจากนี้หากพิจารณาตัวชี้วัดอื่นๆ ประกอบด้วย NDCG@5 (สนใจสินค้าเฉพาะอันดับต้นๆ) และ NDCG@20 (สนใจสินค้าในปริมาณที่มากขึ้น) พบว่าสูงกว่าเมื่อเทียบกับ MF ทั้งหมด แต่จุดที่น่าสนใจคือควาแม่นยำที่เพิ่มขึ้นจะลดลงเรื่อยๆ ตามปริมาณของสินค้าที่พิจารณา (5, 10 และ 20) อาจตีความได้ว่าการเพิ่มสัญญาณแบบ CARMS อาจบอกได้แค่สินค้าที่ผู้ใช้อาจจะเลือกในชิ้นถัดไปสั้นๆ ไม่ได้บอก Preference ในระยะยาว ทั้งนี้โมเดลทั้ง 2 โมเดล คือ MF และ CARMS MF จะมีความแม่นยำสูงกว่าถ้าเทียบกับ MF และ CARM MF

ตารางที่ 4.1: ตารางสรุปผล (NDCG@5)

Dataset	Model	Latent	Learning Rate	Lambda	Gamma	Epoch	NDCG@5 (All User)	NDCG@5 (Cold Start)	NDCG@5 (Not Cold Start)
MovieLen-Small	MF	100	0.001	0.1	-	150	23%	23%	21%
	CARMS MF	100	0.001	0.1	0.5	150	32%	35%	20%
	Bias MF	20	0.001	0.001	-	150	35%	38%	21%
	CARMS Bias MF	20	0.001	0.001	0.5	150	39%	44%	22%
MovieLen-100K	MF	50	0.1	0.1	-	50	23%	27%	14%
	CARMS MF	100	0.001	0.1	0.5	100	28%	36%	13%
	Bias MF	20	0.001	0.01	-	50	32%	42%	13%
	CARMS Bias MF	20	0.001	0.01	0.5	50	34%	45%	14%
MovieLen-1M	MF	50	0.001	0.1	-	100	22%	23%	18%
	CARMS MF	20	0.001	0.1	0.1	50	30%	36%	18%
	Bias MF	20	0.001	0.01	-	50	29%	34%	15%
	CARMS Bias MF	100	0.001	0.01	0.5	50	34%	41%	17%

ตารางที่ 4.2: ตารางสรุปผล (NDCG@10)

Dataset	Model	Latent	Learning Rate	Lambda	Gamma	Epoch	NDCG@5 (All User)	NDCG@5 (Cold Start)	NDCG@5 (Not Cold Start)
MovieLen-Small	**MF**	100	0.001	0.1	-	150	21%	22%	18%
	CARMS MF	100	0.001	0.1	0.5	150	29%	32%	18%
	Bias MF	20	0.001	0.001	-	150	32%	35%	20%
	CARMS Bias MF	20	0.001	0.001	0.5	150	37%	42%	20%
MovieLen-100K	**MF**	50	0.1	0.1	-	50	21%	25%	14%
	CARMS MF	100	0.001	0.1	0.5	100	27%	34%	13%
	Bias MF	20	0.001	0.01	-	50	30%	40%	13%
	CARMS Bias MF	20	0.001	0.01	0.5	50	32%	42%	13%
MovieLen-1M	**MF**	50	0.001	0.1	-	100	21%	22%	17%
	CARMS MF	20	0.001	0.1	0.1	50	29%	35%	17%
	Bias MF	20	0.001	0.01	-	50	29%	35%	15%
	CARMS Bias MF	100	0.001	0.01	0.5	50	32%	39%	16%

ตารางที่ 4.3: ตารางสรุปผล (NDCG@20)

Dataset	Model	Latent	Learning Rate	Lambda	Gamma	Epoch	NDCG@20 (All User)	NDCG@20 (Cold Start)	NDCG@20 (Not Cold Start)
MovieLen-Small	MF	100	0.001	0.1	-	150	19%	19%	17%
	CARMS MF	100	0.001	0.1	0.5	150	26%	29%	17%
	Bias MF	20	0.001	0.001	-	150	29%	32%	17%
	CARMS Bias MF	20	0.001	0.001	0.5	150	33%	38%	18%
MovieLen-100K	MF	50	0.1	0.1	-	50	19%	22%	14%
	CARMS MF	100	0.001	0.1	0.5	100	25%	32%	13%
	Bias MF	20	0.001	0.01	-	50	28%	37%	13%
	CARMS Bias MF	20	0.001	0.01	0.5	50	30%	39%	13%
MovieLen-1M	MF	50	0.001	0.1	-	100	20%	20%	17%
	CARMS MF	20	0.001	0.1	0.1	50	27%	33%	17%
	Bias MF	20	0.001	0.01	-	50	28%	34%	14%
	CARMS Bias MF	100	0.001	0.01	0.5	50	31%	37%	15%

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้เริ่มต้นด้วยปัญหาของระบบแนะนำสินค้า (Recommendation System) ในปัจจุบัน คือ ปัญหาข้อมูลที่เบาบาง (Data Sparsity) และปัญหา Cold Start คือการที่โมเดลไม่สามารถสร้างคำแนะนำได้ดีนัก เมื่อมีผู้ใช้ใหม่ (User Cold Start) หรือสินค้าใหม่ (Item Cold Start) เข้าสู่ระบบ เนื่องจากกลไกการเรียนรู้ของโมเดล MF และ Bias MF อาศัยเฉพาะข้อมูลการให้คะแนนที่มีอยู่จริง (Existing Ratings) ในการเรียนรู้ ปัญหาทั้งสองปัญหาส่งผลกระทบโดยตรงต่อความแม่นยำของโมเดลที่ลดลง โดยปัญหาทั้งสองประการยังคงเป็นช่องว่างในงานวิจัย (Research Gap) ในปัจจุบัน

จุดประสงค์ของวิทยานิพนธ์ฉบับนี้คือการพัฒนาโมเดลการแยกตัวประกอบเมทริกซ์โดยใช้สัญญาณการจัดกลุ่มและการหากฎความสัมพันธ์ (Clustering Association Rule Mining Signal Matrix Factorization – CARMS MF) ที่สามารถสกัดสัญญาณของสินค้าชิ้นถัดไป (Signal) จากข้อมูล โดยใช้การจัดกลุ่ม (Clustering) และการหากฎความสัมพันธ์ (Association Rule Mining) โดยมีสมมุติฐานว่าโมเดลที่นำเสนอจะมีความแม่นยำที่สูงขึ้นเมื่อเทียบกับโมเดลพื้นฐาน คือ MF และ Bias MF โดยเฉพาะอย่างยิ่งกับกลุ่มผู้ใช้งานใหม่ (Cold Start User) จะต้องมีความแม่นยำที่เพิ่มขึ้นอย่างมีนัยสำคัญ ทั้งนี้กระบวนการทั้งหมดจะไม่ใช้ข้อมูลคุณลักษณะ (Feature) ใดเพิ่มเติมเลย

จากการทบทวนวรรณกรรมพบว่า มีความพยายามในการใช้กฎความสัมพันธ์ ARM ในการพัฒนา MF ทั้งสิ้น 2 งาน ซึ่งแต่ละงานก็มีแนวทางที่วิทยานิพนธ์ฉบับนี้สามารถต่อยอดได้ คือ งานวิจัยของ Liang et al., (2016) ที่เสนอแนวคิด Co-Factor โดยให้โมเดลเรียนรู้สัญญาณจาก SPII ไปพร้อมกับข้อมูลคะแนนของผู้ใช้ แต่ก็ไม่ได้แก้ไขปัญหา Cold Start เนื่องจากโมเดลก็ไม่สามารถเรียนรู้กฎความสัมพันธ์จากลูกค้าใหม่หรือสินค้าใหม่ได้ หรืองานวิจัยของ Alsalama (2013) ที่ใช้ ARM เป็นหนึ่งในองค์ประกอบ (Module) หนึ่งของโครงสร้างโมเดล แต่ในส่วนของ Non-Favorite ก็ยังคงใช้ CBF ที่ใช้ข้อมูลคุณลักษณะ (Feature) เข้ามาเพิ่มเติมซึ่งยังไม่ตรงกับงานจุดมุ่งหมายของวิทยานิพนธ์ฉบับนี้

วิทยานิพนธ์ฉบับนี้เสนอโครงสร้างที่จะปรับปรุงธุรกรรม (Augmented Transaction) ของผู้ใช้แต่ละรายโดยวิธีการจัดกลุ่ม (Clustering) ซึ่งเสนอโดย Mirbakhsh & Ling (2015) สามารถสร้างกลุ่มของผู้ใช้และสินค้าได้โดยไม่ใช้ข้อมูลคุณลักษณะเพิ่มเติม และนำธุรกรรมที่ปรับปรุงแล้วไปทำ ARM ต่อเพื่อหากฎความสัมพันธ์ปรับปรุงฟังก์ชันเป้าหมายของ MF และ Bias MF ดั้งเดิมเป็นในลักษณะของ Co-Factor หรือการเรียนรู้ร่วมกันอ้างอิงจากงานวิจัยของ Liang et al., (2016)

ผลลัพธ์เมื่อเทียบกับ โมเดล Baseline แล้ว ทั้ง MF กับ CARMS MF และ Bias MF กับ CARMS Bias MF ถือว่าโมเดลที่วิทยานิพนธ์ฉบับนี้เสนอสามารถชนะโมเดลพื้นฐานได้ทั้ง 2 รูปแบบในทั้ง 3 ชุดข้อมูลที่นำมาทดสอบ คือ MovieLen-Small MovieLen-100k และ MovieLen-1M แสดงให้เห็นว่าโมเดลที่มีการเรียนรู้สัญญาณแบบ CARMS เข้าไปด้วย สามารถเพิ่มความแม่นยำของระบบแนะนำสินค้าได้ โดยเฉพาะกับลูกค้าใหม่ (Cols Start User) ซึ่งเป็นสิ่งที่วิทยานิพนธ์ฉบับนี้ตั้งสมมุติฐานไว้ข้างต้น

5.2 ข้อเสนอแนะสำหรับงานศึกษาในอนาคต

แม้โมเดลที่เสนอทั้ง CARMS MF และ CARMS Bias MF จะมีความแม่นยำมากกว่าโมเดลพื้นฐานทั้ง 2 รูปแบบ และได้ผลลัพธ์อื่นๆ อาทิ ความแม่นยำของกลุ่มผู้ใช้ใหม่ที่เพิ่มขึ้นอย่างมีนัยสำคัญตรงตามสมมุติฐานทุกประการ แต่ทั้งนี้ยังมีสิ่งที่สามารถต่อยอดหรือปรับปรุงจากงานที่ได้นำเสนอ 3 ประการ ดังนี้

1 ปัญหาเรื่องคอขวดของการหากฎความสัมพันธ์ (Bottleneck): ในการคำนวณของ FP-Growth: แม้ในงานศึกษา จะใช้เทคนิค FP-Growth ซึ่งเป็นหนึ่งในเทคนิค ARM ที่มีประสิทธิภาพสูง แต่งานศึกษายังคงมีคอขวด (Bottleneck) สำคัญที่ระยะเวลาในการประมวลผลในส่วนของการหากฎความสัมพันธ์ที่ช้า และทำให้ไม่สามารถหา Hyperparameter ที่ดีที่สุดในส่วนของสัญญาณของสินค้าชิ้นถัดไปได้ เช่น การปรับค่าสนับสนุน (Support) และค่าความเชื่อมั่น (Confidence) ดังนั้นในงานศึกษาในอนาคตอาจจะต้องมีการแก้ไขปัญหาดังกล่าวด้วยการใช้วิธีอื่นในการหากฎความสัมพันธ์ ซึ่งจะทำให้ CARMS MF มีความแม่นยำที่เพิ่มขึ้น
2 การต่อยอดในส่วนของการจัดกลุ่ม (Clustering): จากงานศึกษาของ งานวิจัยของ Mirbakhsh & Ling (2015) ผลจากอัลกอริทึมการจัดกลุ่มที่ต่างกันได้ผลลัพธ์ที่แตกต่างกัน ดังนั้นในงานศึกษาในอนาคตสามารถลองปรับวิธีการจัดกลุ่มข้อมูลได้ โดยอาจเปลี่ยนไปใช้โมเดลอื่นๆ เพื่อทดสอบผล เช่น โมเดลในตระกูลใช้ความหนาแน่น (Density-based) อาทิ DBSCAN หรืออาจเลือกใช้เทคนิคการจัดกลุ่มแบบอัตโนมัติ (Auto Clustering) ก็จะเป็นการลด Hyperparameter และลดความซับซ้อนในการใช้โมเดลลง

3 การต่อยอดสถาปัตยกรรม: ทั้ง CARMS MF และ CARMS Bias MF ยังคงอยู่ในสถาปัตยกรรมเดียวกัน คือสถาปัตยกรรมการแยกองค์ประกอบเมทริกซ์ (Matrix Factorization) ซึ่งในปัจจุบันสถาปัตยกรรมนี้ได้มีการต่อยอดโดยใช้การเรียนรู้เชิงลึก (Deep Learning) คือ Neural Collaborative Filtering (NCF) ซึ่งสามารถนำเอาสัญญาณ (Signal) ที่วิทยานิพนธ์ฉบับนี้เสนอ คือ Clustering Association Rule Mining Signal (CARMS) ไปประกอบกับสถาปัตยกรรมดังกล่าวได้

ภาคผนวก ก

ขั้นตอนการคำนวณ

1. สร้าง User-Item Matrix

กำหนดให้ชุดข้อมูลมีจำนวนผู้ใช้ 3 ราย (3 Rows) และมีจำนวนสินค้า 3 ชิ้น (3 Columns) ดังนั้น User-Item Matrix (Y) จะมีขนาดเท่ากับ 3x3 เป็นไปดังสมการที่ 1 แทนค่าว่างด้วย 0 ซึ่งหมายถึงสินค้าที่ผู้ใช้ยังไม่ได้ลงคะแนน (ซึ่ง MF จะไม่ได้เรียนรู้จาก 0 แต่ละพยายามหาค่าของ 0)

$$Y = \begin{bmatrix} 1 & 5 & 0 \\ 1 & 0 & 2 \\ 0 & 5 & 3 \end{bmatrix} \qquad 1$$

ส่วนประกอบของ User-Item Matrix ตามสมการที่ ก.1 คือ Rows แสดงถึงผู้ใช้ 3 คน ซึ่งสามารถแบ่งได้เช่น y_0 ตามสมการที่ 2 คือ $user\_0$ ลงคะแนนให้กับ $item\_0$ เท่ากับ 1 $item\_1$ เท่ากับ 5 และยังไม่เคยลงคะแนนให้ $item\_2$

$$y_0 = \begin{bmatrix} 1 & 5 & 0 \end{bmatrix} \qquad 2$$

ตามสมการที่ 3 คือ Columns ซึ่งแสดงถึงสินค้า 3 ชนิด ซึ่งสามารถแบ่งได้เช่น y_0^T คือ $item\_0$ คือสินค้าชิ้นที่ 1 ซึ่ง $user\_0$ ลงคะแนนเท่ากับ 1 $user\_1$ ลงคะแนนเท่ากับ 1 และ $user\_2$ ไม่เคยลงคะแนน

$$y_0^T = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \qquad 3$$

การมอง User-Item Matrix ทั้งในมุมของผู้ใช้และในมุมของสินค้า (Transpose) เปรียบเทียบได้กับคุณลักษณะของผู้ใช้ (User Feature) และคุณลักษณะของสินค้า (Item Feature) โดยไม่ต้องใช้ข้อมูลเพิ่มเติมเลย

2. จัดกลุ่มผู้ใช้และกลุ่มของสินค้า

ขั้นตอนถัดไปจะจัดกลุ่มของผู้ใช้โดยใช้ K-Mean จาก Rows ของ Y กำหนด $K_u = 3$ หมายถึง แบ่งกลุ่มของผู้ใช้ออกเป็น 2 กลุ่ม และจัดกลุ่มของสินค้าจาก Columns ของ Y กำหนด $K_i = 2$ หมายถึง แบ่งกลุ่มของสินค้าออกเป็น 2 กลุ่ม ซึ่งทั้งคู่เป็น Hyperparameter ใน CARMS MF

จากนั้นหาจุดศูนย์กลาง (Centroid) ที่เหมาะสมสำหรับแต่ละกลุ่มตามกระบวนการของ K-Mean ซึ่งผลลัพธ์จากกระบวนการดังกล่าวจะได้ฟังก์ชันที่ใช้จัดกลุ่มผู้ใช้ แทนด้วย G_u ตามสมการที่ 4 และฟังก์ชันที่ใช้จัดกลุ่มสินค้าแทนด้วย G_i ตามสมการที่ 5

$$G_u(y_u) = argmin_{k \in \{1,..,K_u=3\}} \| y_u - \mu_k^{(u)} \|^2 \quad 4$$

$$G_i(y_i^T) = argmin_{k \in \{1,..,K_u=2\}} \| y_i^T - \mu_k^{(u)} \|^2 \quad 5$$

เนื่องจากกำหนดกลุ่มของผู้ใช้เท่ากับ $K_u = 3$ ดังนั้นรายชื่อกลุ่มผู้ใช้จะเป็นไปดังสมการที่ 6 และเช่นกัน เนื่องจากกำหนดกลุ่มของสินค้าเท่ากับ $K_i = 2$ ดังนั้นรายชื่อกลุ่มสินค้าจะเป็นไปดังสมการที่ 7

$$C^U = \{user\_group\_0,\ user\_group\_1,\ user\_group\_2\} \quad 6$$

$$C^I = \{item\_group\_0,\ item\_group\_1\} \quad 7$$

เมื่อได้ฟังก์ชันทั้ง 2 ตามสมการที่ ก.6 และ ก.7 โดยฟังก์ชันทั้งสองจะใช้ Mapping ระหว่างคุณลักษณะของผู้ใช้กับกลุ่มของผู้ใช้และคุณลักษณะของสินค้ากับกลุ่มของสินค้า ตัวอย่างการใช้งานฟังก์ชันทั้งสองจะเป็นไปตามตารางที่ 1

ตารางที่ 1: ตัวอย่างการ Mapping กลุ่มผู้ใช้และกลุ่มสินค้า

Input	Calculation	Output
$y_0 = [1 \quad 5 \quad 0]$	$G_u(y_0 = [1 \quad 5 \quad 0])$	$user\_group\_2$
$y_0^T = [1 \quad 1 \quad 0]$	$G_i(y_0^T = [1 \quad 1 \quad 0])$	$item\_group\_1$

3. หา Favorite Item ของผู้ใช้แต่ละราย

Favorite Item ของผู้ใช้แต่ละราย คือสินค้าที่ผู้ใช้ลงคะแนนให้มากกว่า Personalized Threshold ของตนเอง ซึ่งกำหนดให้คือสินค้าที่ผู้ใช้ลงให้มากกว่า Percentile ที่ θ ของคะแนนที่ผู้ใช้ลงให้กับสินค้าทั้งหมด โดย θ เป็น Hyperparameter อีกตัวหนึ่งของ CARMS MF

$$\tau_u = min_u + (max_u - min_u) \times \theta \qquad 8$$

เริ่มจากการหาค่าสูงสุดและต่ำสุดในการให้คะแนนโดยไม่รวม 0 จากนั้นใช้สมการที่ 8 ในการคำนวณหา τ_u ซึ่งเป็นค่า Personalized Threshold ของแต่ละคน ตัวอย่างเช่น $user\_0$ มีค่า Min=1 และ Max = 5 ดังนั้น τ_0 จะมีค่าเท่ากับ 4.2 ตามตารางที่ 2

ตารางที่ 2: ตัวอย่างการคำนวณ Personalized Threshold ของผู้ใช้รายคน

Input	Min	Max	Calculation	Personalized Threshold
$y_0 = [1 \quad 5 \quad 0]$	1	5	1+(5-1)*0.8	$\tau_0 = 4.2$
$y_1 = [1 \quad 0 \quad 2]$	1	2	1+(2-1)*0.8	$\tau_2 = 1.8$
$y_2 = [0 \quad 5 \quad 3]$	3	5	3+(5-3)*0.8	$\tau_3 = 4.6$

เมื่อได้ค่าแล้ว จะนำไปกรองธุรกรรมของผู้ใช้แต่ละรายเพื่อหา Set ของ Favorite Item คือสินค้าที่ได้คะแนนจากผู้ใช้เกินกว่า Personalized Threshold โดยเริ่มจากสร้าง Favorite Binary แทนด้วย B เพื่อกรองว่าสินค้าใดเกิน (แทนด้วย 1) และไม่เกิน (แทนด้วย 0) ตามตารางที่ 3

ตารางที่ 3: ตัวอย่างการหา Favorite Binary ของผู้ใช้แต่ละราย

Input	Personalized Threshold	Favorite Binary
$y_0 = [1 \quad 5 \quad 0]$	$\tau_0 = 4.2$	$B_0 = [0 \quad 1 \quad 0]$
$y_1 = [1 \quad 0 \quad 2]$	$\tau_2 = 1.8$	$B_1 = [0 \quad 0 \quad 1]$
$y_2 = [0 \quad 5 \quad 3]$	$\tau_3 = 4.6$	$B_2 = [0 \quad 1 \quad 0]$

4. สร้าง Augmented Transaction

ตอนนี้ผู้ใช้แต่ละคนจะมี Set ส่วนตัวทั้งหมด 3 Set คือ A ซึ่งบอกกลุ่มของของผู้ใช้ (ใช้สมการที่ 6) Set ของสินค้าที่ชอบ (ที่คัดมาจาก Personalized Threshold ตามขั้นตอนที่ 3) และ Set ของกลุ่มของสินค้าที่ชอบ (ได้จากการนำ Set ของสินค้าที่ชอบไปผ่านสมการที่ 7) โดยแสดงตามตารางที่ 4

ตารางที่ 4: ตัวอย่างการหา Favorite Item ของผู้ใช้แต่ละราย

User	$\{\text{user\_group}_{G_u(u)}\}$	$\{\text{item}_i \mid B_{ui}=1\}$	$\{\text{item\_group}_{G_i(i)} \mid B_{ui}=1\}$
$user\_0$	$user\_group\_2$	$item\_1$	$item\_group\_0$
$user\_1$	$user\_group\_0$	$item\_2$	$item\_group\_1$
$user\_2$	$user\_group\_1$	$item\_1$	$item\_group\_0$

เมื่อได้ข้อมูลทั้งหมด ประกอบด้วย Set ของกลุ่มของผู้ใช้ Set ของสินค้าที่ชอบ และ Set ของกลุ่มของสินค้าที่ชอบ ทั้ง 3 Set จะนำมารวม Union กัน ตามสมการที่ 8

	$T_u' = \{\text{user\_group}_{G_u(u)}\} \cup \{\text{item}_i \mid B_{ui}=1\} \cup \{\text{item\_group}_{G_i(i)} \mid B_{ui}=1\}$	*8*

ซึ่งจะทำให้ Augmented Transaction (T') ของแต่ละผู้ใช้เป็นไปตามตารางที่ 5

ตารางที่ 5: ตัวอย่างการหา Favorite Item ของผู้ใช้แต่ละราย

User	**Augmented Transaction**
$user\_0$	$T_0' = \{user\_group\_2,\ item\_1,\ item\_group\_0\}$
$user\_1$	$T_1' = \{user\_group\_0,\ item\_group\_1,\ item\_2\}$
$user\_2$	$T_2' = \{user\_group\_1,\ item\_1,\ item\_group\_0\}$

ธุรกรรมที่รวมทั้ง 3 ส่วนเข้าไปแล้ว คือ เราจะเรียกว่าธุรกรรมที่ปรับปรุงแล้ว (Augmented Transaction) อยู่ใน Set $D = \{T_1', T_2', \ldots, T_N'\}$ ข้อสังเกตคือ ในการทำ ARM แบบปกติ ธุรกรรมจะมีแค่สินค้าที่ชอบ แต่หากใช้ Augmented Transaction ผู้ใช้ที่ไม่มีการโต้ตอบกับระบบเลย หรือสินค้าเป็นสินค้าใหม่ที่ไม่มีใครให้คะแนนเลย โมเดลก็ยังสามารถเรียนรู้และทำนายพฤติกรรมได้อยู่

5. สร้างกฎความสัมพันธ์จาก Augmented Transaction

จากนั้น Augmented Transaction จะถูกนำมาแบ่ง Subset ซึ่งจำนวน Subset ที่เป็นไปได้เท่ากับ $(N! + 1)$ ตัวอย่างเช่นหาก Augmented Transaction มีสมาชิกเท่ากับ 3 จะมี Subset ที่เป็นไปได้ 10 Subset และตัวที่ซ้ำกันจะถูกตัดออก จากตัวอย่างตารางที่ ก.5 เราจะมี Subset ทั้งสิ้น 30 Subset เมื่อตัดตัวที่ซ้ำกันออกแล้วจะเหลือ 18 Subset จากนั้นจะตัดเฉพาะที่มีค่า Support เกินกว่า $min\_support$ ซึ่งกำหนดให้เท่ากับ 0.5 เป็น Hyperparameter ของ CARMS MF ที่จะต้องเลือก จากตารางที่ 6 จะมี Subset ที่ผ่านเกณฑ์ทั้งสิ้น 3 Subset

ตารางที่ 6: Subset ที่ผ่าน Minimum Support

Subset	Support
$\{item\_group\_0\}$	0.6667
$\{item\_1, item\_group\_0\}$	0.6667
$\{item\_1\}$	0.6667

จากนั้นจะนำ Subset ที่ผ่าน $min\_support$ มาจับคู่ และหา Probability ที่แต่ละคู่จะเกิดขึ้นพร้อมกัน (B given A) โดยใช้ Support จากตารางที่ 6 เป็น Probability ในการคำนวณ และจะต้องมีการกำหนด $min\_confidence$ คือค่าขั้นต่ำของความน่าจะเป็นที่จะเกิดขึ้นพร้อมกันมากพอ (Strong Rule) ที่จะนำไปใช้ทำนายสินค้าหรือกลุ่มของสินค้าชิ้นถัดไป ทั้งนี้หากกฎอยู่ในรูปของ $\{X\} \rightarrow \{user\_group\}$ จะถูกกรองทิ้งอัตโนมัติเนื่องจากกฎลักษณะดังกล่าวไม่สามารถบอกถึงสินค้าชิ้นถัดไปได้ จึงไม่มีประโยชน์ในการใช้เป็น Signal ให้โมเดลเรียนรู้ โดยกฎทั้งหมดที่จะนำไปใช้ตามตารางที่ 7

ตารางที่ 7: เงื่อนไขที่ผ่านเงื่อนไข Minimum Confidence

# Rule	Antecedents	Consequents	Confidence
1	$item\_1$	$item\_group\_0$	1.0
2	$item\_group\_0$	$item\_1$	1.0

6. สร้าง Signal Matrix

จากนั้นนำเงื่อนไขจากตารางที่ 7 มา Mapping กับ Augmented Transaction ของผู้ใช้แต่ละราย ซึ่งจากตารางที่ 8 พบว่า มีผู้ใช้ 2 รายที่เข้าเงื่อนไข ประกอบด้วย

ตารางที่ 8: การ Mapping เงื่อนไขกับ Augmented Transaction

User	Augmented Transaction	Matched Rule
$user\_0$	$T_0' = \{user\_group\_2,\ item\_1,\ item\_group\_0\}$	• Rule #1 • Rule #2
$user\_1$	$T_1' = \{user\_group\_0,\ item\_group\_1,\ item\_2\}$	-
$user\_2$	$T_2' = \{user\_group\_1,\ item\_1,\ item\_group\_0\}$	• Rule #1 • Rule #2

จากนั้นนำคะแนนที่รวบรวมได้ทั้งหมดมารวมกัน (Cumulative) เพื่อสร้างเป็น Signal สำหรับสินค้าชิ้นถัดไปในผู้ใช้แต่ละราย ซึ่งจะได้ผลลัพธ์ตามสมการที่ 9 โดยสามารถตีความได้ว่า $user_0$ และ $user_1$ มีโอกาสที่จะซื้อสินค้าชิ้นถัดไปคือ $item\_1$

$$S = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix} \qquad 9$$

แต่ค่าดังกล่าวอาจมีค่ามีสูงเกินไป ซึ่งอาจเกิดจากการที่จำนวนผู้ใช้และจำนวนสินค้ามากขึ้น (เนื่องจากจะมีกฎเกิดขึ้นเป็นจำนวนมาก) หรือค่า Support และ Confidence มีค่าน้อยเกินไป เมื่อค่าของ Signal สูงมาก อาจไปรบกวนการเรียนรู้ของโมเดล หรือเกิดปัญหา Exploding Gradients ดังนั้นจึงมีการทำ Log Transformation จะได้ Signal ที่มีค่าลดลง ตามสมการที่ 10

$$S = \begin{bmatrix} 0 & 1.099 & 0 \\ 0 & 0 & 0 \\ 0 & 1.099 & 0 \end{bmatrix} \qquad 10$$

7. การปรับปรุง Objective Function

ในที่นี้จะใช้ CARMS MF ในการแสดงตัวอย่าง (ซึ่ง CARMS Bias MF ก็ทำในลักษณะเดียวกัน) โดยสมการ Objective Function ตามสมการที่ 11 คือ Objective Function ของ MF ดั้งเดิม

$$\min_{p,q} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2) \qquad 11$$

จากนั้นสมการที่ 10 จะถูกปรับปรุง โดยจากเดิมเดิมโมเดลจะเรียนรู้จากแค่ Y อย่างเดียว เมื่อปรับปรุงสมการแล้วจำเป็นต้องเรียนรู้จาก S ด้วย หรือเรียกว่า Co Factorization ตามสมการที่ 12 หากสังเกตจะเห็นว่า p_u จะต้องเรียนรู้จากทั้ง Y และ S ซึ่งอาจตีความโดยนัยได้ว่า P เป็น Preference ของผู้ใช้ต่อทั้ง User-Item Matrix และ Signal Matrix

$$\min_{p,q,r} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - q_i^T p_u)^2 + \delta(s_{ui} - r_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2 + \| r_i \|^2) \qquad 12$$

จากสมการที่ 12 จะเห็นได้ว่ามีพารามิเตอร์ที่ต้องเรียนรู้ (Learnable parameter) ทั้งสิ้น 3 ตัว คือ P, Q และ R โดยจะ Solve ด้วยวิธี Gradient Descent ซึ่งจะค่อยๆ ปรับพารามิเตอร์ที่ต้องเรียนรู้ตามทิศทางของ Gradient ตามสมการที่ 13 ถึง 15

$$p_u \leftarrow p_u + \eta(e_{ui} q_i - \lambda p_u) \qquad 13$$

$$q_i \leftarrow q_i + \eta(e_{ui} p_u - \lambda q_i) \qquad 14$$

$$r_i \leftarrow r_i + \eta(\delta e_{ui} p_u - \lambda r_i) \qquad 15$$

ทั้งนี้การ Solve ด้วย Gradient Descent แบ่งออกย่อยได้อีก 3 แบบ (1) Full-Batch Gradient Descent ที่จะแก้ทุกๆ Observation พร้อมกัน (2) Mini-Batch Gradient Descent ซึ่งจะแก้ในจำนวน Observation ที่กำหนดพร้อมกัน และ (3) Stochastic Gradient Descent ที่จะแก้ทีละ 1 Observationแต่ในตัวอย่างนี้จะใช้ Full-Batch Gradient Descent ในการยกตัวอย่างเพื่อความง่ายในการอธิบาย นอกจากนี้ยังมีตัวแปรที่ต้องกำหนดค่าอีกคือ $K = 2$ (ขนาดของ Latent) $Learning\_rate = 0.1$ (ความเร็วในการอัพเดท Gradient) $\lambda = 0.01$ (Regularization) $\delta = 0.1$ (Weight ของ CARM Signal) และ $max\_epoch = 50$ (จะ Update กี่รอบ) โดยขั้นตอนในการอัพเดท จะเป็นไปตามตารางที่ 9

ตารางที่ 9: การ Solve ด้วย Full-Batch Gradient Descent

Step	Explanation	Calculation
1	กำหนดค่าเริ่มต้นของ P, Q และ R ด้วยการสุ่มค่าที่มีค่าน้อยมาก	$P = \begin{bmatrix} 0.0077 & -0.0075 \\ 0.0135 & 0.0069 \\ -0.0033 & 0.0079 \end{bmatrix}$ $Q = \begin{bmatrix} 0.0028 & 0.0006 \\ 0.0052 & -0.0024 \\ -0.0005 & 0.0053 \end{bmatrix}$ $R = \begin{bmatrix} -1.00e-04 & 7.30e-03 \\ 1.30e-03 & 8.60e-03 \\ -1.02e-02 & -8.90e-03 \end{bmatrix}$
2.1	คำนวณค่า $\hat{Y}$ และ $\hat{S}$ จาก P, Q และ R ปัจจุบัน	$\hat{Y} = \begin{bmatrix} -0.e+00 & -0.e+00 & -0.e+00 \\ 0.e+00 & 1.e-04 & 0.e+00 \\ -0.e+00 & -0.e+00 & 0.e+00 \end{bmatrix}$ $\hat{S} = \begin{bmatrix} -1.e-04 & -1.e-04 & 1.e-04 \\ 0.e+00 & 1.e-04 & -2.e-04 \\ 1.e-04 & 1.e-04 & -0.e+00 \end{bmatrix}$
2.2	คำนวณ Error ตามสมการที่ 12	65.720901
2.3	คำนวณค่า Gradients (Net Direction) ตามสมการที่ 3.11 ถึง 3.13	$\nabla P = \begin{bmatrix} -0.0582 & 0.0208 \\ -0.0036 & -0.0222 \\ -0.0496 & -0.0096 \end{bmatrix}$ $\nabla Q = \begin{bmatrix} -0.0117 & 0.0013 \\ 0.1094 & -0.0044 \\ -0.0344 & -0.0751 \end{bmatrix}$ $\nabla R = \begin{bmatrix} 0.0e+00 & 0.0e+00 \\ 2.4e-03 & -1.0e-04 \\ 0.0e+00 & 0.0e+00 \end{bmatrix}$

Step	Explanation	Calculation
2.4	Update ค่าของ P, Q และ R ปัจจุบัน ทำให้ได้ค่า P, Q และ R ใหม่	$P = \begin{bmatrix} 0.0923 & -0.1075 \\ 0.1135 & 0.1069 \\ 0.0967 & 0.1079 \end{bmatrix}$ $Q = \begin{bmatrix} 0.1028 & -0.0994 \\ -0.0948 & 0.0976 \\ 0.0995 & 0.1053 \end{bmatrix}$ $R = \begin{bmatrix} 0.0988 & -0.0927 \\ -0.0987 & 0.1085 \\ 0.0898 & 0.0911 \end{bmatrix}$
2.5	ทำขั้นตอนที่ 2.1 ถึง 2.4 ซ้ำ จนถึงจำนวนรอบที่กำหนด	$epoch = max\_epoch$
3	ทำนาย $\hat{Y}$ จาก Latent P และ Q ที่เรียนรู้แล้ว (การทำนายเมื่อทำ 2.1-2.5 ครบ 50 รอบ)	$\hat{Y} = \begin{bmatrix} 0.8835 & 5.2471 & 2.7673 \\ 0.6583 & 3.3183 & 1.7582 \\ 0.8578 & 5.0418 & 2.6598 \end{bmatrix}$