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) มาเสริมกัน เพื่อดึงจุดแข็งของแต่ละฝ่ายมาใช้งานและกลบจุดอ่อนของกันและกัน ซึ่งโดยทั่วไปจะช่วยเพิ่มประสิทธิภาพและความแม่นยำของระบบโดยรวม

ในบรรดาเทคนิคการแนะนำทั้งหมด CF โดยเฉพาะอย่างยิ่ง Matrix Factorization (MF) ได้รับการยอมรับอย่างกว้างขวางว่าเป็นหนึ่งในโมเดลที่มีประสิทธิภาพสูง เนื่องจากความสามารถในการเรียนรู้ปัจจัยแฝง (Latent Factors) ของผู้ใช้และสินค้า (Aggarwal, 2016) ต่อมาได้มีการพัฒนาโมเดลโดยการเพิ่มความเอนเอียง (Bias) เข้าไปในโมเดล ซึ่งเพิ่มความแม่นยำในการทำนายได้ดียิ่งขึ้นไปอีก โมเดลดังกล่าวได้รับความสนใจและโด่งดังอย่างมากจากการแข่งขัน Netflix Prize (Koren et al., 2009) แต่ในการประยุกต์ใช้งานจริง โมเดลกับต้องเผชิญกับข้อจำกัดที่ส่งผลต่อความแม่นยำคือปัญหาความเบาบางของของมูล (Sparsity)

ปัญหาความเบาบางของข้อมูล คือ สภาวะที่ระบบมีข้อมูลการให้คะแนน (Rating) หรือการโต้ตอบ (Interaction) น้อยมาก เมื่อเทียบกับปริมาณผู้ใช้และสินค้าทั้งหมด อาทิ ชุดข้อมูล Amazon ซึ่งเป็นชุดข้อมูลมาตรฐานและเป็นชุดข้อมูลที่ใช้ในงานนี้ (และจะใช้ทดสอบในงานวิจัยครั้งนี้) พบว่ามีการให้คะแนนจริงอยู่ที่น้อยกว่า 1% ของช่องคะแนนทั้งหมด และส่วนที่เหลือไม่มีการให้คะแนนเกิดขึ้นจริง

และเนื่องจากโมเดลจำเป็นต้องใช้คะแนนที่เกิดขึ้นจริง (Existing Ratings) ในการเรียนรู้ เมื่อเกิดปัญหา Sparsity ทำให้ข้อมูลที่โมเดลจะต้องเรียนรู้มีไม่เพียงพอและส่งผลให้ความแม่นยำลดลง เพื่อแก้ไขปัญหา แนวทางที่นักวิจัยนิยมใช้กันมากที่สุดคือการพัฒนาโมเดลแบบผสม (Hybrid Methods) ซึ่งมักจะใช้วิธีการดึงเอาข้อมูลคุณลักษณะ (Feature) ของสินค้า อาทิ เพศ อายุ หรือชนิดของภาพยนตร์เข้ามาประกอบ แต่ก็ยังมีข้อจำกัดอีกประการหนึ่ง คือ ข้อมูลคุณลักษณะ (Feature) ไม่ได้มีในทุกชุดข้อมูล

จากปัญหาดังกล่าว จึงเกิดเป็นช่องว่างของงานวิจัย (Research Gap) คือ การพัฒนาระบบแนะนำสินค้าและบริการที่สามารถบรรเทาปัญหา Sparsity ได้ กล่าวคือโมเดลที่สามารถทำงานได้ดีกว่าโมเดลพื้นฐานโดยไม่ใช้ข้อมูลคุณลักษณะ (Feature) ใดเพิ่มเติมเลย

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

จากช่องว่างของงานวิจัย (Research Gap) วิทยานิพนธ์ฉบับนี้จึงมีวัตถุประสงค์ คือ

1. ออกแบบและพัฒนาสัญญาณของสินค้าชิ้นถัดไปโดยใช้เทคนิคการจัดกลุ่ม (Clustering) และการหากฎความสัมพันธ์ (Association Rule Mining: ARM) จากเมทริกซ์ผู้ใช้และสินค้า (User-Item Matrix) โดยไม่ใช้ข้อมูลคุณลักษณะอื่นประกอบเลย
2. ปรับปรุงฟังก์ชันเป้าหมาย (Objective Function) ให้สามารถใช้ประโยชน์จากสัญญาณที่ได้จาก CARMS เพื่อให้สัญญาณ CARMS เข้าไปช่วยปรับปรุงผลการแนะนำ
3. การเพิ่มสัญญาณ CARMS เข้ากับ Matrix Factorization ที่ผ่านการปรับ Objective Function สามารถเพิ่มความแม่นยำของระบบแนะนำได้จริงเมื่อเทียบกับโมเดลพื้นฐาน โดยจะพิจารณาความสามารถในการจัดอันดับรายการแนะนำให้ตรงกับสินค้าที่ผู้ใช้น่าจะเลือกถัดไป

1.3 สมมุติฐานของการศึกษา

สมมติฐานของสถาปัตยกรรม CARMS MF ประกอบไปด้วย

1. CARMS MF ที่ใช้สัญญาณแบบ CARMS ในการให้สัญญาณสินค้าชิ้นถัดไป จะให้ค่าความแม่นยำในการเรียงลำดับ (Ranking) ในภาพรวมสูงกว่าโมเดลพื้นฐาน ทั้ง NDCG@K และ HR@K
2. และเมื่อเปรียบเทียบการเรียงลำดับ (Ranking) ในขนาดที่แตกต่างกัน คือ K เท่ากับ 5, 10, 20, 50, 100 แล้ว CARMS MF ก็จะยังคงให้ผลลัพธ์ในการเรียงลำดับดีกว่าโมเดลพื้นฐานในทุกขนาดและทุกตัวชี้วัด
3. Hyperparameter ของ CARMS คือ ขนาดของ Cluster (k) ค่าน้อยที่สุดของ Support ($min\_support$) ค่าน้อยที่สุดของ Confidence ($min\_confidence$) และ Gamma (γ) มีความจำเป็นสำหรับการทำให้โมเดลมีความแม่นยำเพิ่มขึ้น

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

แนวคิดหลักของวิทยานิพนธ์ฉบับนี้ คือนำเสนอโมเดลการแยกตัวประกอบเมทริกซ์โดยใช้สัญญาณการจัดกลุ่มและการหากฎความสัมพันธ์ (Clustering Association Rule Mining Signal Matrix Factorization – CARMS MF) โดย CARMS MF เป็นโมเดลแบบผสม (Hybrid Method) ที่ใช้พื้นฐานของ Bias MF แต่จะมีความใหม่และแตกต่าง (Novelty) คือการเพิ่มสัญญาณของสินค้าชิ้นถัดไปให้แก่ผู้ใช้รายคนตามประวัติของสินค้าเพื่อให้โมเดลได้เรียนรู้สัญญาณนี้ไปพร้อมกับ User-Item Matrix โดยมีขั้นตอนในการคำนวณ 4 ขั้นตอน คือ

1. สกัดสินค้าที่ชอบ (Liked) ไม่ชอบ (Disliked) และไม่เคยลอง (No Rating) จาก User-Item Matrix และนำมาจัดกลุ่มผู้ใช้ (User Clustering)
2. ใช้การหากฎความสัมพันธ์ (Association Rule Mining) ในการหากฎในลักษณะ 1 ต่อ 1 ซึ่งมีทั้งกฎที่เป็นลักษณะของสินค้าต่อสินค้า (Item-to-Item) และกลุ่มผู้ใช้ต่อสินค้า (Cluster-to-Item)
3. นำกฎความสัมพันธ์ของแต่ละกลุ่มไปเทียบกับประวัติของผู้ใช้รายคน และนำมาสร้างเมทริกซ์สัญญาณสินค้าชิ้นถัดไป (CARMS) ให้กับผู้ใช้ทุกคน รวมถึงผู้ใช้ที่ไม่เคยให้คะแนนใดๆ เลย (Coldest Start)
4. ปรับปรุงสมการเป้าหมาย (Objective Function) ให้โมเดลสามารถเรียนรู้ทั้งจาก User-Item Matrix และใช้ Bayesian Personalized Ranking (BPR) ร่วมกับเมทริกซ์สัญญาณสินค้าชิ้นถัดไป (CARMS)

1.5 ขอบเขตของงานวิจัย

ขอบเขตของวิทยานิพนธ์และรวมถึงการวิจัยและการทดสอบผล มีดังนี้

1. ใช้ชุดข้อมูลสาธารณะ (Public Dataset) คือชุดข้อมูล Amazon ซึ่งเป็นชุดข้อมูลที่ใช้กันอย่างแพร่หลาย ทั้งสิ้น 5 หมวดหมู่เพื่อพิสูจน์ความแม่นยำของโมเดลในหลากหลายชุดข้อมูล
    - หมวดความงาม (Amazon Luxury Beauty – 5 Rating)
    - หมวดอุตสาหกรรม (Amazon Industry – 5 Rating)
    - หมวดของกินของใช้ในครัว (Amazon Pantry – 5 Rating)
    - หมวดเพลง (Amazon Music– 5 Rating)
    - หมวดเครื่องดนตรี (Amazon Instruments – 5 Rating)

2. การวัดผลจะมุ่งวัดผลจากคุณภาพของลำดับ (Ranking) มากกว่าการทำนายคะแนนให้ถูกต้องซึ่งนิยมกว่าในงานพัฒนาระบบแนะนำสินค้า (Roy et al., 2022) โดยใช้ Normalized Discounted Cumulative Gain (NDCG@K) เป็นตัวชี้วัดหลัก และประกอบกับดู Hit Rate (HR@K) ซึ่งเป็นตัวชี้วัดที่เข้าใจได้ง่ายกว่าเป็นตัวชี้วัดรอง

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 คือการแยกตัวประกอบเมทริกซ์ที่เพิ่มความเอนเอียง (Bias Matrix Factorization) การเรียนรู้การเรียงลำดับ (Bayesian Personalized Ranking) งานวิจัยที่มีผู้วิจัยแล้วในอดีต รวมถึงช่องว่างในการวิจัย (Research Gap)
3. บทที่ 3 วิธีวิจัย: อธิบายสถาปัตยกรรมของโมเดล CARMS MF โดยแบ่งเป็นการจัดกลุ่มข้อมูล (Clustering) การสร้างกฎความสัมพันธ์ (Association Rule Mining) การสร้างเมทริกซ์สัญญาณสินค้าชิ้นถัดไป และการปรับปรุงสมการเป้าหมาย (Objective Function) ด้วยเทคนิคเรียงลำดับ (Bayesian Pairwise Ranking) รวมไปถึงการนำเสนอลักษณะของชุดข้อมูลที่จะใช้ทดสอบ
4. บทที่ 4 ผลการวิจัยและอภิปรายผล: นำเสนอผลลัพธ์ที่ได้จากการเปรียบเทียบ CARMS MF กับโมเดลพื้นฐาน ในชุดข้อมูลที่แตกต่างกัน พร้อมทั้งสรุปอภิปรายผล
5. บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ: สรุปใจความสำคัญของงานวิจัย ข้อจำกัด และแนวทางในการต่อยอดงานวิจัยในอนาคต

บทที่ 2

ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ระบบแนะนำสินค้าและบริการ

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 อาทิ ผู้กำกับ นักแสดง หรือหมวดหมู่ของภาพยนต์ (Genre) กับการให้คะแนนของผู้ใช้ อาทิ หากผู้ใช้ A เคยให้คะแนนสูงกับภาพยนตร์ในหมวดหมู่ Sci-Fi ที่มีผู้กำกับคนหนึ่งๆ ระบบจะแนะนำภาพยนตร์มีคุณลักษณะคล้ายคลึงกัน มีขั้นตอนการทำงานคือ
    - การสร้างโปรไฟล์สินค้า (Item Profiling): สินค้าแต่ละชิ้นจะถูกแปลงเป็นเวกเตอร์ของคุณลักษณะ (Feature Vector) ซึ่งอาจใช้เทคนิคต่างๆ เช่น TF-IDF (Term Frequency-Inverse Document Frequency) หากคุณลักษณะเป็นข้อความ (เช่น คำบรรยายสินค้า)
    - การสร้างโปรไฟล์ผู้ใช้ (User Profiling): โปรไฟล์ผู้ใช้ $\boldsymbol{p_u}$ จะถูกสร้างโดยการรวม (Aggregation) เวกเตอร์คุณลักษณะของสินค้าที่ผู้ใช้ $\boldsymbol{u}$ เคยชอบเข้าด้วยกัน อาจใช้การหาค่าเฉลี่ยแบบถ่วงน้ำหนัก (Weighted Average) ในการรวมคะแนน ตามสมการที่ 2.1

$$p_u = \frac{\sum_{i \in I_u} w_{u,i} f_i}{\sum_{i \in I_u} w_{u,i}} \qquad 2.1$$

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

$$\text{score}(u, \text{new}) = \cos(\mathbf{p}_u, \mathbf{f}_{\text{new}}) = \frac{\mathbf{p}_u^{\top} \mathbf{f}_{\text{new}}}{\| \mathbf{p}_u \| \| \mathbf{f}_{\text{new}} \|} \qquad 2.2$$

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 โดยเฉพาะปัญหาความเบาบางของข้อมูล

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)

2.1.2 การแยกตัวประกอบเมทริกซ์ (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

$$y = q_i^T p_u \tag{2.3}$$

โมเดลจะเรียนรู้ P และ Q โดยการพยายามลดค่าความคลาดเคลื่อน (Error) บนชุดข้อมูลที่มีการลงคะแนน (Observed Ratings) พร้อมตัวกำกับ (Regularization) (λ) เพื่อป้องกันปัญหา Overfitting ผ่านฟังก์ชันเป้าหมาย ตามสมการที่ 2.4

$$\min_{p,q} \sum_{y>0} (y_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2) \tag{2.4}$$

2.1.3 การแยกตัวประกอบเมทริกซ์ที่เพิ่มความเอนเอียง (Bias Matrix Factorization)

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

แนวคิดนี้ตั้งอยู่บนข้อสังเกตที่ว่าคะแนนที่ผู้ใช้ให้ (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) ในขณะที่ผู้ใช้บางคนอาจเข้มงวดและมักจะให้คะแนนต่ำกว่าค่าเฉลี่ย
- Item Bias (ความเอนเอียงของสินค้า): โดยสินค้าบางชิ้น (เช่น ภาพยนตร์บางเรื่อง) ได้รับการยอมรับในวงกว้างว่าดี จึงมักจะได้รับคะแนนสูงกว่าค่าเฉลี่ย ในทางกลับกัน บางเรื่องก็อาจจะถูกมองว่าแย่กว่าเรื่องอื่น

Bias MF จึงพยายามระบุอิทธิพลของความเอนเอียงเหล่านี้ก่อน ดังนั้นสมการประมาณคะแนน ($\hat{y}$) ของ Bias MF จึงถูกปรับจากสมการที่ 2.3 เป็น 2.5

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

และสมการเป้าหมาย (Objective Function) จะถูกปรับไปเป็นตามสมการที่ 2.6

$$\min_{p,q,\mu,b_i,b_u} \sum_{y>0} (y_{ui} - \mu + b_i + b_u + q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2) \qquad 2.6$$

2.1.4 การเรียงลำดับแบบ Bayesian Personalized Ranking (BPR)

แนวคิดของ Bayesian Personalized Ranking (BPR) จะเปลี่ยนเป้าหมายจากการทำนายคะแนนเป็นการจัดอันดับให้ถูกต้องเป็นคู่ (Pairwise Ranking) (Rendle et al., 2009) โดย BPR จะเริ่มจากการพิจารณาเซ็ตเหตุการณ์ที่สังเกตได้ S (Observed positive) กล่าวคือ สำหรับผู้ใช้ u หากสินค้าชิ้นใดพบหรือสามารถได้ว่าสังเกตว่ามีพฤติกรรมเชิงบวก $(u,i) \in S$ เช่น มีการเข้าชมสินค้าเกิดขึ้น จะอนุมานว่าผู้ใช้จะชอบสินค้านั้นมากกว่าสินค้าที่ไม่ถูกสังเกต $(u,j) \notin S$ เช่น ถ้าผู้ใช้ดูภาพยนตร์ j แต่ไม่ดู i ก็จะอนุมานได้ว่า $j > i$

ชุดข้อมูลที่จะใช้ฝึกโมเดลจะไม่เป็นจุด (Pointwise) แต่เป็นคู่ (Pairwise) ซึ่งจะถูกเก็บใน Triplet (u,i,j) หมายถึงสินค้า i ที่สังเกตได้ (Positive) และสินค้า j ที่ไม่มีปฏิสัมพันธ์ (Negative) ของผู้ใช้ u

$$D_S = \{(u,i,j) \mid (u,i) \in S, (u,j) \notin S\} \qquad 2.7$$

เมื่อได้คู่เปรียบเทียบแล้ว ขั้นตอนถัดไปคือการสร้างความน่าจะเป็นความถูกต้องในการเรียงลำดับ โดยเริ่มจากการหาความแตกต่างของคะแนนของสินค้าที่เป็น Positive และ Negative ตามสมการที่ 2.8

$$x_{uij} = \hat{y}_{ui} - \hat{y}_{uj} \qquad 2.8$$

จากนั้นนำเอาความแตกต่างดังกล่าวแปลงเป็นความน่าจะเป็นที่โมเดลจะทำนายได้ถูกต้อง โดยใช้ Sigmoid Function ซึ่งจะทำให้ค่าอยู่ระหว่าง 0 กับ 1 ตามสมการที่ 2.9

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad 2.9$$

สุดท้ายจะนำค่าความน่าจะเป็นไปคำนวณใน BPR loss ตามสมการที่ 2.10 โดยตีความว่าหากโมเดลทำนายค่าความแตกต่างของคะแนนระหว่างสินค้าที่เป็น Positive และ Negative ได้มาก (สินค้าที่ควรได้คะแนนมากกว่า มีคะแนนสูงกว่าสินค้าที่ควรได้คะแนนน้อยกว่าอย่างชัดเจน) โอกาสที่ทำนายผิดจะมีค่าน้อย และในทางกลับกันหากโมเดลทำนายค่าความแตกต่างได้ต่ำ (สินค้าที่ควรได้คะแนนมากกว่าแต่มีคะแนนต่ำกว่าสินค้าที่ควรได้คะแนนน้อยกว่า) โอกาสทำนายผิดจะมีค่ามาก

$$\sum[-\log \sigma(\hat{y}_{ui} - \hat{y}_{uj})] \qquad 2.10$$

2.1.5 ทฤษฎีการจัดกลุ่ม (Clustering)

Clustering เป็นอีกแนวทางหนึ่งของการเรียนรู้แบบไม่มีผู้สอน (Unsupervised) เป้าหมายคือการจัดกลุ่มของข้อมูลซึ่งมีสมมุติฐานหลักว่าข้อมูลที่อยู่ในกลุ่ม (Cluster) เดียวกันคือจะมีความคล้ายคลึงกัน (Similarity)

K-Means เป็นหนึ่งในอัลกอริทึม (Algorithm) ในการจัดกลุ่มที่ได้รับความนิยมมากที่สุด (Sinaga & Yang, 2020) และจัดเป็นวิธีการจัดกลุ่มโดยใช้ระยะทาง (Distance Base) โดย K-Means จะแบ่งข้อมูลทั้งหมดออกเป็นกลุ่มย่อยโดยมีเป้าหมายที่จะแบ่งจำนวนกลุ่มให้ได้ทั้งหมด k กลุ่มโดยทำงานแบบวนซ้ำเพื่อลดค่าในฟังก์ชันเป้าหมาย Within-Cluster-Sum-of-Squares (WCSS) ตามสมการที่ 2.11 จนกระทั่งไม่มีการเปลี่ยนแปลง หรือเปลี่ยนแปลงน้อยมาก (Convergence)

$$WCSS = \sum_{j=1}^{k} \sum_{x_i \in C_j} || x_i - \mu_j ||^2 \qquad 2.11$$

ขั้นตอนในการคำนวณของ K-Means จะมีทั้งสิ้น 4 ขั้นตอน ดังนี้

1. Initialization (การเริ่มต้น): เลือกจุดศูนย์กลาง (Centroids) จำนวน k จุด โดยแต่ละจุดแทนด้วย μ_k
2. Assignment Step (การกำหนดสมาชิก)
    - คำนวณระยะห่างระหว่างจุดข้อมูล x_i แต่ละจุดกับ Centroid μ_k ทุกจุด (โดยทั่วไปมักใช้ฟังก์ชันระยะทางแบบ Euclidean)
    - กำหนดให้จุดข้อมูล x_i เป็นสมาชิกของ k ที่มีระยะทางใกล้ที่สุด (มีค่า $\|x_i - \mu_k\|^2$ น้อยที่สุด)
3. Update Step (การ Update จุด Centroid)
    - คำนวณตำแหน่ง Centroid ที่ k (μ_k) ใหม่ โดยการหาค่าเฉลี่ย (Mean) ของจุดข้อมูล x_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 คือเทคนิคที่ใช้ค้นหากฎความสัมพันธ์ (Association Rule Mining: ARM) ในข้อมูลขนาดใหญ่ (Big Data) เพื่อทำความเข้าใจพฤติกรรมผู้ใช้ ตัวอย่างเช่น การหาว่าผู้ใช้มักซื้อสินค้าอะไรคู่กัน เพื่อนำไปจัดโปรโมชั่นขายพ่วง (Bundle)

ผลลัพธ์ที่ได้จาก ARM ไม่ว่าจะเป็น Algorithms ใดก็ตามคือกฎ (Rule) ซึ่งถูกนิยามในรูปแบบ $A \rightarrow B$ โดยที่ A และ B คือซับเซต (Subset) ของสินค้าทั้งหมด $(A \subset I, B \subset I)$ และไม่มีสินค้าชิ้นใดร่วมกัน $(B \cap I = \emptyset)$ หากแปลเงื่อนไขเหล่านี้กับข้อมูลในโลกแห่งความเป็นจริง เช่น ห้างสรรพสินค้า (Supermarket) กฎ $A \rightarrow B$ หมายความว่าการซื้อ A (Antecedent หรือ กฎทางซ้ายมือ) ซึ่งอาจเป็นสินค้าชิ้นเดียวหรือสินค้าหลายชิ้น สื่อโดยนัยถึงการซื้อ B (Consequent หรือ กฎทางขวามือ) ซึ่งอาจเป็นสินค้าชิ้นเดียวหรือสินค้าหลายชิ้นก็ได้เช่นกัน

หลังจากที่ได้กฎแล้ว จำเป็นต้องมีกระบวนการประเมินคุณภาพเพื่อวัดความน่าสนใจของกฎ (ซึ่งอาจมีการเลือกบางกฎที่มีความสำคัญน้อยออกในขั้นตอนนี้) โดยตัวชี้วัดคุณภาพที่รู้จักกันดีที่สุดคือ Support และ Confidence และ Lift

1. Support (ค่าสนับสนุน): คือค่าที่จะบอกว่าเราพบเจอกฎทั้งที่เป็นสินค้าเดี่ยว เช่น A หรือที่มีทั้งเซต เช่นที่มีทั้ง A และ B เกิดขึ้นในธุรกรรมเดียวกันบ่อยแค่ไหน ตามสมการที่ 2.12 และ 2.13

$$sup(A \rightarrow B) = P(A \cap B) \qquad 2.12$$

$$sup(A) = P(A) \qquad 2.13$$

ซึ่งคำนวณตามสมการที่ 2.14 และ 2.15

$$P(A \cap B) = \#A \cap B / \#Transaction \qquad 2.14$$

$$P(A) = \#A / \#Transaction \qquad 2.15$$

2. Confidence (ค่าความเชื่อมั่น): หรือความแข็งแกร่งของกฎ (Strength of the rule) เป็นค่าที่บ่งชี้ว่าหากสินค้าหรือชุดของสินค้ามีการเกิดขึ้นอยู่แล้ว (Antecedent) จะมีสินค้าหรือชุดของสินค้าที่สนใจ เกิดขึ้นตามมาบ่อยเพียงใด (Consequent) ตามสมการที่ 2.16

$$conf(A \rightarrow B) = P(B \mid A) = \frac{supp(A \cap B)}{supp(A)} \qquad 2.16$$

ซึ่งจำนวน Transaction สามารถหักล้างกันได้พอดี ดังนั้นอาจเขียนให้อยู่ในรูปของความน่าจะเป็นแบบมีเงื่อนไข (Conditional probability) ได้ตามสมการที่ 2.17

$$conf(A \rightarrow B) = P(B \mid A) = \frac{P(A \cap B)}{P(A)} \qquad 2.17$$

3. Lift (ค่าแรงสนับสนุน): เป็นหนึ่งในตัววัดคุณภาพเพิ่มเติมที่ใช้ประเมินว่าการมีอยู่ของสินค้าหรือชุดของสินค้ามีปฏิสัมพันธ์กันในเชิงบวกหรือเชิงลบ หรือเป็นเพียงการเกิดขึ้นพร้อมกันโดยบังเอิญ (เนื่องจาก อาจเป็นสินค้ายอดนิยมที่เกิดขึ้นบ่อยโดยปกติอยู่แล้ว) สมการที่ 2.18 สามารถตีความได้ว่ามี A ทำให้โอกาสเจอ B เพิ่มขึ้นกี่เท่า เมื่อเทียบกับโอกาสปกติที่จะเจอ B อยู่แล้ว

$$lift(A \rightarrow B) = \frac{P(B \mid A)}{P(B)} = \frac{conf(A \rightarrow B)}{P(B)} \qquad 2.18$$

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 นั้นเป็นความสัมพันธ์ที่เกี่ยวข้องกันจริงมากกว่าเป็นเพียงแค่สินค้ายอดนิยม

เทคนิคที่นิยมใช้ทำ ARM มีทั้งสิ้น 3 รูปแบบด้วยกัน คือ (1) Apriori เป็นขั้นตอนวิธีที่ใช้ในการนำ ARM มาใช้งานในยุคแรกเริ่ม แต่อาจมีปัญหาในเรื่องการประมวลผล (2) FP-Growth (Frequent Pattern-Growth) ซึ่งเป็นวิธีคล้ายกับ Apriori ที่มีความไวในการประมวลผลมากกว่า ทั้งนี้กฎที่เล็กที่สุดของ Apriori และ FP-Growth คือกฎขนาด 1 ต่อ 1 หรืออาจเรียกว่าการวิเคราะห์สินค้าร่วม (Co-occurrence) ซึ่งจะมีความได้เปรียบเรื่องเวลาในการประมวลผล

2.1.7 Leave one out (LOO)

เพื่อประเมินความแม่นยำของระบบแนะนำสินค้าจึงมีความจำเป็นต้องแบ่งข้อมูลฝึกและข้อมูลสำหรับทดสอบอย่างเหมาะสม เพื่อหลีกเลี่ยงการรั่วไหลของข้อมูล (Data Leakage) ที่ทำให้ค่าความแม่นยำสูงเกินจริงและเสี่ยงเกิดการ Overfitting ดังนั้นการแบ่งข้อมูล (Data Segmentation) จึงเป็นขั้นตอนที่สำคัญก่อนทดลองและเปรียบเทียบโมเดล (Aggarwal, 2016)

ในการสร้างระบบแนะนำสินค้า พบว่ามักจะมีแบ่งข้อมูลในรูปแบบที่สามารถซ่อนรายการของผู้ใช้ (Entry-wise) มากกว่าการซ่อนผู้ใช้ทั้งคน (Row-wise) เนื่องจากการทดสอบมักจะทดสอบในลักษณะว่าระบบสามารถเรียงลำดับ (Ranking) และแนะนำรายการที่ถูกซ่อนไว้ในขั้นตอนนี้ได้ดีเพียงใด

เพื่อให้ใกล้เคียงการใช้งานจริง งานวิจัยจำนวนมากนิยมใช้ Leave-One-Out (หรือ Leave-Last-One-Out เมื่อมีเวลาเป็นจุดอ้างอิงที่สำคัญ) (Zangerle & Bauer, 2022) โดยสำหรับผู้ใช้แต่ละคนจะมีการซ่อนไว้ 1 รายการเป็นชุดทดสอบ และใช้รายการที่เหลือทั้งหมดเป็นชุดฝึก แนวทางปฏิบัติโดยทั่วไปคือ

1. เรียงลำดับการให้คะแนน (หากมีเวลาในชุดข้อมูล)
2. เลือกรายการล่าสุดตามเวลาเป็นข้อมูลชุดทดสอบเพื่อหลีกเลี่ยงการเห็นข้อมูลใน Test set
3. ใช้รายการที่เหลือทั้งหมดฝึกโมเดล
4. ประเมินแบบ Top-K ranking โดยตรวจว่าโมเดลสามารถทำนายการให้คะแนนล่าสุดที่นำออกไปหรือไม่

ด้วยเหตุนี้ การแบ่งข้อมูลแบบ Leave-One-Out (LOO) จึงมีข้อดีที่เหมาะสมกับโมเดลนี้ คือช่วยลดปัญหาข้อมูลรั่วไหล (Data Leakage) และสามารถใช้ข้อมูลได้อย่างคุ้มค่า (เสียข้อมูลไปทดสอบเพียง 1 รายการต่อผู้ใช้เท่านั้น) ทำให้ยังคงมีข้อมูลเพียงพอสำหรับการเรียนรู้รูปแบบความชอบของผู้ใช้ได้ดี นอกจากนี้ LOO ยังสอดคล้องกับโจทย์ของระบบแนะนำที่ต้องทำนายรายการถัดไปจากประวัติที่ผ่านมา โดยเฉพาะแบบ Leave-last-one-out ที่อ้างอิงเวลาเป็นพื้นฐาน ซึ่งช่วยรักษาลำดับเหตุการณ์ให้ถูกต้องและสะท้อนสถานการณ์ใช้งานจริงได้มากกว่า จึงเป็นวิธีที่สอดคล้องและเหมาะสมกับระบบถัดไปที่จะนำมาใช้ และยังนิยมใช้เป็นมาตรฐานในการเปรียบเทียบโมเดลในระบบแนะนำหลายสินค้าอีกด้วย

2.1.8 Normalized Discounted Cumulative Gain (NDCG)

NDCG เป็นหนึ่งในกลุ่มตัวชี้วัดที่ใช้วัดความแม่นยำของการการจัดอันดับ (Ranking) โดยเป้าหมายสำคัญของ NDCG คือการวัดคุณภาพของรายการแนะนำ โดยเน้นว่ารายการที่เกี่ยวข้องหรือมีอรรถประโยชน์ (Utility) สูงควรถูกจัดไว้ในอันดับต้น เพราะผู้ใช้มักให้ความสนใจกับผลลัพธ์ช่วงต้นมากกว่าช่วงท้ายอย่างมีนัยสำคัญ นอกจากนี้ NDCG ยังเหมาะกับงานระบบแนะนำสินค้า เนื่องจากสามารถสะท้อนลำดับของรายการได้โดยตรง ไม่ได้พิจารณาเพียงว่าทำนายถูกหรือผิด แต่พิจารณาว่ารายการที่สำคัญถูกจัดไว้สูงแค่ไหน

แนวคิดพื้นฐานของ NDCG มาจาก Discounted Cumulative Gain (DCG) ซึ่งคำนวณผลรวมค่าความเกี่ยวข้อง (Gain หรือ Utility, g_{ui}) ของแต่ละรายการ โดยมีส่วนหัก (Discount) กับรายการที่อยู่อันดับที่ใกล้เคียงอันดับท้าย ทั่วไปใช้น้ำหนัก $log_2(v_j + 1)$ เป็นตัวหาร โดยที่ตัวแปร v_j คืออันดับของรายการตามสมการที่ 2.19

$$\mathrm{DCG} = \sum_{j \in I_u, v_j \le L} \frac{g_{uj}}{\log_2(v_j + 1)} \qquad 2.19$$

เพื่อให้คะแนนเป็นมาตรฐาน (Normalize) และเปรียบเทียบได้โดยง่ายจึงมีความจำเป็นต้องทราบคะแนนที่ดีที่สุดที่เป็นไปได้ ซึ่งสามารถทราบได้จากการเปรียบเทียบกับลำดับที่เหมาะสมที่สุดในรายการแต่ละชุดโดยการใช้ค่า Ideal Discounted Cumulative Gain (IDCG) ซึ่งคือคำนวณมากจากค่า DCG ของรายการชุดเดิม แต่มีการเรียงลำดับจากความเกี่ยวข้องมากไปน้อยอย่างสมบูรณ์ (Perfect Sorting) ตามสมการที่ 2.20

$$IDCG_k = \sum_{i=1}^{|REL|} \frac{rel_i^{ideal}}{\log_2(i+1)} \qquad 2.20$$

สุดท้ายจึงแปลงค่าให้อยู่ระหว่าง 0 ถึง 1 โดยการนำคะแนนที่โมเดลทำได้ (DCG) หารด้วยคะแนนที่ดีที่สุดที่เป็นไปได้ (IDCG) ตามสมการที่ 2.21 เพื่อให้สามารถเปรียบเทียบได้อย่างชัดเจนและเข้าใจได้ง่าย โดยค่า 0 หมายถึงเรียงลำดับไม่ถูกต้องเลย และ 1 หมายถึงเรียงลำดับได้ถูกต้องทุกอันดับ

$$NDCG_k = \frac{DCG_k}{IDCG_k} \qquad 2.21$$

2.1.9 Hit Rate (HR)

Hit Rate เป็นหนึ่งในกลุ่มตัวชี้วัดที่ใช้วัดคุณภาพของการจัดอันดับ (Ranking) สำหรับระบบแนะนำแบบ Top-N หรือการคาดการณ์รายการที่มีความน่าจะเป็นสูงที่อยู่ภายใน N อันดับต้น (Aggarwal, 2016) ซึ่งการวัดความเที่ยงมีหลักเกณฑ์ในการวัดคือ การใช้รายการที่เกี่ยวข้อง (Relevant item) ของผู้ใช้งานแต่ละคนที่อยู่ในรายการภายใต้ K อันดับต้นว่ามีรายการที่ตรงกันกับที่คาดการณ์หรือไม่ กล่าวคือ หากใช้รายการแนะนำ K รายการจาก Top-K จะมีอย่างน้อย 1 รายการที่ตรงกับรายการจริงที่พบในชุดทดสอบ (Ground Truth) จะถือว่า Hit (ทายถูก) และถูกต้องจึงให้ค่าเป็น 1 แต่ถ้าไม่พบเลยจะเป็น 0

ให้ L_u^K แทนเซตรายการที่แนะนำแก่แนะนำให้ผู้ใช้ u ในอันดับที่ 1 ถึง K และให้ R_u แทนเซตรายการที่เกี่ยวข้องอยู่ในชุดทดสอบ (Test set) จะนิยามค่า Hit ของผู้ใช้ u ที่ K ดังสมการที่ 2.22

$$HR_u@K = \begin{cases} 1, \text{if } L_u^K \in R_u \\ 0, \text{if } L_u^K \notin R_u \end{cases} \qquad 2.22$$

จากนั้นนิยาม HR@K โดยใช้ค่าเฉลี่ยของ HR@K ของผู้ใช้ทั้งหมด ตามสมการที่ 2.23

$$HR@K = \frac{1}{|U|}\sum_{u \in U} hit_u@K \qquad 2.23$$

โดย HR@K มีค่าระหว่างช่วง 0 ถึง 1 และสะท้อนถึงผลลัพธ์ว่าระบบสามารถแนะนำรายการที่เกี่ยวข้องให้ผู้ใช้ภายใต้ K อันดับ (Top-K) ได้มากน้อยเพียงใด ซึ่งจัดเป็นตัวชี้วัดที่ดูความสามารถในการจัดอันดับเช่นเดียวกับตัวชี้วัด NDCG@K

2.1.10 Randomized Search

การปรับ Hyperparameter (Hyperparameter Tuning) เป็นขั้นตอนสำคัญในการสร้างโมเดลระบบแนะนำที่มีประสิทธิภาพ เพราะ Hyperparameter ส่งผลโดยตรงต่อทั้งความแม่นยำและความสามารถในการทำงานกับข้อมูลใหม่ (Generalization) (Aggarwal, 2016) อีกทั้งงานวิจัยจำนวนมากชี้ว่าค่าประสิทธิภาพของโมเดลอาจเปลี่ยนไปมากเพียงเพราะเลือก Hyperparameter ที่ต่างกัน ดังนั้นหากกำหนดค่าไม่เหมาะสม ตัวอย่างเช่นการเลือก λ ในการทำ Regularization โมเดลอาจเกิด Overfitting (โมเดลเรียนรู้มากไป) หรือ Underfitting (โมเดลเรียนรู้น้อยไป) และทำให้ความแม่นยำลดลง

เพื่อทดแทนการค้นหาแบบตาราง (Grid Search) ที่จำเป็นต้องทดสอบกับทุกชุด Hyperparameter การค้นหาแบบสุ่ม (Randomized Search) เป็นวิธีที่นิยมในทางปฏิบัติ (Bergstra & Bengio, 2012) เนื่องจากวิธีดังกล่าวจะสุ่มชุด Hyperparameterจากช่วงหรือการกระจาย (Distribution) ที่กำหนด จากนั้นฝึกโมเดลและประเมินผลโมเดลตามรอบที่ทำหนด (ซึ่งปริมาณรอบนั้นขึ้นอยู่กับต้นทุนในการฝึกฝนโมเดลว่ามีมากขนาดไหน)

Bergstra & Bengio (2012) ระบุว่า Randomized Search มักหาชุด Hyperparameter ที่ได้ผลลัพธ์ที่ดีได้เร็วกว่า Grid Search โดยเฉพาะเมื่อมีชุดของ Hyperparameter มาก เนื่องจากโดยทั่วไปมักมีเพียงบางมิติของเซต Hyperparameter ที่ส่งผลต่อผลลัพธ์อย่างมีนัยสำคัญ โดยขั้นตอนของ Random Search มีขั้นตอนดังนี้

1. กำหนดช่วงค่า/การกระจายของไฮเปอร์พารามิเตอร์ (เช่น Uniform Distribution)
2. กำหนดจำนวนรอบการสุ่ม (เช่น 200 รอบ)
3. ในแต่ละรอบสุ่ม Hyperparameter 1 ชุด ใช้ฝึกฝนบน Training set และประเมินผลบน Validation set
4. เมื่อได้ชุดของ Hyperparameter ที่ดีที่สุดแล้ว ทำการฝึกโมเดลอีกรอบ (Retrain) บน Training และ Validation set และประเมินผลสุดท้ายบน Test set เพื่อใช้รายงานผลจริง

2.2 วิจัยที่เกี่ยวข้อง

2.2.1 การเพิ่มความเอนเอียงในการแยกตัวประกอบเมทริกซ์ (Bias MF)

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

โมเดล Matrix Factorization จะประมาณการคะแนน ($\hat{y}_{ui}$) โดยอาศัยการคำนวณ Dot Product ระหว่างเวกเตอร์แฝง (Latent Vector) ของผู้ใช้ (p_u) และสินค้า (q_i) ดังสมการที่ 2.24

$$y = q_i^T p_u \tag{2.24}$$

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

ด้วยเหตุนี้ การพยายามอธิบายคะแนนทั้งหมดด้วย $q_i^T p_u$ เพียงอย่างเดียวจึงอาจไม่เพียงพอ จึงเสนอโมเดลที่เพิ่ม Bias เข้าไป เพื่อแยกองค์ประกอบของคะแนน ดังสมการที่ 2.25

$$y = \mu + b_i + b_u + q_i^T p_u \tag{2.25}$$

และสมการเป้าหมายจะถูกปรับไปตามสมการที่ 2.26

$$\min_{p,q,\mu,b_i,b_u} \sum_{(u,i)\in\mathcal{K}} (y_{ui} - \mu + b_i + b_u + q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2) \tag{2.26}$$

การเพิ่ม Bias เข้าไปใน MF แบบดั้งเดิมช่วยเพิ่มความแม่นยำในการทำนายได้อย่างมีนัยสำคัญ ทั้งนี้แม้ Bias MF ยังคงเผชิญกับปัญหาความเบาบางของข้อมูลซึ่งสามารถนำไปต่อยอดได้

2.2.2 การประยุกต์ใช้เทคนิคการจัดกลุ่ม (Clustering)

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) ในลักษณะของ Contextual Matrix Factorization โดยให้โมเดลเรียนรู้ความสนใจในระดับกลุ่มผู้ใช้และกลุ่มสินค้า การทำนายคะแนนเป็นไปตามสมการที่ 2.27

$$\hat{y}_{ui} = T_\alpha(q_i, q_{C_i})^T T_\alpha(p_u, p_{C_u}) + T_\beta(b_u, b_{C_u}) + T_\beta(b_i, b_{C_i}) \qquad 2.27$$

ซึ่งผลลัพธ์คือความแม่นยำสูงขึ้นเพียงเล็กน้อยเท่า โดย RMSE ลดลง จาก 0.9025 เป็น 0.9020 แต่ความซับซ้อนของโมเดลที่เพิ่มสูงขึ้นมาก เนื่องจากมี Hyperparameter เพิ่มขึ้นเป็นจำนวนที่มาก และงานดังกล่าวละเลยการประเมินผลแบบ Top-N เช่น NDCG@K ซึ่งสะท้อนความเป็นจริงมากกว่าในงานที่เกี่ยวข้องกับระบบแนะนำสินค้า (Roy et al., 2022)

สิ่งที่สามารถนำไปต่อยอดได้จากงานดังกล่าวได้คือ เราสามารถใช้ Clustering ในการหากลุ่มของผู้ใช้และสินค้าได้โดยไม่ต้องใช้ข้อมูลคุณลักษณะ (Feature) เพิ่มเติมเลย แต่ใช้ User-Item Matrix ในการดึงคุณลักษณะของกลุ่มผู้ใช้และกลุ่มของสินค้าออกมาแทน ซึ่งกลุ่มดังกล่าวก็มี Information แม้จะไม่มากนักดังผลลัพธ์จากงาน

2.2.3 การประยุกต์ใช้เทคนิคสินค้าร่วม (Co-occurrence)

งานวิจัยของ Liang et al., (2016) นำเสนอโมเดล Co-Factor Matrix Factorization (Co-Factor MF) มุ่งเน้นการแก้ไขปัญหาสินค้าที่ไม่ได้เป็นที่นิยมมากนัก (Rare Item) ไม่ถูกแนะนำซึ่งทำให้ความแม่นยำของระบบลดลง โดยสร้างเมทริกซ์ของสินค้าร่วมกัน (Item-Item Co-occurrence Matrix) จากนั้นให้โมเดลเรียนรู้ 2 ส่วน

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 2.28

$$PMI(i,j) = \log \frac{\#(i,j) \cdot D}{\#(i) \cdot \#(j)} \qquad 2.28$$

เมื่อได้ค่า PMI แล้ว จากนั้นทำการแก้ไขฟังก์ชันเป้าหมาย (Objective Function) เพื่อให้โมเดลเรียนรู้ทั้ง User-Item Matrix และ Item-Item Co-occurrence Matrix ดังสมการที่ 2.29 วิธีนี้จึงเปรียบเสมือนการใช้ข้อมูล Co-occurrence เป็นตัวกำกับ (Regularization) เพื่อบังคับให้ Item Factor ที่ได้มีคุณภาพและสมเหตุสมผล

$$\mathcal{L}_{co} = \sum_{u,i} c_{ui}(y_{ui} - p_u^\top q_i)^2 + \sum_{m_{ij} \neq 0} (m_{ij} - q_i^\top \gamma_j - w_i - c_j)^2 \qquad 2.29$$

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 ในงานดังกล่าวน่าสนใจ เพราะเป็นการหาสินค้าประกอบกัน (Complementary) ในเชิงเศรษฐศาสตร์ ซึ่งจะทำให้รับมือกับผู้ใช้ที่แม้ว่าผู้ใช้จะมีประวัติการให้คะแนนน้อย เช่น 1 ชิ้น แต่หากรู้สินค้าที่ใช้ประกอบกันกับสินค้าชิ้นแรกได้ ก็จะสามารถแนะนำสินค้าชิ้นถัดๆ ไปได้ดียิ่งขึ้น การต่อยอดจากงานนี้เนื่องจากผู้ใช้อาจมีหลายกลุ่มความชอบ (Persona) และแต่ละกลุ่มอาจะมีกฎความชอบที่เหมือนกัน (Group Prior) ก็สามารถดึงออกมาประกอบเพื่อเพิ่มความแม่นยำได้
2. การใช้ Co-occurrence Matrix อาจมี Noise ผสมอยู่ หากใช้แนวคิดของ ARM มาคัดกรองกฎบางกฎออก (Confidence/Support/Lift) อาจจะทำให้โมเดลมีความแม่นยำมากขึ้น

การให้โมเดลเรียนรู้แบบ Co-Factor ซึ่งจะทำให้โมเดลเก่งทั้งการทำนายตามเป้าหมายหลัก (First Objective) และเป้าหมายรอง (Second Objective)

2.2.4 การประยุกต์ใช้กฎความสัมพันธ์ (Association Rule Mining)

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

จุดเด่นของงานนี้คือการพยายามใช้ประโยชน์จากข้อมูลทั้งหมดที่ผู้ใช้ให้มาโดยระบุว่าระบบส่วนใหญ่จะเน้นไปที่สินค้าที่ได้คะแนนสูง (Favorite) เพื่อแนะนำของที่คล้ายกัน แต่งานวิจัยนี้ตั้งสมมติฐานว่า แม้แต่สินค้าที่ผู้ใช้ไม่ชอบ (Non-Favorite) ก็ยังสามารถให้ข้อมูลที่เป็นประโยชน์ได้ โมเดลนี้แบ่งการทำงานออกเป็น 2 ส่วนหลัก

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 หมวดหมู่ของภาพยนต์) คล้ายกับสินค้าที่ผู้ใช้ไม่ชอบ และแนะนำสินค้าเหล่านั้น (แนวคิดคือผู้ใช้อาจจะไม่ชอบหนัง Action เรื่องหนึ่ง แต่ก็อาจจะชอบหนัง Action อีกเรื่องหนึ่งก็ได้)

งานวิจัยนี้ชี้ให้เราเห็นว่าการใช้ ARM ในการพัฒนาระบบแนะนำสินค้าสามารถเพิ่มความแม่นยำได้ เนื่องจากการตัดสินใจเลือกสินค้าของผู้ใช้มีลำดับขั้นตอน ซึ่งเป็นจุดที่สามารถนำมาต่อยอดได้

บทที่ 3

วิธีวิจัย

3.1 แบบจำลองที่ใช้ในการวิจัย

แบบจำลองการแยกตัวประกอบเมทริกซ์โดยใช้สัญญาณการจัดกลุ่มและการหากฎความสัมพันธ์ (Clustering Association Rule Mining Signal Matrix Factorization – CARMS MF) ถือเป็นแบบจำลองแบบผสม (Hybrid) เริ่มต้นจากการสร้าง CARMS ด้วยการจัดกลุ่ม (Clustering) และการหากฎความสัมพันธ์ (Association Rule Mining) แบบ 1 ต่อ 1 จากนั้นนำสัญญาณดังกล่าวไปใช้ผสมกับการแยกตัวประกอบเมทริกซ์ (Matrix Factorization) โดยใช้เทคนิคการเรียงลำดับ (Bayesian Personalized Ranking) เป็นเป้าหมายร่วมเพื่อสร้างคำแนะนำสินค้า โดยขั้นตอนจะมี 2 ส่วน คือ สร้าง CARMS (2) ปรับปรุง Objective Function

3.1.1 การสร้างสัญญาณ CARMS

การสร้างสัญญาณ CARMS เพื่อระบุความสัมพันธ์ไปยังสินค้าชิ้นถัดไป ประกอบด้วย 9 ขั้นตอน ดังนี้

1. กำหนด Cold User และ Warm User

เนื่องจากในแต่ละขั้นตอนในการสร้างสัญญาณต่างกันใน Cold User และ Warm User เนื่องจากมีพฤติกรรมที่ต่างกัน โดยเริ่มจากการสร้างคะแนนที่สังเกตได้ (Observed) ตามสมการที่ xx

$$O_{ui} = \mathbb{I}\left[Y_{ui} > 0\right];\ Y \in \mathbb{R}_{\geq 0}^{U \times I} \tag{3.1}$$

จากนั้นแบ่ง Cold User และ Warm User ซึ่งนิยามให้ผู้ใช้ที่มีเซตของคะแนนอย่างน้อย 1 ครั้งจัดเป็น Warm User และหากไม่มีจะนิยามเป็น Cold User ตามสมการที่ 3.2 และ 3.3

$$\Omega_u = \{i |\ Y_{ui} > 0\} \tag{3.2}$$

$$\text{warm} = \{u : |\ \Omega_u\ | > 0\},\ \text{cold} = \{u : |\Omega_u\ | = 0\} \tag{3.3}$$

2. สร้าง Liked/Disliked Matrix

เนื่องจากผู้ใช้แต่ละคนมีพฤติกรรมในการให้คะแนนไม่เหมือนกัน เช่น ผู้ใช้บางรายให้คะแนน 5 กับสินค้าที่ชอบ ในขณะที่ผู้ใช้อีกรายหนึ่งให้คะแนน 3 กับสินค้าที่ชอบ ดังนั้นเมื่อต้องการแบ่งกลุ่มตามสินค้าที่ผู้ใช้ชอบ (หรือไม่ชอบ) จึงเริ่มจากการหาเกณฑ์ขั้นต่ำที่ผู้ใช้จะชอบจากค่ากลาง คือ มัธยฐาน (Median) ของผู้ใช้รายคน ตามสมการที่ 3.4

$$m_u = \text{median}\{Y_{ui} \mid i \in \Omega_u\};\ \text{if}\ \Omega_u = \varnothing, \text{define}\ m_u = 0 \qquad 3.4$$

จากนั้นนำค่าดังกล่าวไปเปรียบเทียบกับการให้คะแนนจริงของผู้ใช้รายคนในกลุ่ม Warn หากผู้ใช้มีคะแนนดังกล่าวมากกว่าหรือเท่ากับค่ากลางจะถือว่าชอบ แทนด้วย 1 หากน้อยกว่าจะถือว่าไม่ชอบ แทนด้วย -1 และสินค้าที่ไม่เคยให้คะแนนเลยจะแทนด้วย 0 เมื่อรวมทั้งหมดเข้าด้วยกันจะได้ Liked/Disliked Matrix ของผู้ใช้ที่มีการให้คะแนนอย่างน้อย 1 ครั้ง ตามสมการที่ 3.5 และจะได้ $Z \in \{-1,0,1\}^{U \times I}$

$$Z_{ui} = \begin{cases} +1, & i \in \Omega_u \wedge Y_{ui} \geq m_u \\ -1, & i \in \Omega_u \wedge Y_{ui} < m_u \\ 0, & i \notin \Omega_u \end{cases} \qquad 3.5$$

เมื่อได้ Liked/Disliked Matrix ของผู้ใช้ที่มีการให้คะแนนอย่างน้อย 1 ครั้ง นำไปจัดกลุ่ม $k-1$ กลุ่ม ซึ่งเป็น Hyperparameter ด้วย K-Means ดังนั้นจะมี Group label ทั้งสิ้น $k-1$ กลุ่ม ตามสมการที่ 3.6

$$c_u \in \{0,1,\ldots,k-1\} \qquad 3.6$$

สุดท้ายจะนำ Centroid ไปใช้กับกลุ่มที่ไม่เคยให้คะแนนด้วย ดังนั้นกลุ่มที่ผู้ใช้ผู้ใช้ที่ไม่เคยให้คะแนนได้อยู่จะเป็นกลุ่มที่ Centroid มีลักษณะเป็นกลาง เช่น กลุ่มที่ผู้ใช้ให้คะแนน Ratings ที่ชอบและไม่ชอบหักล้างกันพอดี หรือสำคัญที่สุดคือกลุ่มที่ให้คะแนนกับสินค้าไม่มากชิ้น (ซึ่งเป็นสิ่งที่ต้องการพอดี) เหมาะกับผู้ใช้ที่เป็น Cold User (โดยเฉพาะ Coldest หรือไม่เคยให้คะแนนกับสินค้าใดเลย) เพราะจะยังมีสัญญาณให้กับผู้ใช้ทุกราย

3. เลือกเซตของ Active Item

สินค้าที่จะนำไปใช้สร้าง ARM นั้นจะต้องเป็นสินค้าที่ Active กล่าวคือเป็นสินค้าที่ผู้ใช้มีการให้คะแนนมากกว่า 1 ครั้ง (ดังนั้น Signal นี้จะไม่ได้แก้ไขปัญหา Item Cold Start) โดยนับจำนวนผู้ใช้ที่เคยให้คะแนน

$$\mathrm{pop}(i) = \sum_{u \in \mathrm{warm}} O_{ui} \tag{3.7}$$

และกำหนดสินค้าที่จัดว่าเป็น Active Item หมายถึงสินค้าที่มีการให้คะแนนมากกว่า 1 ครั้ง

$$\mathcal{I}_a = \{i : \mathrm{pop}(i) > 0\};\ I_a = |\mathcal{I}_a| \tag{3.8}$$

4. สร้าง Augmented Transaction

เนื่องจากเราอยากได้กฎในลักษณะแบบ Co occurrence ที่หลากหลายมากขึ้น ทั้งกฎที่เป็นในลักษณะสินค้าต่อสินค้า (Item-to-Item) และกลุ่มของผู้ใช้ต่อสินค้า (Cluster-to-Item) ดังนั้นจะมีการปรับแก้ Transaction ของผู้ใช้รายคน (Augmented) โดยเริ่มจากแปลง Transaction ของผู้ใช้เป็น Dummy

$$\mathbf{o}_u = \left(O_{ui}\right)_{i \in \mathcal{I}_a} \in \{0,1\}^{I_a} \tag{3.9}$$

นิยาม One-hot ของแต่ละ Cluster

$$\mathbf{e}_{c_u} \in \{0,1\}^k \tag{3.10}$$

นำ Cluster One-hot ไปรวมกับ Transaction เดิมของผู้ใช้

$$\mathbf{x}_u = \left[\mathbf{o}_u ; \mathbf{e}_{c_u}\right] \in \{0,1\}^{I_a + k} \tag{3.11}$$

ดังนั้น X จะแทน Augmented Transaction ที่จะมี Dimension ที่มีขนาดเท่ากับ $N \times (I_a + k)$

$$\mathbf{X} \in \{0,1\}^{N \times (I_a + k)} \tag{3.12}$$

5. สร้างความสัมพันธ์แบบ ARM

จากนั้นคำนวณ Indicator ที่เป็นประกอบของ ARM โดยเริ่มจากการหาสินค้าที่เกิดร่วมกันในลักษณะแบบ 1 ต่อ 1 ซึ่งสิ่งนี้คือจำนวนของสินค้าร่วมกัน (จำนวน $a \cap b$) ของทุกสินค้าในระบบ

$$\mathbf{C} = \mathbf{X}^{\top}\mathbf{X} \in \mathbb{R}^{(I_a+k)\times(I_a+k)} \tag{3.13}$$

และตัด Columns ที่เป็น Cluster ออกเนื่องจากต้องการ Consequent ที่เป็นสินค้าเท่านั้น

$$\mathbf{C}^{(\text{item})} = \mathbf{C}[:, 1{:}I_a] \in \mathbb{R}^{(I_a+k)\times I_a} \tag{3.14}$$

หา Support โดยให้ n_a คือจำนวนของ a ที่ปรากฏใน $C^{(item)}$ และ n_{ab} คือจำนวนของ a และ b ที่ปรากฏใน $C^{(item)}$ ส่วน N ใช้จำนวน Warm User เนื่องจากผู้ใช้หนึ่งคนจะแทน 1 Transaction

$$\sup(a \to b) = \frac{n_{ab}}{N} \tag{3.15}$$

ส่วน Confidence จะเห็นว่าไม่จำเป็นต้องคำนวณ Support ของ A แยก เนื่องจาก N จะตัดกันพอดี

$$\text{conf}(a \to b) = \frac{n_{ab}}{n_a} \tag{3.16}$$

จากนั้นคำนวน Support ของ b เพื่อใช้เป็นส่วนประกอบของ Lift

$$p(b) = \frac{n_b}{N};\ b \in \{1, \ldots, I_a\} \tag{3.17}$$

และนำผลจาก 3.19 และ 3.17 แทนค่าในสมการที่ xx เพื่อหา Lift (แต่ใส่ eps กันหารด้วย 0)

$$\ell(a \to b) = \frac{\text{conf}(a \to b)}{\max\big(p(b), \varepsilon\big)} \tag{3.18}$$

เพื่อป้องกันค่าของ Lift สูงเกินในสินค้าที่หายาก (Rare Item) จะลดค่าด้วยการหา Square Root

$$\tilde{\ell}(a \to b) = \sqrt{\ell_c(a \to b)} \tag{3.19}$$

6. สร้าง Score Matrix

กรองเอากฎที่เป็น Noise ออกตาม Hyperparameter ที่กำหนด

$$M_{ab} = \mathbb{I}\left[\sup(a \to b) \geq \sigma \wedge \text{conf}(a \to b) \geq \kappa\right] \quad 3.20$$

หากกฎใดผ่านค่า Minimum Confidence และ Minimum Support จะแทนด้วย 1

$$M_{ii} = 0 \quad \forall i \in \{1, \ldots, I_a\} \quad 3.21$$

จากนั้นคำนวณคะแนน ซึ่งจะใช้ Confidence คูณกับ Lift (ความโดดเด่นของกฎ) และคูณด้วย Masking

$$W_{ab} = \text{conf}(a \to b) \times \tilde{\ell}(a \to b) \times M_{ab} \quad 3.22$$

สุดท้ายจะได้ W ซึ่งเป็นตัวบอกน้ำหนักของแต่ละกฎ

$$W \in \mathbb{R}^{(I_a + k) \times I_a} \quad 3.23$$

5. สร้าง CARMS

จากนั้นจะ Mapping กฎเข้าไปหา Transaction ของผู้ใช้รายคน เริ่มจาก Warm user

$$s_u = x_u W \in \mathbb{R}^{I_a} \quad 3.24$$

แต่สำหรับ Cold user สามารถดึงเอาคะแนนตาม Cluster ได้เลย

$$s_u = W_{(I_a + c),:} \in \mathbb{R}^{I_a} \quad 3.25$$

สุดท้ายจะใช้สัญญาณเฉพาะสินค้าที่ผู้ใช้ยังไม่เคยให้คะแนน ดังนั้นจะลบสินค้าที่ผู้ใช้เคยเห็นออก

$$s_{ui} = 0 \quad \forall i \in \Omega_u \quad 3.26$$

3.1.5 การปรับปรุงสมการเป้าหมายด้วยเทคนิคเป้าหมายร่วม (Co-Objective)

เมื่อได้สัญญาณ CARMS ซึ่งบอกสินค้าชิ้นถัดไปแล้ว ขั้นตอนต่อไปคือการใช้ CARMS ในการช่วยเรียงลำดับสินค้าที่ทำนายค่ามาจาก MF ผ่านการแก้ไขฟังก์ชันเป้าหมาย (Objective Function) เพื่อปรับให้โมเดลสามารถเรียนรู้ได้ทั้งข้อมูลที่มาจาก User-Item Matrix และ Signal Matrix โดยใช้วิธีการปรับปรุงสมการเป้าหมายด้วยวิธีที่เรียกว่าเทคนิคเป้าหมายร่วม (Co-Objective) จากงานของ Liang et al., (2016) โดยค่าความคลาดเคลื่อนจะมาจาก (1) การแก้ปัญหาของ MF (SSE) ซึ่งพจน์ดังกล่าวจะทำงานเมื่อ $Y > 0$ และมาจาก (2) การจัดอันดับของ BPR (BPR Loss) ซึ่งพจน์ดังกล่าวจะทำงานเมื่อ $Y = 0$ และ $S > 0$

สมการที่ 3.27 คือสมการของ CARMS MF จะมีทั้งหมด 3 ส่วน คือ (1) ส่วนที่เป็น MF ปกติ (2) ส่วนที่เป็น BPR ซึ่งใช้จัดอันดับ และเสนอให้เพิ่มน้ำหนักด้วยความเข้มข้นของ CARMS และ (3) ส่วนที่เป็น Regularization เพื่อกันโมเดลเรียนรู้มากเกินไป (Overfitting)

$$\mathcal{L}_{co} = \sum_{y_{ui}>0} (y_{ui} - \hat{y}_{ui})^2 + \gamma \sum_{y_{ui}=0, s_{ui}>0} [-w_{ui} \log \sigma(\hat{y}_{ui} - \hat{y}_{uj})] + \lambda(\| P \|_F^2 + \| Q \|_F^2) \qquad 3.27$$

1. ส่วน SSE

จากสมการที่ 3.1 เมื่อ $Y > 0$ โมเดลจะเรียนรู้เฉพาะจาก User-Item Matrix (First Objective) เป้าหมายคือการลดค่าความต่างของ y_{ui} และ $\hat{y}_{ui}$ เป็นลักษณะของการทำนายค่าแบบจุด (Pointwise) หากความต่างระหว่างค่าจริงที่สังเกตได้ (Observed) กับค่าที่ทำนาย (Predicted) มีค่าน้อย ค่าความคลาดเคลื่อนก็จะน้อยลง ตามสมการที่ 3.28

$$\mathcal{L}_{SSE} = \sum_{y_{ui}>0} (y_{ui} - \hat{y}_{ui})^2 \qquad 3.28$$

แต่จะมีการเพิ่มความเอนเอียง (Bias) ทั้งความเอนเอียงรวม (Global Bias) ความเอนเอียงของผู้ใช้ (User Bias) และความเอนเอียงของสินค้า (Item Bias) ทำให้ $\hat{y}_{ui}$ ถูกนิยามตามสมการที่ 3.29

$$\hat{y}_{ui} = \mu + b_u + b_i + p_u^T q_i \qquad 3.29$$

2. ส่วน BPR

จากสมการที่ 3.1 เมื่อ $Y = 0$ และ $S > 0$ โมเดลจะเรียนรู้เฉพาะจาก CARMS (Second Objective) ตามสมการที่ 3.30 และเนื่องจากเป้าหมายของ CARMS คือการบอกความสำคัญ (ไม่ได้บอกคะแนน 1 ถึง 5) ดังนั้นเราต้องการส่วนช่วยปรับอันดับ (Re-ranking) ให้ถูกต้อง ส่วน BPR ที่เพิ่มเข้ามาเพื่อเปลี่ยนจากเป้าหมายเดิมที่ทำนายค่า $\hat{y}_{ui}$ ให้แม่นยำ (Pointwise) มาเป็นการเรียงลำดับของคู่ข้อมูล $\hat{y}_{ui}$ และ $\hat{y}_{uj}$ ให้ถูกต้อง (Pairwise) โดยใช้ s_{ui} เป็นตัวบอกค่าที่เป็นบวกเทียม (Pseudo-positive) และค่าความสำคัญ (Weight) ของความผิดพลาดในการเรียงลำดับ

$$\mathcal{L}_{BPR} = \sum_{y_{ui}=0, s_{ui}>0} [-w_{ui} \log \sigma(\hat{y}_{ui} - \hat{y}_{uj})] \tag{3.30}$$

ในส่วนของ BPR ขั้นตอนในการคำนวณค่า BPR Loss จะมีความซับซ้อนมากกว่า ดังนี้

1. สุ่มเลือก $s_{ui} > 0$ (Observed) แบบ Uniform Distribution ใช้เป็นตัวแทนของ I^+

$$\text{positive item } i \in \mathcal{P}_u = \{i |\ y_{ui} = 0, s_{ui} > 0\} \tag{3.31}$$

2. สุ่มเลือก $s_{ui} = 0$ (Unobserved) ด้วยการกระจายแบบ Uniform ใช้เป็นตัวแทนของ I^-

$$\text{negative item } j \in \mathcal{N}_u = \{j |\ y_{uj} = 0, s_{uj} = 0\} \tag{3.32}$$

3. คำนวณส่วนต่างระหว่าง I^+ (Positive) และ I^- (Negative) ซึ่งส่วนต่างดังกล่าวจะแสดงถึงความถูกต้องในการจัดอันดับ

$$x_{uij} = \hat{y}_{ui} - \hat{y}_{uj} \tag{3.33}$$

4. นำค่าความแตกต่างเปลี่ยนเป็นความน่าจะเป็นที่จะทำนายถูกต้อง โดยผ่าน Sigmoid Function ที่จะเปลี่ยนค่าให้มีช่วงอยู่ใน 0 ถึง 1

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.34}$$

5. เนื่องจากต้องการคงเป้าหมายรวมของโมเดล คือการลดความผิดพลาด (Minimize Loss) จะนำค่าของข้อ 4 ไปเปลี่ยนเป็นส่วนกลับของความน่าจะเป็นที่จะทำนายได้ถูกต้อง (เป็น ความน่าจะเป็นที่ทำนายไม่ถูกต้อง) กล่าวคือหากค่าดังกล่าวมีค่ามากแสดงว่าโมเดลเรียงลำดับได้ไม่ถูกต้อง ดังนั้นเป้าหมายคือการลดค่านี้ให้ได้มากที่สุด

$$\mathcal{L}_{\text{BPR}}(u,i,j) = -w_{ui} \log\left(\sigma(x_{uij})\right) \qquad 3.35$$

6. จากข้อ 1 ถึง 5 คือการเรียงลำดับแบบ BPR ทั่วไป (Vanila BPR) แต่เนื่องจากงานวิจัยต้องการให้ CARMS บ่งบอกถึงความสำคัญของคู่นั้นๆ กล่าวคือหาก CARMS มีค่ามาก ก็ควรจะต้องเรียงลำดับได้ถูกต้อง (และโดนลงโทษหนักหากเรียงไม่ถูกต้อง) ดังนั้นค่าความผิดพลาดจากข้อ 5 จะถูกนำมาถ่วงน้ำหนักด้วย โดยแปลงค่า (Normalized) เพื่อทำให้ค่าไม่ใหญ่จนเกินไป

$$w_{ui} = \log(1 + s_{ui}) \qquad 3.36$$

7. เมื่อได้ค่าความคลาดเคลื่อนที่มาจากพจน์ทั้ง 2 พจน์ คือ MF แล BPR ค่าทั้งสองจะถูกนำมารวมกัน โดยมี γ เป็นค่า Hyperparameter ที่จะบอกว่า ให้ BPR มีอิทธิพลกับความคลาดเคลื่อนรวมมากเพียงใด ตามสมการที่ 3.37

$$\min_{\mathbf{P},\mathbf{Q},\mathbf{b}_u,\mathbf{b}_i,\mu} \mathcal{L}_{\text{SSE}} + \gamma \mathcal{L}_{\text{BPR}} + \mathcal{R} \qquad 3.37$$

3. ส่วน Regularization

ส่วนนี้มีไว้เพื่อป้องกันไม่ให้โมเดลเรียนรู้มากเกินไป (Overfitting) โดยเป็นกำลังสองของผลรวมของ Embedding คือ P และ Q โดยคุมความแรงของ Regularization ด้วย λ กล่าวคือ หากกำหนดค่าดังกล่าวไว้มาก หมายความว่าต้องการป้องกันไม่ให้โมเดลเรียนรู้มากจนเกินไป (Overfitting) แต่ความแม่นยำก็จะลดลง

$$\lambda\left(\| P \|_F^2 + \| Q \|_F^2\right) \qquad 3.38$$

3.1.6 การเรียนรู้ (Training)

การแก้ปัญหา (Solver) ในงานนี้ใช้วิธี Gradient Descent โดยเฉพาะรูปแบบ Stochastic Gradient Descent (SGD) ตามแนวทางของ Koren et al. (2009) ซึ่งสามารถจำแนกวิธีการปรับปรุงพารามิเตอร์ได้ทั้งสิ้น 3 แบบ ได้แก่

1. Full-batch Gradient Descent: คำนวณและปรับปรุงพารามิเตอร์โดยใช้ข้อมูลทุกแถว (Observation) พร้อมกันในแต่ละรอบการเรียนรู้
2. Stochastic Gradient Descent (SGD): ปรับปรุงพารามิเตอร์ทีละแถว ทำให้ปรับค่าได้ถี่และเร็วกว่า แต่มีความผันผวนมากกว่า
3. Mini-batch Gradient Descent: ปรับปรุงพารามิเตอร์ครั้งละชุดย่อยของข้อมูล โดยกำหนดขนาดชุดของข้องมูล (Batch) เช่น 64 หรือ 512 แถว ซึ่งเป็นแนวทางที่นิยมใช้มากที่สุดในทางปฏิบัติ เพราะสมดุลระหว่างความเร็วและความเสถียรของการเรียนรู้ และ CARMS MF เลือกใช้แนวทางนี้
4. กระบวนการเรียนรู้เริ่มต้นจากการกำหนดตัวแปรที่โมเดลต้องเรียนรู้ ได้แก่ คือ P และ Q รวมถึงพารามิเตอร์ความเอนเอียง (Bias) อีก 3 ส่วน คือ เอนเอียงรวม (Global bias) ความเอนเอียงของผู้ใช้ (User bias) และความเอนเอียงของสินค้า (Item bias) รวมทั้งหมด 5 ตัวแปรหลัก

$$\theta = \{P, Q, b_u, b_i, \mu\} \tag{3.39}$$

จากนั้นระบุความคลาดเคลื่อนรวม (Total Loss) ซึ่งมาจาก 3 ส่วน คือ SSE กับ BPR Loss และ Regularization เป็นไปตามสมการที่ x

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{SSE}}(\theta) + \gamma \mathcal{L}_{\text{WBPR}}(\theta) + \lambda \left(\| P \|_F^2 + \| Q \|_F^2 \right) \tag{3.40}$$

คำนวณอนุพันธ์ต่อพารามิเตอร์ทุกตัว ซึ่งในกรณีนี้เรามีตัวแปรที่ต้องเรียนรู้ทั้งสิ้น 5 ตัว ดังนั้นจะต้องมีอนุพันธ์ของทั้ง 5 ตัวแปรเพื่อจะนำไปปรับปรุง (Update) ค่าต่อไป

$$\frac{\partial \mathcal{L}}{\partial p_u}, \frac{\partial \mathcal{L}}{\partial q_i}, \frac{\partial \mathcal{L}}{\partial b_u}, \frac{\partial \mathcal{L}}{\partial b_i}, \frac{\partial \mathcal{L}}{\partial \mu} \tag{3.41}$$

ขั้นสุดท้ายจะต้องใช้อนุพันธ์เป็นตัวบอกทิศทางในการปรับปรุงค่าของตัวแปรทั้ง 5 ตัว ตามสมการที่ 3.42 ถึง 3.46 โดยความเร็วในการปรับปรุงค่าจะขึ้นอยู่กับ η (Learning Rate) หาก η มีค่าสูง โมเดลอาจเรียนรู้ได้เร็ว แต่มีความเสี่ยงที่จะก้าวข้ามจุดต่ำสุด (Global Minimum) ทำให้ไม่ลู่เข้าสู่คำตอบที่เหมาะสมที่สุดได้ ในทางกลับกัน หาก η มีค่าต่ำเกินไป โมเดลจะเรียนรู้ช้าและอาจไม่สามารถลู่เข้าใกล้ Global Minimum ได้ภายในเวลาที่กำหนด ดังนั้นจึงต้องกำหนด η ให้เหมาะสม โดยควรพิจารณาควบคู่กับจำนวนรอบการเรียนรู้ (Epochs) เพื่อให้เกิดความสมดุลระหว่าง ความเร็วในการเรียนรู้ และความเสถียรของการลู่เข้าของโมเดล

$$p_u \leftarrow p_u - \eta \frac{\partial \mathcal{L}}{\partial p_u} \tag{3.42}$$

$$q_i \leftarrow q_i - \eta \frac{\partial \mathcal{L}}{\partial q_i} \tag{3.43}$$

$$\mu \leftarrow \mu - \eta \frac{\partial \mathcal{L}}{\partial \mu} \tag{3.44}$$

$$b_u \leftarrow b_u - \eta \frac{\partial \mathcal{L}}{\partial b_u} \tag{3.45}$$

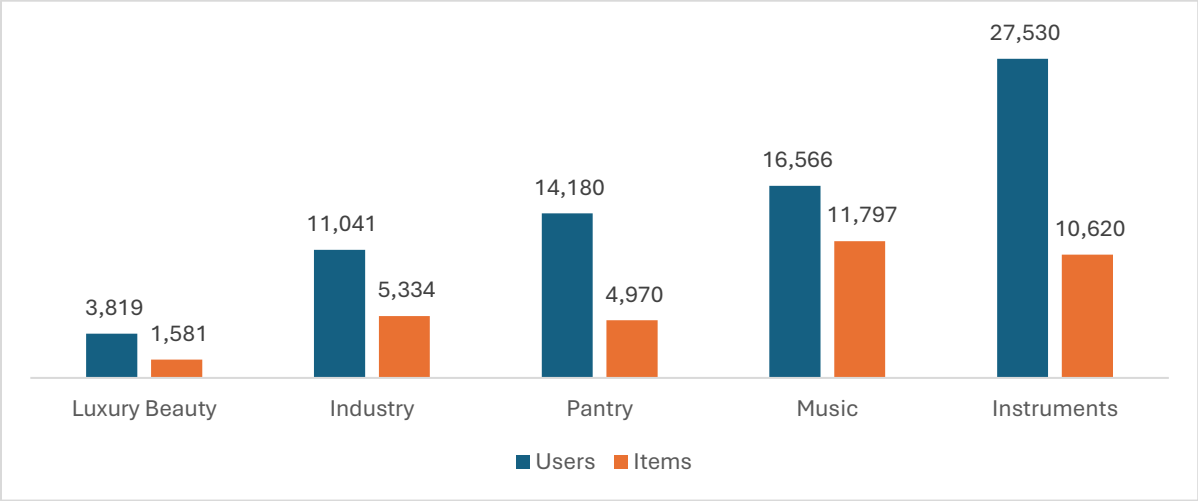
$$b_i \leftarrow b_i - \eta \frac{\partial \mathcal{L}}{\partial b_i} \tag{3.46}$$

3.2 ข้อมูลที่ใช้

งานวิจัยนี้วัดประสิทธิภาพโมเดล CARMS-MF ด้วยชุดข้อมูลมาตรฐาน Amazon 5 Ratings ซึ่งเป็นชุดข้อมูลของเว็บไซต์ Amazon ที่มีการตัดทอนสินค้าและผู้ใช้ออกเฉพาะผู้ที่มี Ratings มากกว่า 5 เท่านั้น โดยเพื่อความแม่นยำ (Robust) ในการวัดผล จะใช้ชุดข้อมูลที่แตกต่างกันทั้งสิ้น 5 หมวดหมู่ประกอบด้วย ความงาม (Amazon Luxury Beauty อุตสาหกรรม (Amazon Industry ของกินของใช้ในครัว (Amazon Pantry ดนตรี (Amazon Music และเครื่องดนตรี (Amazon Instruments เพื่อดูว่าโมเดล CARMS MF ที่นำเสนอสามารถทำงานได้ดีหรือไม่เมื่อเผชิญกับสินค้าต่างชนิดกัน

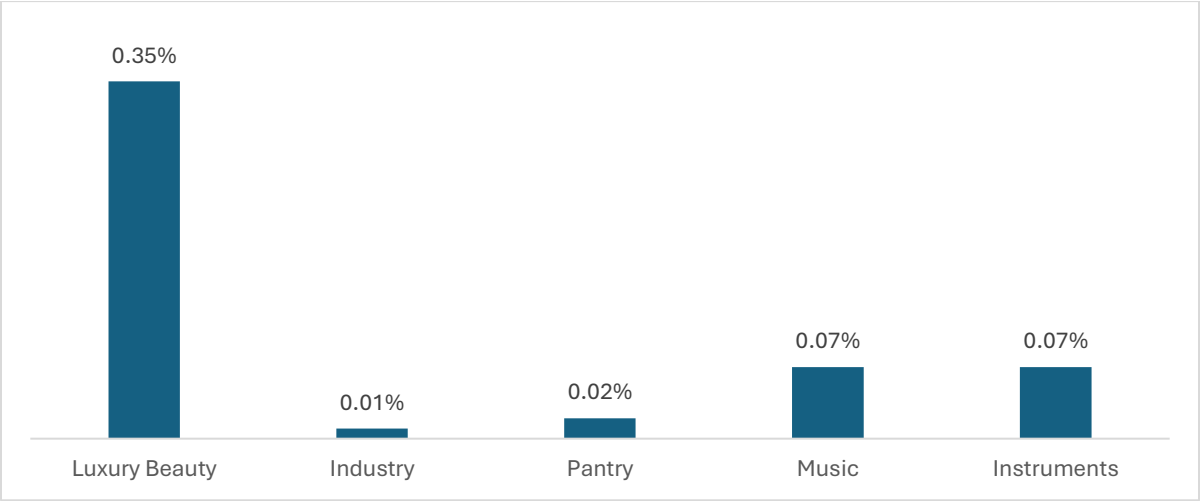
พิจารณาจากจำนวนสินค้าและผู้ใช้ในแต่ละชุดข้อมูล ชุดข้อมูลที่มีจำนวนผู้ใช้และสินค้ามากที่สุดคือ Amazon Luxury Beauty โดยมีจำนวนผู้ใช้ 3,819 ราย และจำนวนสินค้า 1,581 ชิ้น (6,037,839 ช่องคะแนน) และชุดข้อมูลที่มีข้อมูลมากที่สุดคือ Amazon Instruments โดยมีจำนวนผู้ใช้ 27,530 ราย และจำนวนสินค้า 10,620 ชิ้น และมี (292,368,600 ช่องคะแนน)

รูปที่ 3.1: จำนวนผู้ใช้และสินค้าในชุดข้อมูล Amazon



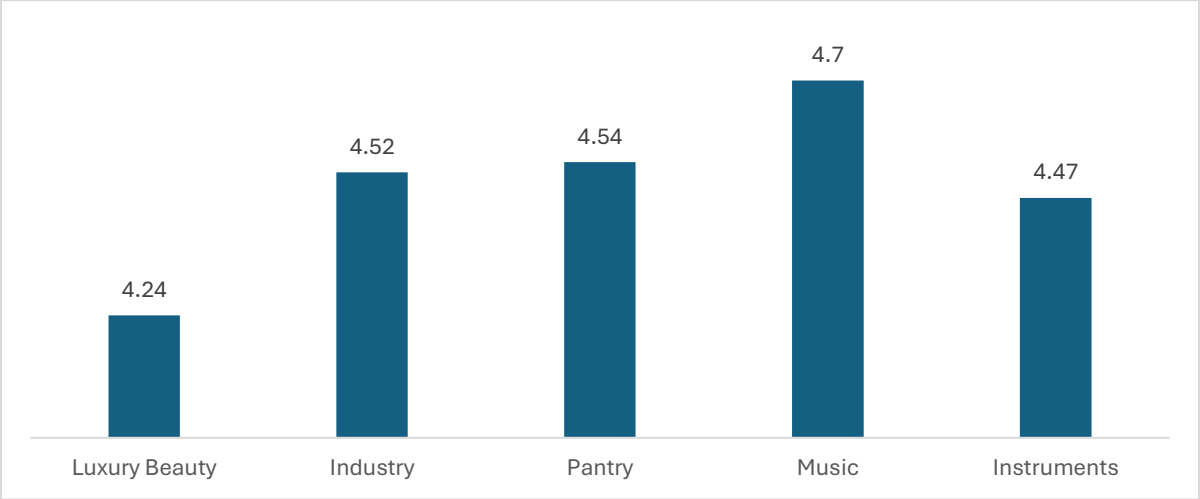
เมื่อพิจารณาขนาดแล้ว อีกสิ่งหนึ่งที่ต้องพิจารณาคือความเบาบางของข้อมูลซึ่งเป็นหนึ่งในปัญหาที่ใหญ่ที่สุดในการสร้างระบบแนะนำสินค้า จากข้อมูลพบว่าข้อมูลทั้งหมดมีความเบาบางมาก โดยข้อมูลชุดที่เบาบางน้อยที่สุดคือ Amazon Luxury Beauty มีข้อมูลราว 0.35% จากช่องข้อมูลทั้งหมด และชุดข้อมูลที่เบาบางมากที่สุดคือ Amazon คือมีข้อมูลจริงเพียง 0.01% ของช่องข้อมูลมั้งหมด และชุดข้อมูล Amazon Pantry ที่มีข้อมูลจริงเพียง 0.02% เท่านั้น

รูปที่ 3.2: ความเบาบางของชุดข้อมูล Amazon



สุดท้ายคือค่าเฉลี่ยของการให้คะแนน ชุดข้อมูลทั้ง 5 ชุดมีคะแนนสูงสุดและต่ำสุดเท่ากัน คือมีค่าตั้งแต่ 1 ถึง 5 (Rating Scale) หากดูค่าเฉลี่ยของคะแนนที่อยู่ที่มากกว่า 4 เหมือนกันทั้ง 5 ชุดข้อมูล แปลว่าการให้คะแนนโดยเฉลี่ยค่อนไปทางสูงเมื่อดูจากค่าเฉลี่ย

รูปที่ 3.3: ค่าเฉลี่ยของการให้คะแนนของชุดข้อมูล Amazon



กล่าวโดยสรุปคือชุดข้อมูล Amazon ทั้ง 5 ชุดข้อมูล มีปัญหาเดียวกันคือ ความเบาบางของข้อมูล โดยเฉพาะข้อมูลในชุด Amazon Luxury Beauty และชุดข้อมูล Amazon Pantry และในส่วนของการให้คะแนนพบว่ามีการให้คะแนนไปในทางสูง จึงอาจต้องพิจารณาเรื่องการปรับฐานข้อมูลหากต้องการดูความชอบ (Preference)

3.3 ชุด Hyperparameter

ในงานวิจัยดังกล่าวใช้ Randomized search โดยชุด Hyperparameter ที่จะค้นหาจะมี Hyperparameter และ 5 ตัวแปร และจะใช้วิธีการ Random ขึ้นมาแบบ Uniform (ทุก Hyperparameter มีโอกาสถูกหยิบขึ้นมาเท่ากันหนด)

จากตารางที่ 3.1 จะพบว่า จะมีการใส่ Hyperparameter ไว้เพื่อตรวจสอบความสำคัญของขั้นตอนต่างๆ ที่ในงาน CARMS MF นำเสนอ ดังนี้

1. Latent: ตัวแปรนี้จะกำหนดมิติ (Dimension) ของ Latent Vector P และ Q หากค่า Latent มีค่ามาก แสดงว่าลักษณะของทั้งฝั่งผู้ใช้ (P) และสินค้า (Q) มีความหลากหลายต่างกัน
2. Learning rate: ตัวแปรนี้จะกำหนดความไวในการ Update ค่า Parameter ต่างๆ ในระหว่างกระบวนการเรียนรู้
3. Lambda: ตัวแปรนี้จะเป็นตัวกำหนดขนาดของชุด Regularization เพื่อป้องกันการ Overfitting หากมีค่ามากแปลว่าเราป้องกันไม่ให้โมเดลเกิด Overfitting มาก (แต่ก็อาจจะเสี่ยงต่อการ Underfitting)
4. Epochs: ตัวนี้แปรเป็นตัวกำหนดรอบของการอัพเดทค่า Parameter ของ Gradient decent
5. User cluster: ตัวแปรนี้จะกำหนดจำนวนกลุ่มของผู้ใช้ตามความชอบ (Preference) โดยเริ่มตั้งแต่ 1 จนถึง 10 หากโมเดลเลือก User cluster เท่ากับ 1 แสดงว่าส่วนการแบ่งกลุ่มเพื่อหา Group Prior ประกอบกับกฎความสัมพันธ์ตามที่งานเสนอไม่มีผลกับความแม่นยำ
6. Minimum support & confidence: ตัวแปรนี้จะเป็นตัวกรอง Noise ออกจากกฎทั้งหมดในแบบของ ARM หากค่ามีค่าเท่ากับ 0 หมายความว่าส่วนการกรอง Noise ด้วยกฎแบบ ARM ตามที่งานเสนอไม่มีผลกับควาแม่นยำ
7. Gamma: ตัวแปรนี้จะบอกถึงความสำคัญของพจน์ BPR ที่ใช้ Re-rank ด้วยสัญญาณ CARSM หากโมเดลเลือกค่านี้เท่ากับ 0 หมายความว่า CARMS นั้นไม่มีผลกับความแม่นยำเลย

ตารางที่ 3.1: ชุด Hyperparameter ที่ใช้ในงานศึกษา

Hyperparameter	Search space
Latent	20, 30, 50, 70, 100
Learning rate	0.001, 0.005, 0.01, 0.05, 0.1
Lambda	0.001, 0.005, 0.01, 0.05, 0.1
Epochs	20, 30, 50, 70, 100
User cluster	1, 3, 5, 7, 10
Minimum support	0.0, 0.00001, 0.0001, 0.001, 0.01
Minimum confidence	0.0, 0.0001, 0.001, 0.01, 0.1
Gamma	0.0, 0.1, 1, 10, 100

บทที่ 4

ผลการวิจัยและอภิปรายผล

4.1 ความแม่นยำของโมเดล

จากการทดสอบโมเดล CARMS MF เมื่อเทียบกับตัวเทียบ (Baseline) คือ Bias MF ตามตารางที่ 4.1 พบว่า คะแนน NDCG@10 ซึ่งเป็นตัวชี้วัดหลักที่จะใช้ในงานศึกษาครั้งนี้ เพิ่มขึ้นทุกชุดข้อมูล โดย Amazon Luxury Beauty เพิ่มขึ้นจาก 18.2% เป็น 26.7% (เพิ่มขึ้น 1.46 เท่า) Amazon Industry เพิ่มขึ้นจาก 4.6% เป็น 6.3% (เพิ่มขึ้น 1.35 เท่า) Amazon Pantry เพิ่มขึ้นจาก 0.7% เป็น 2.1% (เพิ่มขึ้น 3 เท่า) Amazon Music เพิ่มขึ้นจาก 2.1% เป็น 3.6% (เพิ่มขึ้น 1.73 เท่า) และ Amazon Instruments เพิ่มขึ้นจาก 4.2% เป็น 7% (เพิ่มขึ้น 1.69 เท่า) ทุกชุดข้อมูลแสดงถึงประสิทธิภาพในการทำนายที่สูงขึ้นมาก โดยเฉพาะกับชุดข้อมูล Amazon Pantry สัญญาณ CARMS ทำให้ความแม่นยำในการเรียงลำดับเพิ่มขึ้นมากโดยคาดว่าสินค้าประเภทของชำนั้นเป็นของที่เหมาะกับกฎในลักษณะ ARM อยู่แล้ว

หากพิจารณาถึงความยาว (List) ในการแนะนำที่แตกต่างกัน พบว่า CARMS MF สามารถชนะโมเดลพื้นฐานได้ทุกความยาวและทุกชุดข้อมูล โดยพิจารณา NDCG ในความยาวที่แตกต่างกัน คือ 5, 10, 20, 50 และ 100 ตรงกับเป้าหมายของงาน

นอกจากนั้นหากพิจารณาตัวชี้วัดอื่น (แต่ยังคงเป็นตัวชี้วัดเพื่อวัดประสิทธิภาพ) คือ HR@K ซึ่งตีความได้ง่ายกว่า NDCG โมเดล CARMS MF ก็ยังคงชนะโมเดลพื้นฐานได้ทุกความยาวและทุกชุดข้อมูลเช่นกัน แสดงถึงความสามารถในการจัดอันดับที่สูงขึ้นของ CARMS MF

4.2 จำนวน Cluster ของผู้ใช้

ในชุด Hyperparameter จะมีชุดที่ให้ค่าของ User cluster เท่ากับ 1 เพื่อพิจารณาว่าจำนวน Cluster ที่งานนำเสนอจำเป็นหรือไม่ จากผลลัพธ์ตามตารางที่ 4.2 ชี้ให้เห็นว่าโมเดล CARMS MFทั้ง 4 ชุดข้อมูล ยังคงใช้จำนวนกลุ่มผู้ใช้มากกว่า 7 ทั้งหมด แสดงให้เห็นว่ามีพฤติกรรมลูกค้าที่ต่างกันและสามารถจัดกลุ่มได้ รวมถึงกลุ่มของกฎมีผลต่อความแม่นยำ ยกเว้นชุดข้อมูล Amazon Music ที่โมเดลเลือกใช้ K = 1 เท่านั้น (ตีความได้ว่ากลุ่มของผู้ใช้ไม่มีผลในชุดข้อมูลดังกล่าว)

4.2 จำนวน Cluster ของผู้ใช้

ในชุด Hyperparameter จะมีชุดที่ไม่ใช้ค่าต่ำสุดของ Support และ Confidence เลย (กำหนดให้เท่ากับ 0) เพื่อพิจารณาว่าการใช้กฎในลักษณะของ ARM เข้าไปกรองคู่ของความสัมพันธ์ (Co-occurrence) สามารถเพิ่มความแม่นยำได้หรือไม่ ซึ่งโมเดลทั้ง 5 ชุดข้อมูลเลือกใช้ค่าที่แตกต่างกันออกไป แต่ไม่มีชุดข้อมูลใดที่โมเดลเลือกค่าเป็น 0 เลย แสดงให้เห็นว่าการกรองกฎที่เกิดขึ้นน้อย (Noise) ออก จะช่วยให้ CARMS มีสัญญาณที่ดีขึ้น และเสริมให้โมเดลมีความแม่นยำ

4.3 ค่า Gamma

ในชุด Hyperparameter จะมีค่า Gamma ตั้งแต่ไม่เลือกใช้พจน์ BPR เลย จนถึง Gamma เท่ากับ 100 (ให้ Loss ในฝั่ง BPR มีผลมากกว่าฝั่ง SSE 100 เท่า) จากผลลัพธ์ค่อนข้างแสดงให้เห็นว่า CARMS มีผลต่อความแม่นยำของโมเดลจริง เนื่องจากทั้ง 5 ชุดข้อมูลโมเดลเลือกให้ Gamma = 100 ทั้งหมด แสดงให้เห็นถึงความสำคัญของพจน์ BPR ที่นำความมาเสริมต่อความแม่นยำ เป็นอีกหนึ่งตัวที่พิสูจน์ได้ว่า CARMS สามารถช่วยเพิ่มความแม่นยำให้กับโมเดลพื้นฐาน คือ Bias MF ได้อย่างมีนัยสำคัญจริง

4.4 เวลาในการคำนวณ

ความซับซ้อนของโมเดล (Complexity) ยังคงเป็นจุดอ่อนสำคัญของโมเดล CARMS MF เนื่องจากจะต้องเริ่มต้นสร้างสัญญาณ CARMS ก่อน และการแก้ไขปัญหาโดยใช้ BRP ร่วมเป็น Co-objective ก็ยิ่งทำให้โมเดลมีความซับซ้อนยิ่งขึ้น หากพิจารณาชุด Hyperparameter ที่ดีที่สุดของแต่ละโมเดล (ไม่สนใจ Hyperparameter ที่ต่างกัน) พบว่า โดยเฉลี่ย CARMS จะใช้ระยะเวลาในการฝึกฝนสูงกว่า 5.64 เท่า เทียบกับโมเดลพื้นฐานคือ Bias MF แต่ก็แลกมาด้วยความแม่นยำก็เพิ่มขึ้นมาก

ตารางที่ 4.1: ตารางสรุปผลของโมเดลพื้นฐานและ CARMS MF

Dataset	Model	NDCG@K					HR@K				
		@5	@10	@20	@50	@100	@5	@10	@20	@50	@100
Amazon Luxury	Bias MF	17.8%	18.2%	18.6%	19.3%	19.8%	19.6%	21.1%	22.7%	26.1%	29.5%
	CARMS Bias MF	25.7%	26.7%	27.3%	28.2%	28.9%	29.5%	32.4%	35.1%	39.3%	43.9%
Amazon Industry	Bias MF	4.4%	4.6%	4.9%	5.3%	5.6%	5.2%	6.1%	7.0%	9.0%	10.9%
	CARMS Bias MF	5.6%	6.3%	7.0%	7.8%	8.6%	7.0%	9.0%	11.8%	16.1%	20.8%
Amazon Pantry	Bias MF	0.5%	0.7%	0.9%	1.3%	1.8%	0.8%	1.4%	2.3%	4.4%	7.0%
	CARMS Bias MF	1.7%	2.1%	2.7%	3.6%	4.5%	2.6%	4.0%	6.2%	11.0%	16.4%
Amazon Music	Bias MF	1.9%	2.1%	2.4%	2.8%	3.2%	2.7%	3.4%	4.6%	6.5%	8.8%
	CARMS Bias MF	3.0%	3.6%	4.3%	5.3%	6.1%	4.4%	6.3%	9.1%	14.0%	19.2%
Amazon Instrument	Bias MF	3.9%	4.2%	4.4%	4.7%	4.9%	4.6%	5.5%	6.3%	7.8%	9.3%
	CARMS Bias MF	6.4%	7.0%	7.6%	8.3%	8.8%	8.0%	10.0%	12.3%	15.7%	19.0%

ตารางที่ 4.2: Hyperparameter (Randomized Search Iteration=200, Target=NDCG@10)

Dataset	Model	Latent	LR	Lambda	Epoch	K Cluster	Support	Confidence	Gamma	Time (Minutes)
Amazon Luxury	Bias MF	100	0.001	0.1	20	-	-	-	-	0.028115
	CARMS Bias MF	100	0.01	0.001	50	10	0.001	0.1	100	0.119709
Amazon Industry	Bias MF	100	0.001	0.05	20	-	-	-	-	0.124298
	CARMS Bias MF	30	0.001	0.005	100	7	0.0001	0.1	100	0.922929
Amazon Pantry	Bias MF	20	0.001	0.05	30	-	-	-	-	0.195214
	CARMS Bias MF	100	0.01	0.005	70	7	0.0001	0.1	100	0.955132
Amazon Music	Bias MF	100	0.001	0.001	30	-	-	-	-	0.397533
	CARMS Bias MF	100	0.01	0.01	50	1	0.0001	0.1	100	1.476288
Amazon Instrument	Bias MF	100	0.001	0.005	20	-	-	-	-	0.462623
	CARMS Bias MF	100	0.01	0.001	100	7	0.0001	0.1	100	3.669551

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้เริ่มต้นด้วยปัญหาของระบบแนะนำสินค้า (Recommendation System) ในปัจจุบัน คือ ปัญหาข้อมูลที่เบาบาง (Data Sparsity) ซึ่งทำให้โมเดลพื้นฐาน คือ Bias Matrix Factorization ไม่สามารถเรียนรู้ได้ดีนักเนื่องจากกลไกการเรียนรู้ของโมเดลอาศัยเฉพาะข้อมูลการให้คะแนนที่มีอยู่จริงในการเรียนรู้ ส่งผลกระทบโดยตรงต่อความแม่นยำของโมเดลที่ลดลง ซึ่งปัญหาดังกล่าวเป็นช่องว่างในงานวิจัย (Research Gap)

จุดประสงค์ของวิทยานิพนธ์ฉบับนี้คือการพัฒนาโมเดลการแยกตัวประกอบเมทริกซ์โดยใช้สัญญาณการจัดกลุ่มและการหากฎความสัมพันธ์ (Clustering Association Rule Mining Signal Matrix Factorization – CARMS MF) ที่สามารถสกัดสัญญาณของสินค้าชิ้นถัดไป (Signal) จากข้อมูล โดยใช้การจัดกลุ่ม (Clustering) และการหากฎความสัมพันธ์ (Association Rule Mining) โดยมีสมมุติฐานว่าโมเดลที่นำเสนอจะมีความแม่นยำที่สูงขึ้นเมื่อเทียบกับโมเดลพื้นฐานโดยไม่ใช้ข้อมูลคุณลักษณะ (Feature) เพิ่มเติมเลย (ซึ่งเป็นสิ่งที่งานวิจัยสายนี้นิยมทำ)

จากการทบทวนวรรณกรรมพบว่า มีความพยายามในการใช้กฎความสัมพันธ์หรือการหาการเกิดขึ้นร่วมกันของสินค้า (Co-occurrence) ในการพัฒนา MF ทั้งสิ้น 2 งาน โดยงานวิจัยของ Alsalama (2013) ที่ใช้ ARM เป็นหนึ่งในองค์ประกอบ (Module) หนึ่งของโครงสร้างโมเดล แต่ในส่วนของ Non-Favorite ก็ยังคงใช้ CBF ที่ใช้ข้อมูลคุณลักษณะ (Feature) เข้ามาเพิ่มเติม หรืองานวิจัยของ Liang et al., (2016) ที่เสนอแนวคิด Co-Factor โดยให้โมเดลเรียนรู้สัญญาณจาก SPII ไปพร้อมกับข้อมูลคะแนนของผู้ใช้ ซึ่งนำมาต่อยอดคือการใช้ Cluster ในการหา Group prior พร้อมกับการใช้ Indicator ของ ARM เพื่อกรอง Noise ออกจากชุดข้อมูล

วิทยานิพนธ์ฉบับนี้เสนอโครงสร้างที่จะปรับปรุงธุรกรรม (Augmented Transaction) ของผู้ใช้แต่ละรายโดยวิธีการจัดกลุ่ม (Clustering) ซึ่งสามารถสร้างกลุ่มของผู้ใช้และสินค้าได้โดยไม่ใช้ข้อมูลคุณลักษณะเพิ่มเติม (Mirbakhsh & Ling, 2015) และนำธุรกรรมที่ปรับปรุงแล้วไป หากฎความสัมพันธ์ด้วยแนวคิดของ ARM จากนั้นปรับปรุงฟังก์ชันเป้าหมายของโมเดลพื้นฐานด้วยแนวคิด Co-objective

ผลลัพธ์เมื่อเทียบกับโมเดลพื้นฐานแล้วถือว่าโมเดลที่วิทยานิพนธ์ฉบับนี้เสนอสามารถชนะโมเดลพื้นฐานได้ทั้ง 2 ตัวชี้วัด คือ NDCG@K และ HR@K ในทุกชุดข้อมูล และในทุกความยาวของคำแนะนำ โดยเฉพาะกับชุดข้อมูลที่มีความเบาบางของข้อมูลมาก คือ Amazon Industry และ Amazon Pantry ซึ่งเป็นสิ่งที่วิทยานิพนธ์ฉบับนี้ตั้งสมมุติฐานไว้ข้างต้น

5.2 ข้อจำกัดในงานศึกษา

แม้โมเดลจะมีความแม่นยำมากกว่าโมเดลพื้นฐานตามสมมุติฐานแต่ทั้งนี้ยังมีข้อจำกัด ดังนี้

1. จำนวน Hyperparameter: โมเดล CARMS MF จำเป็นต้องใช้ Hyperparameter เพิ่มขึ้นจาก 4 ตัว (ในโมเดลพื้นฐาน) เป็น 8 ตัว สิ่งนี้ทำให้จำนวนชุด Hyperparameter ที่เป็นไปได้ทั้งหมดเพิ่มขึ้นอย่างมาก จากบทที่ 3 ที่กล่าวถึงชุดของ Hyperparameter ที่ใช้ จะกำหนดเท่ากับ 5 ตัวในแต่ละ Hyperparameter ดังนั้นโมเดล Bias MF จะมีชุด Hyperparameter ที่เป็นไปได้ทั้งสิ้น 625 ชุด แต่ CARMS MF มีชุด Hyperparameter ทั้งสิ้น 390,625 ชุด ทำให้การหา Best hyperparameter ด้วย Randomized search จำนวน 200 รอบ สำหรับ CARMS จะยังไม่ครอบคลุมนัก
2. เวลาในการคำนวณ: โมเดล CARMS MF จะใช้เวลาในการคำนวณมากขึ้น (เฉลี่ย 5 เท่า โดยวัดจากชุด Hyperparameter ที่ได้ความแม่นยำสูงสุด) เนื่องจากจะต้องคำนวณ CARMS เพิ่มขึ้นมา และนอกจากนี้ Objective function ยังมีส่วนประกอบทั้งส่วน SSE เดิม ควบคู่ไปกับส่วน BPR

5.3 ข้อเสนอแนะสำหรับงานศึกษาในอนาคต

จากข้อจำกัดในงานศึกษา สามารถต่อยอดหรือปรับปรุงจากงานที่ได้นำเสนอ 3 ประการ ดังนี้

1. การต่อยอดการหา Hyperparameter ที่ดีที่สุด: เนื่องจากโมเดลมาตรฐานและ CARMS MF มีจำนวนของ Hyperparameter ที่แตกต่างกัน แต่งานศึกษาปัจจุบันใช้Randomized Search เท่ากับ 200 รอบ เนื่องจากข้อจำกัดของการประมวลผล ซึ่งหากดูปริมาณสัดส่วนต่อความเป็นไปได้ของ Hyperparameter พบว่าโมเดลมาตรฐานถูกพิจารณาชุดของ Hyperparameter เท่ากับ 32% ของชุด Hyperparameter ทั้งหมด แต่สำหรับ CARMS พิจารณาไปเพียง 0.05% ดังนั้นอาจกล่าวได้ว่า CARMS MF อาจมีประสิทธิภาพเพิ่มขึ้นได้อีก หากเปลี่ยนวิธีการหา Hyperparameter ที่ดีที่สุด

2. การต่อยอดในส่วนของการจัดกลุ่ม (Clustering): จากงานศึกษาของ Mirbakhsh & Ling (2015) ผลจากอัลกอริทึมการจัดกลุ่มที่ต่างกันได้ผลลัพธ์ที่แตกต่างกัน ดังนั้นในงานศึกษาในอนาคตสามารถลองปรับวิธีการจัดกลุ่มข้อมูลได้ โดยอาจเปลี่ยนไปใช้โมเดลอื่นๆ เพื่อทดสอบผล เช่น โมเดลในตระกูลใช้ความหนาแน่น (Density-based) อาทิ DBSCAN หรืออาจเลือกใช้เทคนิคการจัดกลุ่มแบบอัตโนมัติ (Auto Clustering) ก็จะเป็นการลด Hyperparameter ลงได้อีก
3. การต่อยอดสถาปัตยกรรม: CARMS MF ยังคงใช้ Matrix Factorization ซึ่งในปัจจุบันสถาปัตยกรรมนี้ได้มีการต่อยอดโดยใช้การเรียนรู้เชิงลึก (Deep Learning) คือ Neural Collaborative Filtering (NCF) ซึ่งสามารถนำสัญญาณ CARMS ไปปรับปรุง NCF ได้

ภาคผนวก ก

ขั้นตอนการคำนวณ

1. สร้างสัญญาณ CARMS

1.1 กำหนด Cold user และ Warm user

กำหนดให้ชุดข้อมูลมีจำนวนผู้ใช้ 4 ราย (4 Rows) และมีจำนวนสินค้า 5 ชิ้น (5 Columns) ดังนั้น User-Item Matrix (Y) จะมีขนาดเท่ากับ 3x3 แทนค่าว่างด้วย 0 ซึ่งหมายถึง สินค้าที่ผู้ใช้ยังไม่ได้ลงคะแนน

$$Y = \begin{bmatrix} 5 & 0 & 3 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

จากนั้นสร้าง Binary matrix ของคะแนนที่ผู้ใช้ได้ให้คะแนน โดยกำหนดให้เท่ากับ 1 หากเคยให้คะแนน และให้เท่ากับ 0 หากไม่เคยให้คะแนนกับสินค้าชิ้นนั้น

$$O = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2)$$

นับจำนวนของสินค้าที่ผู้ใช้เคยให้คะแนน ซึ่งจะเห็นว่าผู้ใช้คนสุดท้ายไม่ได้ให้คะแนนกับสินค้าใดเลย

$$\text{obs\_cnt} = [2, 2, 2, 0] \quad (3)$$

สุดท้าย ใช้เงื่อนไข $obs\_cnt > 0$ เพื่อแบ่งกลุ่มที่เป็น Warm user และ Cold user

$$\text{warm\_users} = [0, 1, 2] \quad (4)$$

$$\text{cold\_users} = [3] \quad (5)$$

1.2 สร้าง Liked/Disliked Matrix

Liked/Disliked matrix จะเป็นตัวกำหนด Preference ของผู้ใช้รายคนเพื่อนำมาทำ Clustering เริ่มจากหา Liked threshold ด้วยค่า Median ของการให้คะแนนของผู้ใช้รายคน

$$\text{med} = [4, 3, 3, 0] \tag{6}$$

จากนั้นแบ่งความชอบตามการให้คะแนน หาก $Y \geq median$ จะถือว่าชอบและกำหนดให้เท่ากับ 1 หาก $Y < median$ จะถือว่าไม่ชอบและกำหนดให้เท่ากับ -1 และหากไม่เคยลงคะแนนจะกำหนดให้เท่ากับ 0

$$Z = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

เนื่องจากการทำ Cluster จะทำบน Warm user เท่านั้น ดังนั้นจะตัด Cold User ออก

$$Z = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

จากนั้นทำ Cluster บนกลุ่มลูกค้าที่เป็น Warm user หากสมมุติให้ $K = 2$ ดังนั้นผู้ใช้คนที่ 1 และคนที่ 2 จะได้ Cluster ที่ 0 ส่วนผู้ใช้คนที่ 3 จะได้ Cluster ที่ 1 และผู้ใช้คนสุดท้ายที่เป็น Cold user จะยังไม่มี Cluster

$$\text{labels\_all} = [0, 0, 1] \tag{9}$$

จากนั้น กำหนด Cluster ให้กับผู้ใช้คนสุดท้ายที่เป็น Cold start (สมมุติให้มีระยะทางใกล้กับ Cluster ที่ 0) ดังนั้นตอนนี้ผู้ใช้ทุกคนจะมี Cluster label แล้ว โดย Cold user จะไปอยู่ที่ Cluster เดียวกันทั้งหมด ซึ่งเป็น Cluster ที่ Centroid มีลักษณะเป็นกลาง เช่น กลุ่มที่ผู้ใช้ให้คะแนน Ratings ที่ชอบและไม่ชอบหักล้างกันพอดี หรือสำคัญที่สุดคือกลุ่มที่ให้คะแนนกับสินค้าไม่มากชิ้น

$$\text{labels\_all} = [0, 0, 1, 0] \tag{10}$$

1.3 ระบุ Active Items

การทำ ARM นั้นจะใช้เฉพาะ Active Items กล่าวคือสินค้าที่มีผู้ใช้ให้คะแนนอย่างน้อย 1 ครั้งขึ้นไป พิจารณาจาก Y พบว่าสินค้าทั้ง 5 ชิ้นจัดเป็น Active Items ทั้งหมด เนื่องจากมีผู้ลงคะแนนให้ทุกสินค้า ดังนั้นแล้ว $I_a = 5$

$$\text{active\_items}=[0, 1, 2, 3, 4] \tag{11}$$

1.4 สร้าง Augmented Transaction

เมื่อได้ Cluster (ที่เรากำหนดให้ $K = 2$) และ Active Item (Active ทุกชิ้น) และ Warm User แล้ว จะนำทั้งหมดมาสร้าง Augmented Transaction ของผู้ใช้แต่ละคน โดยนอกจากสินค้าแล้ว จะมีการเติม Cluster label เข้าไป ดังนั้นเมื่อ Active Item = 5 และ Cluster Label = 2 และ Warm User = 3 ขนาดของ Augmented Transaction Matrix จะมีขนาดเท่ากับ 3x7 ซึ่ง 2 Columns สุดท้ายคือ One-hot ของทั้ง 2 Cluster ที่เติมเพิ่มเข้ามา

$$\left[\begin{array}{ccccc|cc} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array}\right] \tag{12}$$

1.5 สร้าง ARM

จากนั้นจะเริ่มสร้างส่วนประกอบของ ARM โดยเริ่มจากนับจำนวนสินค้าที่เกิดร่วมกัน ($A \cap B$) โดยเริ่มจากกำหนด Augmented Transaction Matrix เท่ากับ X การใช้ Augmented Transaction Matrix จะทำให้ได้กฎที่ออกมาในลักษณะทั้ง Item-to-Item และ Cluster-to-Item

$$X = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{13}$$

จากนั้นทำ $X^T X$ แล้วจะได้จำนวนสินค้าร่วมกัน ($A \cap B$) ซึ่งมีขนาด 7x7

$$\text{co\_full}=\begin{bmatrix} 2 & 0 & 1 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 2 & 0 & 1 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad 14$$

เนื่องจากกฎ Item-to-Cluster เป็นกฎที่ไม่มีประโยชน์ในการสร้างกฎ ดังนั้นจะตัด 2 Columns สุดท้าย

$$\text{co\_all}=\begin{bmatrix} 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad 15$$

หาสูตรของ Support คือ $P(A \cap B)/N$ ซึ่งใช้ผลจาก $X^T X$ ส่วนด้วยจำนวน Augmented Transaction เท่ากับ 3 (เท่ากับจำนวน Warm User เนื่องจากเรากำหนดให้ประวัติของ 1 คน เท่ากับ 1 Transaction พอดี)

$$N_f = 3 \quad 16$$

จากนั้นเข้าสูตรคำนวณ Support ตามที่ได้กล่าวไปข้างต้น

$$\text{sup\_all}=\begin{bmatrix} 0.6667 & 0 & 0.3333 & 0.3333 & 0 \\ 0 & 0.3333 & 0 & 0 & 0.3333 \\ 0.3333 & 0 & 0.3333 & 0 & 0 \\ 0 & 0.3333 & 0 & 0 & 0.3333 \\ 0.6667 & 0 & 0.3333 & 0.3333 & 0 \\ 0 & 0.3333 & 0 & 0 & 0.3333 \end{bmatrix} \quad 17$$

จากนั้นนับจำนวนผู้ใช้ที่ลงคะแนนให้สินค้า หรือเรียกอีกอย่างว่า $P(A)$ โดยสาเหตุที่ใช้เฉพาะ $P(A)$ เนื่องจากสูตร Confidence มาจาก $sup(A \cap B)/sup(A)$ แต่เนื่องจาก $sup(A \cap B)$ คำนวณจาก $P(A \cap B)/N$ และ $sup(A)$ คำนวณจาก $P(A)/N$ แต่เนื่องจากเมื่อนำมาหารกันแล้ว N จะตัดกันได้พอดี ดังนั้นสูตรจึงเป็น $P(A \cap B)/P(A)$ ในการหา Confidence

$$\text{pop\_all}=[2,1,1,1,1,2,1] \tag{18}$$

และนำไปเข้าสูตรเพื่อคำนวณ Confidence ตามที่ได้กล่าวไปข้างต้น

$$\text{conf\_all}=\begin{bmatrix}1 & 0 & 0.5 & 0.5 & 0\\ 0 & 1 & 0 & 0 & 1\\ 1 & 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 1\\ 1 & 0 & 0.5 & 0.5 & 0\\ 0 & 1 & 0 & 0 & 1\end{bmatrix} \tag{19}$$

ต่อไปคือการคำนวณ Lift แต่เนื่องจาก Lift มี Component เพิ่มเติมเข้ามา คือ $P(B)$ ซึ่งหาจาก $P(B)/N$ ดังนั้นจะต้องนับจำนวน B โดยนับว่าแต่สินค้ามีผู้ใช้ให้ Ratings กี่ครั้ง และหารด้วย Warm User จะสังเกตว่าขนาดของ $P(B)$ จะไม่เท่ากับ $P(A)$ โดยต่างกัน 2 Members เนื่องจากจะตัดกฎทางขวามือ (B) ซึ่งเป็น Item-to-Cluster

$$p_{\text{item}}=\frac{\text{pop}_{\text{item}}}{N_f}=\left[\frac{2}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3}\right]=\left[0.6667,0.3333,0.3333,0.3333,0.3333\right] \tag{20}$$

นำไปเข้าสูตร $conf(A \cap B)/P(B)$ เพื่อหา Lift

$$lift\_raw=\begin{bmatrix}1.5 & 0 & 1.5 & 1.5 & 0\\ 0 & 3 & 0 & 0 & 3\\ 1.5 & 0 & 3 & 0 & 0\\ 1.5 & 0 & 0 & 3 & 0\\ 0 & 3 & 0 & 0 & 3\\ 1.5 & 0 & 1.5 & 1.5 & 0\\ 0 & 3 & 0 & 0 & 3\end{bmatrix} \tag{21}$$

ค่า Lift จะถูกปรับ Scale ด้วยการใช้ Square Root เพื่อลดขนาดของ Lift ที่สูงมากเกินไปจากสินค้าที่เป็น Rare Item ลดโอกาสที่จะทำให้ Gradient มีขนาดใหญ่เกินไป จากขนาดของ Loss

$$\sqrt{\text{lift}} = \begin{bmatrix} 1.2247 & 0 & 1.2247 & 1.2247 & 0 \\ 0 & 1.7321 & 0 & 0 & 1.7321 \\ 1.2247 & 0 & 1.7321 & 0 & 0 \\ 1.2247 & 0 & 0 & 1.7321 & 0 \\ 0 & 1.7321 & 0 & 0 & 1.7321 \\ 1.2247 & 0 & 1.2247 & 1.2247 & 0 \\ 0 & 1.7321 & 0 & 0 & 1.7321 \end{bmatrix} \quad 22$$

จากนั้นจะกำหนด $min\_support$ และ $min\_confidence$ เท่ากับ 0 (ไม่กรองอะไรเลย) ดังนั้นเราจะสร้าง Mask Matrix มา และกำหนดค่าเท่ากับ 1 หาก Support และ Confidence มีค่ามากกว่าหรือเท่ากับ 0 ในช่องนั้นๆ (ซึ่งจะเห็นได้ว่า เป็น 1 ทุกช่องที่มีค่า)

$$mask = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \quad 23$$

จากนั้นสร้าง Weight Matrix (W) โดยสูตรจะใช้ Confidence คูณกับ Lift โดย Confidence จะแสดงถึงความน่าจะเป็นของกฎนั้นๆ และ Lift จะเป็นตัวดึงค่าขึ้น/ ลง หากกฎเหล่านั้นเป็นกฎที่มีความเฉพาะ ไม่ใช่เป็นเพียงสินค้าที่มีความนิยม และ Mask เพื่อกรองค่า $min\_support$ และ $min\_confidence$ ไม่ถึงเกณฑ์ที่กำหนดออกด้วย เพราะถือว่าเป็น Noise (แต่ในตัวอย่างไม่กรอง)

$$W_{ab} = \text{conf}(a \to b) \times \tilde{\ell}(a \to b) \times M_{ab} \quad 24$$

แทนค่าที่ได้เข้าไป รวมถึง Mask ด้วย

$$W = \begin{bmatrix} 0 & 0 & 0.6124 & 0.6124 & 0 \\ 0 & 0 & 0 & 0 & 1.7321 \\ 1.2247 & 0 & 0 & 0 & 0 \\ 1.2247 & 0 & 0 & 0 & 0 \\ 0 & 1.7321 & 0 & 0 & 0 \\ 1.2247 & 0 & 0.6124 & 0.6124 & 0 \\ 0 & 1.7321 & 0 & 0 & 1.7321 \end{bmatrix} \quad 25$$

จากนั้นจะ Assign W ให้กับผู้ใช้รายคน โดยเริ่มจาก Warm User จะดึงคะแนนไปใช้โดยเทียบกับ Augmented Transaction Matrix ของตนเองว่าเคยลงคะแนนให้กับสินค้าชิ้นใดบ้างหรืออยู่ใน Cluster ใดบ้าง และมีความสัมพันธ์กับสินค้าชิ้นใดต่อไป

$$S_{\text{warm},a} = \begin{bmatrix} 2.4495 & 0 & 1.2247 & 1.2247 & 0 \\ 2.4495 & 0 & 1.2247 & 1.2247 & 0 \\ 0 & 3.4641 & 0 & 0 & 3.4641 \end{bmatrix} \quad 26$$

ส่วนผู้ใช้ที่เป็น Cold User สามารถดึงเอาคะแนนของกลุ่ม Cluster-to-Item 1, 2, 3, 4 และ 5 ไปใช้ได้เลย เนื่องจาก Cold User อยู่ในกลุ่มเดียวกัน คือ Cluster 0 ทั้งหมด

$$S_{cold,a} = [1.2247, 0, 0.6124, 0.6124, 0] \quad 27$$

จากนั้นรวมทั้ง Warm User และ Cold User เข้าด้วยกัน จะได้ CARMS Matrix ซึ่งจะบอกถึงสัญญาณของสินค้าชิ้นถัดไปที่ผู้ใช้มีแนวโน้มจะซื้อต่อ ทั้งนี้สัญญาณจะยังซ้ำอยู่กับสินค้าที่เคยให้คะแนนแล้ว (Seen Items)

$$S_{\text{before}} = \begin{bmatrix} 2.4495 & 0 & 1.2247 & 1.2247 & 0 \\ 2.4495 & 0 & 1.2247 & 1.2247 & 0 \\ 0 & 3.4641 & 0 & 0 & 3.4641 \\ 1.2247 & 0 & 0.6124 & 0.6124 & 0 \end{bmatrix} \quad 28$$

และจะตัดสินค้าที่ผู้ใช้เคยเห็นออก เนื่องจากจะใช้สัญญาณเสริมกับสินค้าที่ผู้ใช้ยังไม่เคยเห็นเท่านั้น

$$\text{rated} = \begin{bmatrix} \text{True} & \text{False} & \text{True} & \text{False} & \text{False} \\ \text{True} & \text{False} & \text{False} & \text{True} & \text{False} \\ \text{False} & \text{True} & \text{False} & \text{False} & \text{True} \\ \text{False} & \text{False} & \text{False} & \text{False} & \text{False} \end{bmatrix} \quad 29$$

สุดท้ายจะได้ CARMS ที่จะนำไปใช้สอนโมเดล

$$S_{\text{final}} = \begin{bmatrix} 0 & 0 & 0 & 1.2247 & 0 \\ 0 & 0 & 1.2247 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1.2247 & 0 & 0.6124 & 0.6124 & 0 \end{bmatrix} \quad 30$$

3. Model training

จาก Objective Function จะมีทั้งพจน์ที่เป็น SSE และ BPR และสมมุติว่าไม่มี Regularization term เพื่อความง่ายในการแสดงตัวอย่างการคำนวณ

$$\mathcal{L}_{co} = \sum_{y_{ui}>0} (y_{ui} - \hat{y}_{ui})^2 + \gamma \sum_{y_{ui}=0, s_{ui}>0} [-w_{ui} \log \sigma(\hat{y}_{ui} - \hat{y}_{uj})] \quad 31$$

ดังนั้น จะต้องมี Input 2 ส่วน คือ Y (User-Item Matrix) ซึ่งจะเป็นสิ่งที่โมเดลใช้เรียนรู้หลัก และ S (CARMS) ซึ่งเป็นสิ่งที่จะช่วยเสริมการเรียงลำดับของสินค้าที่ผู้ใช้ไม่เคยเห็น

$$Y = \begin{bmatrix} 5 & 0 & 3 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad 32$$

$$S_{\text{final}} = \begin{bmatrix} 0 & 0 & 0 & 1.2247 & 0 \\ 0 & 0 & 1.2247 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1.2247 & 0 & 0.6124 & 0.6124 & 0 \end{bmatrix} \quad 33$$

3.1 การกำหนดค่าเริ่มต้น

การแก้ปัญหาจะใช้ Full-batch Gradient Descent โดยเริ่มจากการกำหนดค่าเริ่มต้นของ Parameter ของ P และ Q ด้วยค่า Random เล็กๆ

$$P = \begin{bmatrix} 0.02 & -0.01 \\ 0.00 & 0.01 \\ -0.01 & 0.02 \\ 0.01 & 0.00 \end{bmatrix} \quad 34$$

$$Q = \begin{bmatrix} 0.01 & 0.00 \\ -0.02 & 0.01 \\ 0.00 & 0.02 \\ 0.01 & -0.01 \\ -0.01 & 0.00 \end{bmatrix} \quad 35$$

ค่าเริ่มต้นของ User Bias และ Item Bias จะกำหนดให้เท่ากับ 0

$$b_u = 0(4\times 1), \qquad b_i = 0(5\times 1) \tag{36}$$

ส่วนค่าเริ่มต้น Global Bias เราจะใช้ค่าเฉลี่ยของคะแนนมาเป็นค่าเริ่มต้น เป็นเทคนิคในการกำหนดค่าเริ่มต้นของ Global Bias เพื่อช่วยให้โมเดลสามารถเรียรู้ได้ไวยิ่งขึ้น

$$\text{observed} = \{5,3,4,2,1,5\} \Rightarrow \mu = \frac{20}{6} = 3.3333 \tag{37}$$

3.2 ทำนายค่า

จากนั้นเริ่มทำนายค่า Y ของรอบแรกสุดก่อน ตามสมการ

$$\hat{y}_{ui} = \mu + b_u(u) + b_i(i) + p_u^T q_i \tag{38}$$

เมื่อแทนค่าเริ่มต้นทั้งหมด จะได้

$$\hat{Y} \approx \begin{bmatrix} 3.3335 & 3.3328 & 3.3331 & 3.3336 & 3.3331 \\ 3.3333 & 3.3334 & 3.3335 & 3.3332 & 3.3333 \\ 3.3332 & 3.3337 & 3.3337 & 3.3330 & 3.3334 \\ 3.3334 & 3.3331 & 3.3333 & 3.3334 & 3.3332 \end{bmatrix} \tag{39}$$

3.3 คำนวณความคลาดเคลื่อนของพจน์หน้า (SSE Loss)

ตั้งต้นจากค่า Observed Ratings

$$Y = \begin{bmatrix} 5 & 0 & 3 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{40}$$

จากนั้นคำนวนส่วนต่างของ $Y - \hat{Y}$ ในทุกจุดที่ Observed ดังนี้

1. สำหรับจุด y_{11} แล้ว ค่า Error เท่ากับ $(5 - 3.33335)^2 = 2.7777$
2. สำหรับจุด y_{21} แล้ว ค่า Error เท่ากับ $(4 - 3.3333)^2 = 0.4445$
3. สำหรับจุด y_{32} แล้ว ค่า Error เท่ากับ $(1 - 3.3337)^2 = 5.4452$
4. สำหรับจุด y_{13} แล้ว ค่า Error เท่ากับ $(3 - 3.3331)^2 = 0.1110$
5. สำหรับจุด y_{24} แล้ว ค่า Error เท่ากับ $(2 - 3.3332)^2 = 1.7774$
6. สำหรับจุด y_{35} แล้ว ค่า Error เท่ากับ $(5 - 3.3334)^2 = 2.7776$

สุดท้าย Loss ของทุก Observed Ratings เข้าไว้ด้วยกัน รวมคำนวณจาก 2.7777 + 0.4445 + 5.4452 + 0.1110 + 1.7774 + 2.7776 แล้ว SSE = 13.3338

3.4 คำนวณความคลาดเคลื่อนของพจน์หลัง (BPR Loss)

จากนั้นจะหาความคลาดเคลื่อนของพจน์ BPR โดยเริ่มจากจะต้องหยิบสุ่ม Positive Item ขึ้นมาผู้ใช้ละ 1 Items โดยนิยามของ Positive Items คือ $Y = 0 \,\&\, S > 0$ ความหมายคือ หาก $S > 0$ แปลว่าแม้ผู้ใช้จะไม่เคยใช้ แต่ CARMS บอกว่า ผู้ใช้มีแนวโน้มที่จะใช้สินค้าชิ้นนั้นๆ เป็นสินค้าชิ้นถัดไป และจะใช้ BPR ในการ Re-rank

$$pos\_mask = (Y = 0) \,\&\, (S > 0) \qquad 11$$

เริ่มพิจารณาจาก Positive Items

1. User0 pos = {Item3}
2. User1 pos = {Item2}
3. User2 pos = {}
4. User3 pos = {Item0, item2, Item3}

จากนั้นสุ่มให้เหลือแค่ผู้ใช้และ 1 ด้วย Binary Distribution (ทุกตัวได้ความน่าจะเป็นเท่ากันหมด)

1. User0 pos = {Item3}
2. User1 pos = {Item2}
3. User2 pos = {Item0}

จากนั้นทำการสุ่ม Negative Items $Y = 0 \,\&\, S = 0$ ซึ่ง Negative Items จะเป็นสินค้าที่ควรจะจัดอันดับให้อยู่ต่ำกว่าโดยเปรียบเทียบกับ Positive Items

$$neg\_mask = (Y = 0) \,\&\, (S = 0) \qquad 41$$

เจากนั้นสุ่มให้เหลือแค่ผู้ใช้และ 1 ด้วย Binary Distribution (ทุกตัวได้ความน่าจะเป็นเท่ากันหมด)

1. User0 pos = {Item1 item 4}
2. User1 pos = {Item1, item4}
3. User3 pos = {Item1 item 4}

จากนั้นสุ่มให้เหลือแค่ผู้ใช้และ 1 ด้วย Binary Distribution เช่นเดียวกัน

4. User0 pos = {Item1}
5. User1 pos = {item4}
6. User3 pos = {item 4}

และคำนวณส่วนต่างของคะแนนระหว่าง Positive และ Negative ซึ่งจะเห็นได้ว่า Ranking ถูกทั้งหมด แต่ค่าอาจจะใกล้กันมากเกินไป

7. สำหรับจุด y_{11} แล้ว ค่า Error เท่ากับ $\log\ \sigma(3.333633 - 3.332833)$
8. สำหรับจุด y_{21} แล้ว ค่า Error เท่ากับ $\log\ \sigma(3.3334 - 3.3332)$
9. สำหรับจุด y_{32} แล้ว ค่า Error เท่ากับ $\log\ \sigma(3.3334 - 3.3332)$

จากนั้นนำค่าไปผ่านสมการ และคูณด้วย W ซึ่งเป็นน้ำหนัก และนำค่ามารวมกันทั้งหมด จะได้ BPR Loss = 2.5460

3.5 คำนวณความคลาดเคลื่อนรวม

สุดท้ายแล้วจะนำไปรวมกับ SSE Loss โดยมีตัวปรับความสำคัญของพจน์ BPR อีกทีหนึ่ง ซึ่ง Gamma จะเป็น Hyperparameter ที่จะต้องปรับเพื่อบอกว่าจะให้ BPR มีอิทธิพลกับภาพรวมของโมเดลเท่าใด (กำหนด 0.1)

$$\text{loss} = 13.3338 + 0.1(2.5460) = 13.5884 \tag{41}$$

3.6 คำนวณ Gradient

คำนวณอนุพันธ์ต่อพารามิเตอร์ทุกตัว

$$\frac{\partial \mathcal{L}}{\partial p_u}, \frac{\partial \mathcal{L}}{\partial q_i}, \frac{\partial \mathcal{L}}{\partial b_u}, \frac{\partial \mathcal{L}}{\partial b_i}, \frac{\partial \mathcal{L}}{\partial \mu} \tag{41}$$

เนื่องจากใช้วิธี Full-batch Gradient Descent และเพื่อความเข้าใจง่ายจึงไม่พิจารณาเทอม regularization ในขั้นตอนนี้ ดังนั้นในแต่ละรอบการเรียนรู้ เราจะ คำนวณ Gradient ของ Loss function จาก Observed ทั้งหมดทั้งหมด (ทุกตำแหน่งที่ $Y_{ui} \neq 0$) รวมถึงส่วน BPR จากคู่ตัวอย่าง

$$\nabla_{\mu}^{\text{SSE}} = 0.000800 \tag{42}$$

$$\nabla_{b_u}^{\text{SSE}} = \left[-2.666667, 1.333133, 1.334333, 0\right] \tag{43}$$

$$\nabla_{b_i}^{\text{SSE}} = \left[-4.666267, 4.667467, 0.666267, 2.666467, -3.333133\right] \tag{44}$$

$$\nabla_{P}^{\text{SSE}} = \begin{bmatrix} -0.033329 & 0.013325 \\ 0.013331 & -0.026665 \\ -0.060018 & 0.046675 \\ 0 & 0 \end{bmatrix} \tag{45}$$

$$\nabla_{Q}^{\text{SSE}} = \begin{bmatrix} -0.066659 & 0.019996 \\ -0.046675 & 0.093349 \\ 0.013325 & -0.006663 \\ 0 & 0.026665 \\ 0.033331 & -0.066663 \end{bmatrix} \tag{46}$$

3.7 Update ค่า

และสมการที่ใช้อัพเดท กำหนด Learning rate = 0.05

$$\theta \leftarrow \theta - \mathrm{lr}\nabla_{\theta} \tag{47}$$

เมื่อบวกเข้าไปแล้ว (ตามขนาด Learning rate แล้ว) จะได้ค่า Parameter ใหม่ คือ

$$\mu' = 3.333333 - 0.05(0.000800) = 3.333293 \tag{48}$$

$$b_{u'} = \left[0.133333, -0.066657, -0.066717, 0\right] \tag{49}$$

$$b_{i'} = \left[0.236375, -0.236434, -0.030252, -0.130263, 0.160534\right] \tag{50}$$

$$P' = \begin{bmatrix} 0.021748 & -0.010722 \\ -0.000636 & 0.011389 \\ -0.006994 & 0.017656 \\ 0.010056 & 0.000000 \end{bmatrix} \tag{51}$$

$$Q' = \begin{bmatrix} 0.013359 & -0.001000 \\ -0.017717 & 0.005358 \\ -0.000666 & 0.020354 \\ 0.010056 & -0.011359 \\ -0.011692 & 0.003303 \end{bmatrix} \tag{52}$$

จบการ Update หนึ่งรอบ และจากนั้นก็จะวนซ้ำตั้งแต่ขั้นตอนที่ 3.2 จนถึง 3.7 ตามรอบที่กำหนด