# 效益型樣探勘與軟體操作

國立宜蘭大學資訊工程系
吳政瑋 助理教授
wucw@niu.edu.tw

# 本堂教學重點

- 傳統頻繁項目集探勘缺點
- 效益型樣探勘目的
- 效益型樣探勘技術
- 效益型樣應用
- 效益型樣探勘軟體操作

# 傳統頻繁項目集探勘缺點 (Cont. 1/2)

- 頻繁項目集探勘(Frequent Itemset Mining)是資料探勘領域中一門相當重要的技術。
- 該技術常用於**購物籃分析(Market Basket Analysis)**的應用中。

# 傳統頻繁項目集探勘缺點 (Cont. 2/2)

- 在**購物籃分析**的真實應用中，傳統 FIM 架構往往
  - 找出過多頻繁發生但低獲利的商品組合
  - 遺失不常發生但獲利高的商品組合

- 傳統 FIM 架構
  - 沒有考慮商品的**購買數量(Purchased Quantity)**及**單位利潤(Unit Profit)**
  - 無法計算出商品組合的獲利
  - 無法滿足想找出高獲利商品組合的用戶的需求

# 高效益項目集探勘架構 (High Utility Itemset Mining Framework)

**Utility of an item in a transaction**

$$u(\{A\}, T_1) = 1 \times 5 = 5$$

**Utility of an itemset in a transaction**

$$u(\{AD\}, T_1) = u(\{A\}, T_1) + u(\{D\}, T_1) = 5 + 2 = 7$$

**Utility of an itemset in a database**

$$u(\{AD\}) = u(\{AD\}, T_1) + u(\{AD\}, T_3) = 7 + 17 = 24$$

| Transactional Table | |
|---|---|
| **TID** | **Transaction** |
| $T_1$ | (A, 1) (C, 1) (D, 1) |
| $T_2$ | (A, 2) (C, 6) |
| $T_3$ | (A, 1) (B, 2) (C, 1) (D, 6) |
| $T_4$ | (B, 4) (C, 3) (D, 3) |
| $T_5$ | (B, 2) (C, 2) |

| Profit Table | | | | |
|---|---|---|---|---|
| **Item** | A | B | C | D |
| **Profit** | 5 | 2 | 1 | 2 |

**High utility itemset (HUI)**

If $u(X) \geq$ *minimum utility threshold* (*min_util*) then $X$ is called HUI

Otherwise, $X$ is a low utility itemset

E.g., *min_util* = 30, $u(\{BD\}) = 30$ and $u(\{B\}) = 16$

→{BD} is a HUI but {B} is not

# 傳統頻繁項目集探勘缺點 (Cont. 3/5)

- 以下列的交易資料庫為例，當最小效益門檻值 *min_util* = 30，則所有的HUI有{AC}、{BD}、{BCD}。

| Transactional Table | |
|---|---|
| **TID** | **Transaction** |
| $T_1$ | (A, 1) (C, 1) (D, 1) |
| $T_2$ | (A, 2) (C, 6) |
| $T_3$ | (A, 1) (B, 2) (C, 1) (D, 6) |
| $T_4$ | (B, 4) (C, 3) (D, 3) |
| $T_5$ | (B, 2) (C, 2) |

| Profit Table | | | | |
|---|---|---|---|---|
| **Item** | A | B | C | D |
| **Profit** | 5 | 2 | 1 | 2 |

# 高效益型樣探勘的主要挑戰

- 效益項目集不遵從**向下封閉性(Downward Closure Property；DC Property)**
  - 一個 LUI 的 Superset 可能為 HUI
  - 一個 HUI 的 Subset 可能為 LUI

- 這意味著高效益型樣探勘演算法無法利用傳統的 DC Property 有效縮減搜尋空間。

# 以暴力法探勘 HUI

- **A Brute Force Approach for HUI Mining**
  - 列舉出資料庫中所有 Itemset，並掃描資料庫計算每個 Itemset 的效益值。

- **暴力法的主要缺點**
  - **空間問題：**若將所有 Itemset 儲存於 Memory 中，則需耗費大量空間。
  - **時間問題：**搜尋 Itemset 的位置、比對 Itemset 耗費大量執行時間。

# Transaction-Weighted Utilization Downward Closure Property

- 為了有效率探勘高效益項目集，有學者[1]提出 ***Transaction-Weighted Utilization Downward Closure Property (TWDC Property)***

- TWDC Property 乃基於下列定義設計而成
  - ***Transaction Utility (TU)***
  - ***Transaction-Weighted Utilization (TWU)***
  - ***High TWU Itemset (HTWUI)***

# Transaction Utility (TU)

- **Transaction Utility**
  - $TU(T_1) = u(\{A\}, T_1) + u(\{C\}, T_1) + u(\{D\}, T_1)$
    $= 5 + 1 + 2 = 8$
  - $TU(T_2) = 10 + 6 = 16$
  - $TU(T_3) = 5 + 4 + 1 + 12 = 22$
  - $TU(T_4) = 8 + 3 + 6 = 17$
  - $TU(T_5) = 4 + 2 = 6$

| Transactional Table | |
|---|---|
| **TID** | **Transaction** |
| $T_1$ | (A, 1) (C, 1) (D, 1) |
| $T_2$ | (A, 2) (C, 6) |
| $T_3$ | (A, 1) (B, 2) (C, 1) (D, 6) |
| $T_4$ | (B, 4) (C, 3) (D, 3) |
| $T_5$ | (B, 2) (C, 2) |

| Profit Table | | | | |
|---|---|---|---|---|
| **Item** | A | B | C | D |
| **Profit** | 5 | 2 | 1 | 2 |

# 轉換交易資料庫
# 並計算每筆交易的Transaction Utility

| Transactional Table | |
|---|---|
| **TID** | **Transaction** |
| $T_1$ | (A, 1) (C, 1) (D, 1) |
| $T_2$ | (A, 2) (C, 6) |
| $T_3$ | (A, 1) (B, 2) (C, 1) (D, 6) |
| $T_4$ | (B, 4) (C, 3) (D, 3) |
| $T_5$ | (B, 2) (C, 2) |

| Transactional Table | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

| Profit Table | | | | |
|---|---|---|---|---|
| **Item** | A | B | C | D |
| **Profit** | 5 | 2 | 1 | 2 |

# Transaction-Weighted Utilization (TWU)

- **Transaction-Weighted Utilization (TWU)**
  - $TWU(\{A\}) = TU(T_1) + TU(T_2) + TU(T_3)$
    $= 8 + 16 + 22 = 46$
  - $TWU(\{AC\}) = TU(T_1) + TU(T_2) + TU(T_3) = 46$
  - $TWU(\{AD\}) = TU(T_1) + TU(T_3) = 8 + 22 = 30$

| Transactional Table | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# High TWU Itemset (HTWUI)

- An itemset $X$ is called ***high TWU itemset*** if $TWU(X) \geq min\_util$；Otherwise, $X$ is called ***low TWU itemset***.
  - 若 $min\_util = 30$,
    - {AC} is a **high TWU itemset**, since $TWU(\{AC\}) = 46$
    - {ABC} is a **low TWU itemset**, since $TWU(\{ABC\}) = 22$

| Transactional Table | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Transaction-Weighted Utilization Downward Closure Property

- **TWDC Property**
  - 若一個 Itemset 為 Low TWU Itemset，則它所有 Superset 皆為 Low Utility Itemset。
  - 若一個 Itemset 為 High TWU Itemset，則它所有 Subset 皆為 High TWU Itemset。

# Two-Phase 演算法簡介

- Two-Phase 共分為兩個階段(Phases)
  - **Phase I**：利用 Apriori 架構，從資料庫中找出所有 High TWU Itemset。
  - **Phase II**：掃描原始交易資料庫，計算每個 High TWU Itemset 真正的 Utility 值，以找出所有 HUI 。

# 舉例說明
# Two-Phase 演算法 (Cont. 1/4)

- **Phase I**：搜尋高TWU項目集，從項目次數1開始。
- *min_util* (最小效益門檻值) = ***32***

| Transactional Table | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

$$twu(X) = \sum_{X \subseteq T_q \in D} tu(T_q)$$

| Transactional Table (1st Scan) | | | |
|---|---|---|---|
| **Item** | **Transaction** | **TWU** | **High TWU** |
| *A* | $TU(T_1) + TU(T_2) + TU(T_3)$ | 46 | ☑ |
| *B* | $TU(T_3) + TU(T_4) + TU(T_5)$ | 45 | ☑ |
| *C* | $TU(T_1) + TU(T_2) + TU(T_3) + TU(T_4) + TU(T_5)$ | 69 | ☑ |
| *D* | $TU(T_1) + TU(T_3) + TU(T_4)$ | 47 | ☑ |

# 舉例說明
# Two-Phase 演算法 (Cont. 2/4)

- **Phase I**：搜尋高TWU項目集，依**1st Scan**的HTWU結果，進行**2nd Scan** ，項目次數2。

| Transactional Table (2nd Scan) | | | |
|---|---|---|---|
| **Item** | **Transaction** | **TWU** | **High TWU** |
| *AB* | $TU(T_3)$ | 22 | |
| *AC* | $TU(T_1) + TU(T_2) + TU(T_3)$ | 46 | ☑ |
| *AD* | $TU(T_1) + TU(T_3)$ | 30 | |
| *BC* | $TU(T_3) + TU(T_4) + TU(T_5)$ | 45 | ☑ |
| *BD* | $TU(T_3) + TU(T_4)$ | 39 | ☑ |
| *CD* | $TU(T_1) + TU(T_3) + TU(T_4)$ | 47 | ☑ |

# 舉例說明
# Two-Phase 演算法 (Cont. 3/4)

- **Phase I**：搜尋 High TWU Itemsets，依**2nd Scan**的 HTWUI 結果，進行**3rd Scan** 。

| Transactional Table (3rd Scan) | | | |
|---|---|---|---|
| **Item** | **Transaction** | **TWU** | **High TWU** |
| *BCD* | $TU(T_3) + TU(T_4)$ | 39 | ☑ |

# 舉例說明
# Two-Phase 演算法 (Cont. 4/4)

- **Phase II**：再進行一次原始資料庫掃瞄，計算出每個 High TWU Itemset 真正的 Utility 值，以找出所有 HUI 。

| Transactional Table (Phase II) | | | | | |
|---|---|---|---|---|---|
| **Item** | **Transaction** | **TWU** | **High TWU** | **Utility** | **HUI** |
| *A* | $TU(T_1) + TU(T_2) + TU(T_3)$ | 46 | ☑ | 20 | |
| *B* | $TU(T_3) + TU(T_4) + TU(T_5)$ | 45 | ☑ | 16 | |
| *C* | $TU(T_1) + TU(T_2) + TU(T_3) + TU(T_4) + TU(T_5)$ | 69 | ☑ | 13 | |
| *D* | $TU(T_1) + TU(T_3) + TU(T_4)$ | 47 | ☑ | 20 | |
| *AC* | $TU(T_1) + TU(T_2) + TU(T_3)$ | 46 | ☑ | 28 | |
| *BC* | $TU(T_3) + TU(T_4) + TU(T_5)$ | 45 | ☑ | 22 | |
| *BD* | $TU(T_3) + TU(T_4)$ | 39 | ☑ | 30 | |
| *CD* | $TU(T_1) + TU(T_3) + TU(T_4)$ | 47 | ☑ | 25 | |
| *BCD* | $TU(T_3) + TU(T_4)$ | 39 | ☑ | 34 | ☑ |

# Two-Phase 演算法主要缺點

- Two-Phase 沿用 Apriori 架構，因此會沿襲 Apriori 的主要缺點
  - 重複掃描原始資料庫 (Heavy I/O Cost)
  - 在 Phase I 產生越多 Candidates / HTWUIs
    - 時間成本越大 (如:產生Candidate, 搜尋, 計算Utility/TWU)
    - 空間成本越大
  - Itemset 的 Utility Upper Bound (i.e., TWU) 估計值過高
    - 產生太多 HTWUI
    - Phase II 的執行時間越久

# 常見的效益項目集探勘演算法

- 常見的效益項目集探勘演算法包括：
  - Two-Phase (PAKDD 2005) [1]
  - IHUP (TKDE 2009) [2]
  - UP-Growth (ACM SIGKDD 2010) [3]
  - UP-Growth$^{+}$ (TKDE 2013) [4]
  - HUI-Miner (CIKM 2012) [5]
  - FHM (ISMIS 2014) [6]

# UP-Growth 演算法之執行效率[3]

Table 11. Number of candidates on Chess

| Minimum utility | IHUP+FPG | UP+FPG | UP+UPG |
|---|---|---|---|
| 30% | 38,686,975 | 4,108,608 | 1,592,414 |
| 40% | 6,682,935 | 79,766 | 3,527 |
| 50% | 1,320,445 | 37 | 37 |
| 60% | 264,418 | 34 | 34 |
| 70% | 49,039 | 24 | 24 |

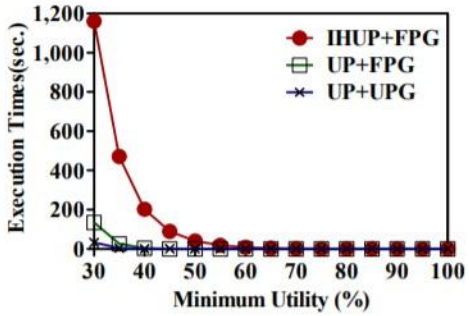



Figure 7. Execution time for phase I on Chess.

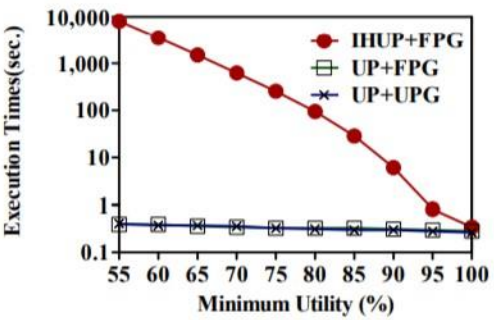



Figure 8. Execution time for phase II on Chess.

Table 12. Number of candidates on BMS-Web-View-1

| Minimum utility | IHUP+FPG | UP+FPG | UP+UPG |
|---|---|---|---|
| 2.9% | 29,561,924 | 206 | 206 |
| 3.0% | 566,651 | 164 | 164 |
| 3.2% | 2,010 | 133 | 133 |
| 3.6% | 387 | 108 | 108 |
| 4.0% | 170 | 76 | 76 |

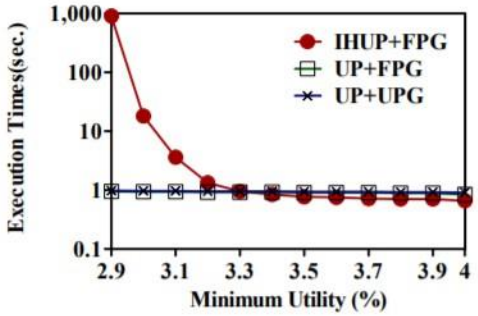



Figure 9. Execution time for phase I on BMS-Web-View-1.

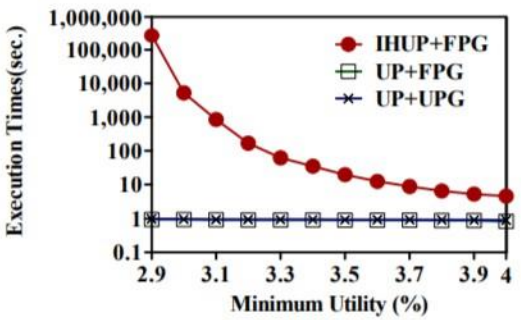
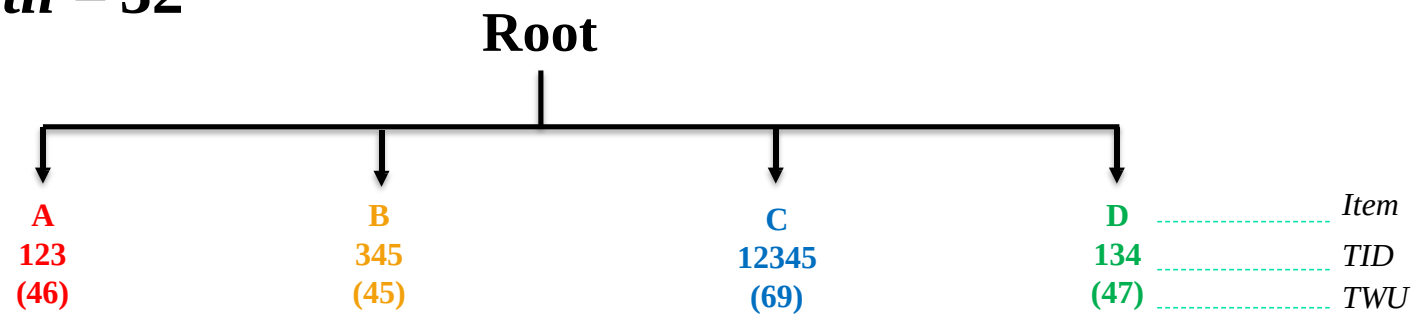



Figure 10. Execution time for phase II on BMS-Web-View-1.

# Two-Phase Algorithm using Vertical Data Representation

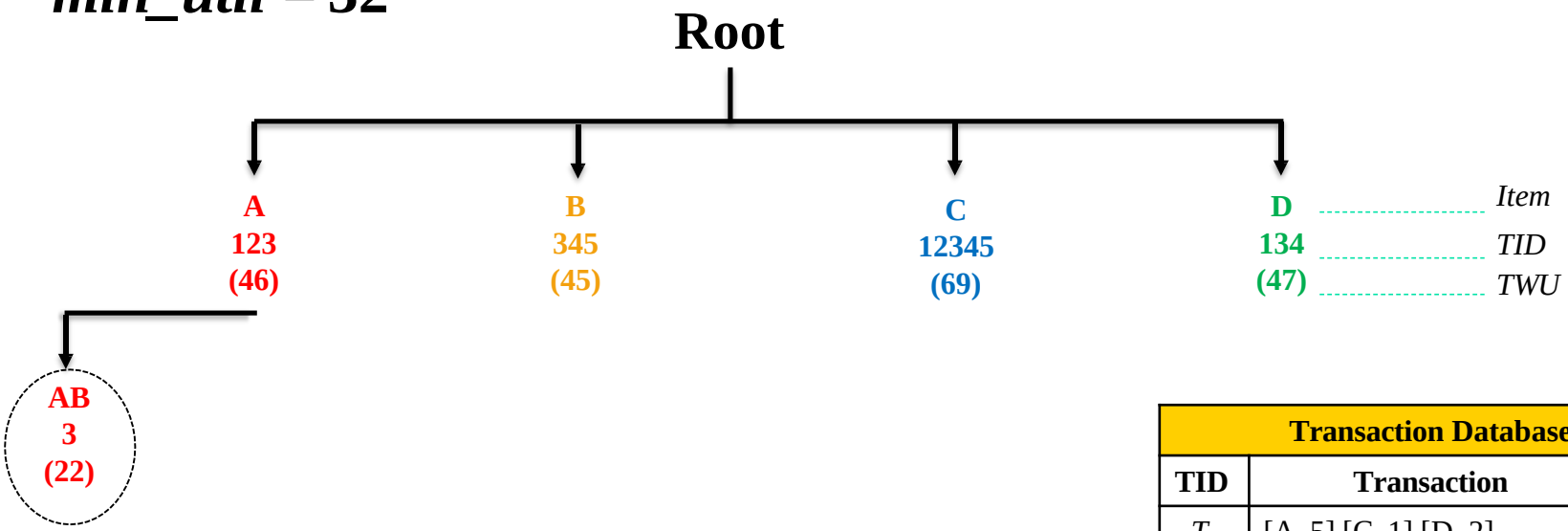
***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Two-Phase Algorithm using Vertical Data Representation

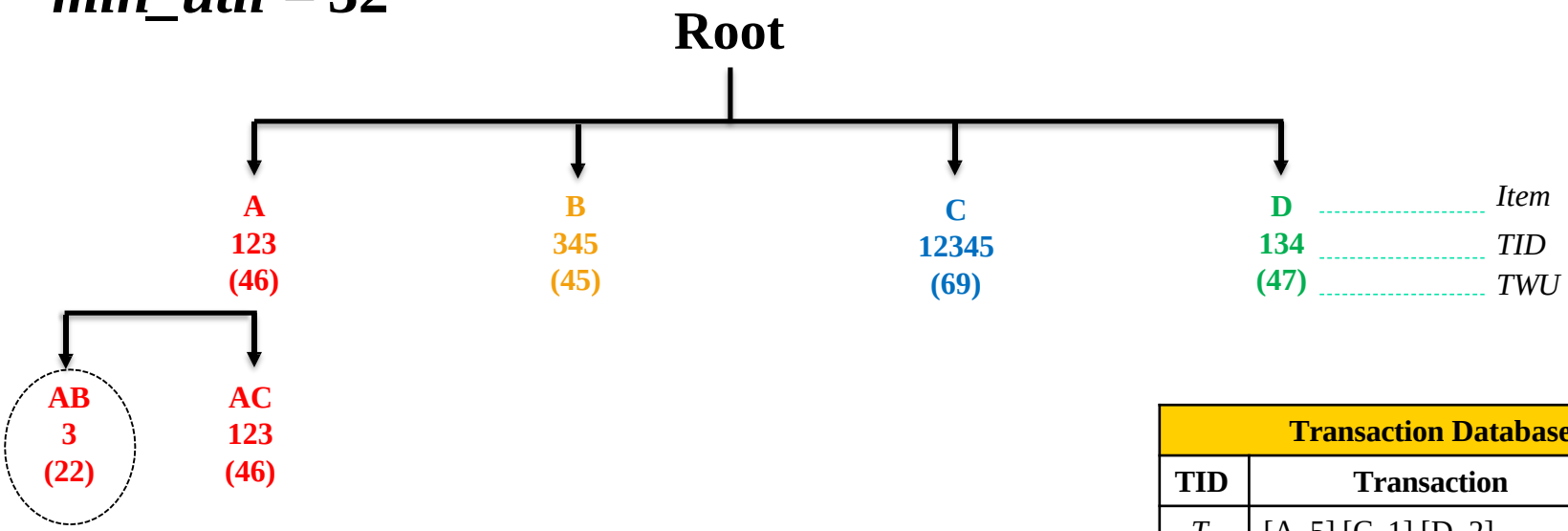
***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Two-Phase Algorithm using Vertical Data Representation

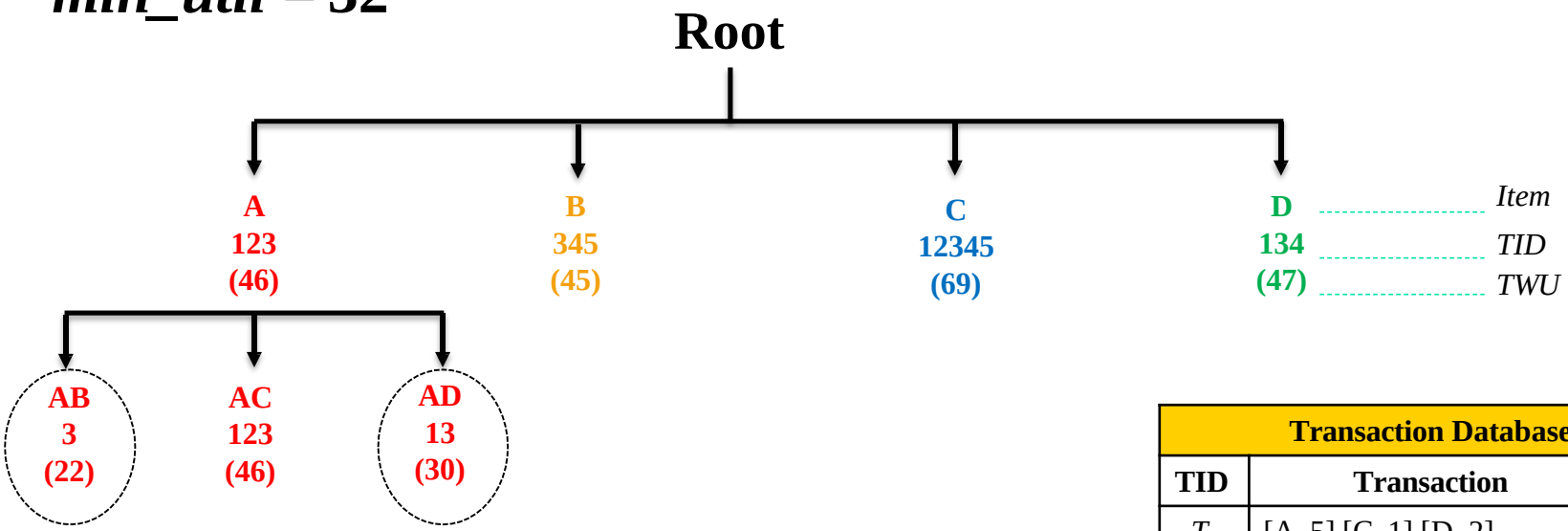
***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Two-Phase Algorithm using Vertical Data Representation

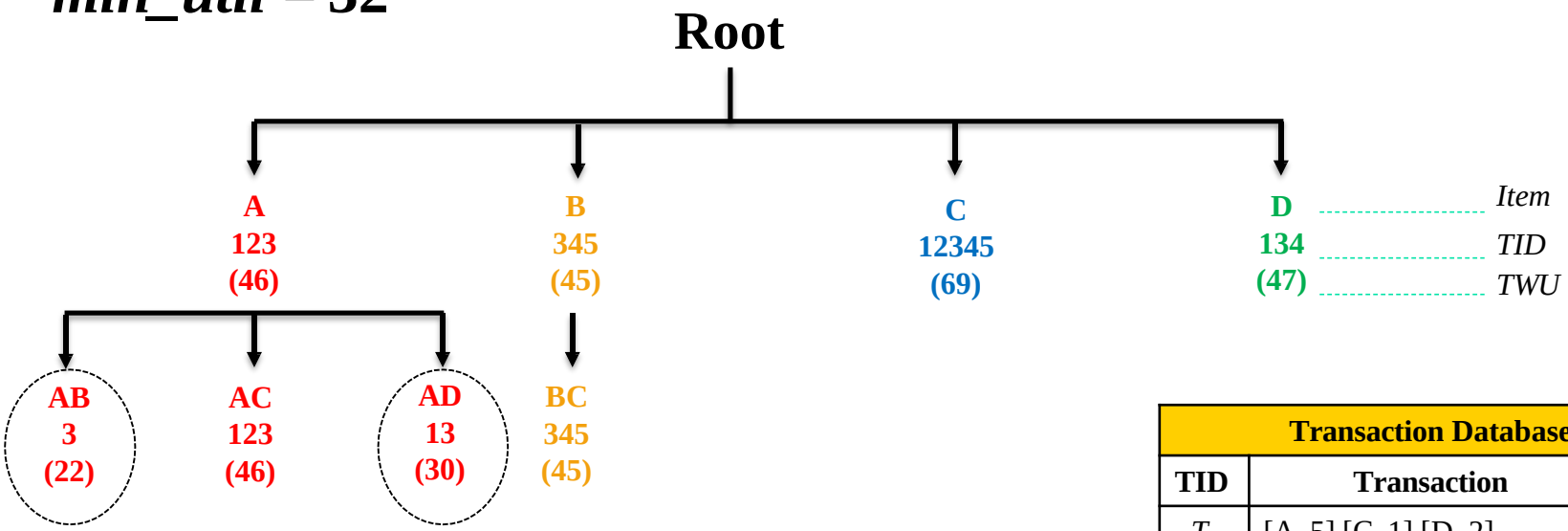
***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Two-Phase Algorithm using Vertical Data Representation

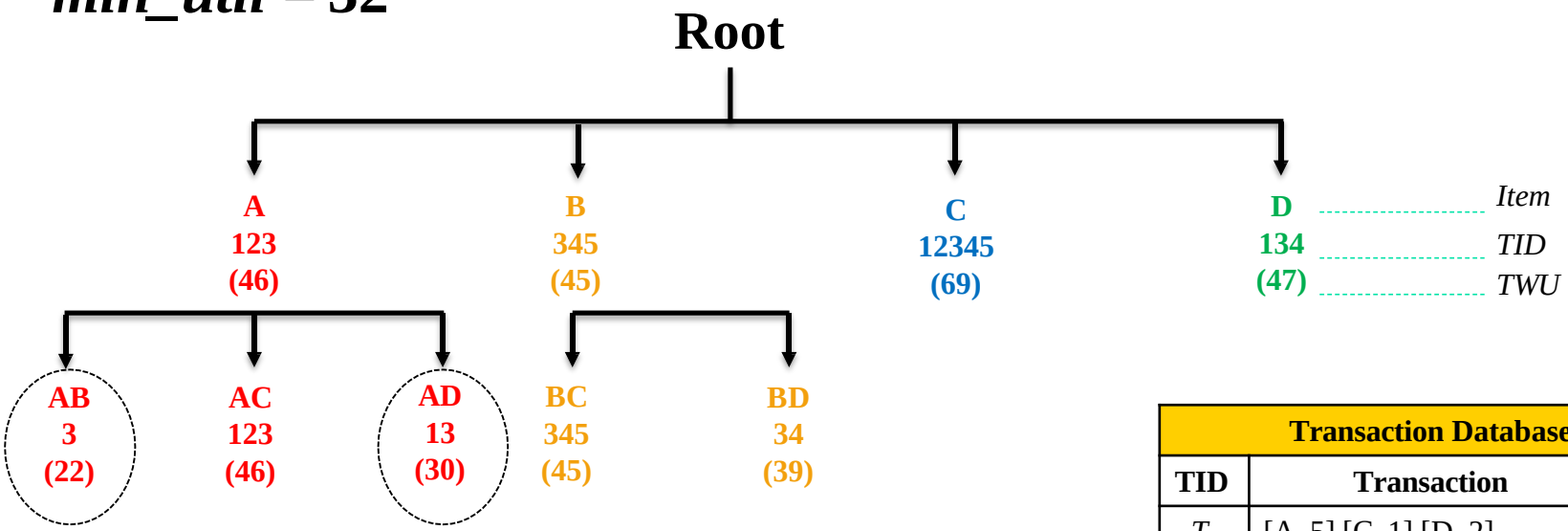
***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Two-Phase Algorithm using Vertical Data Representation

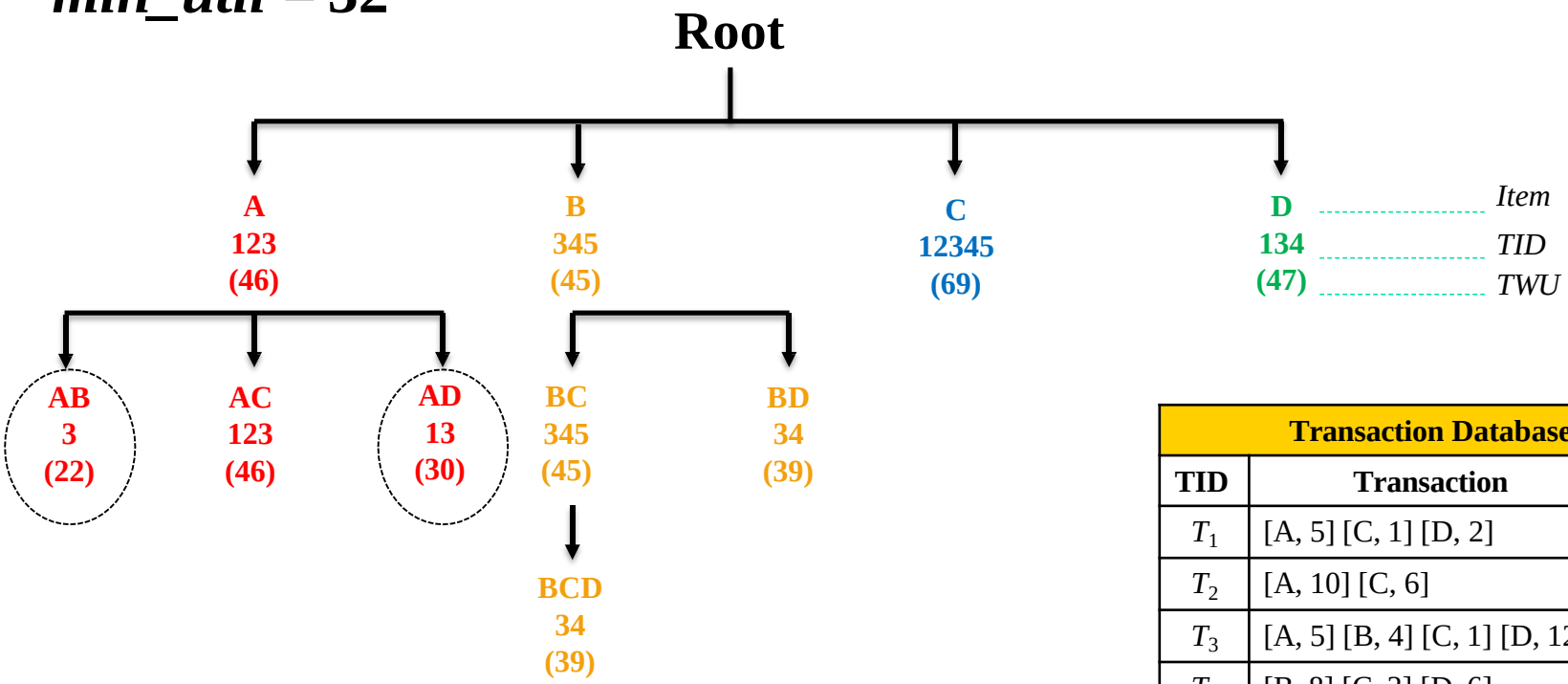
***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Two-Phase Algorithm using Vertical Data Representation

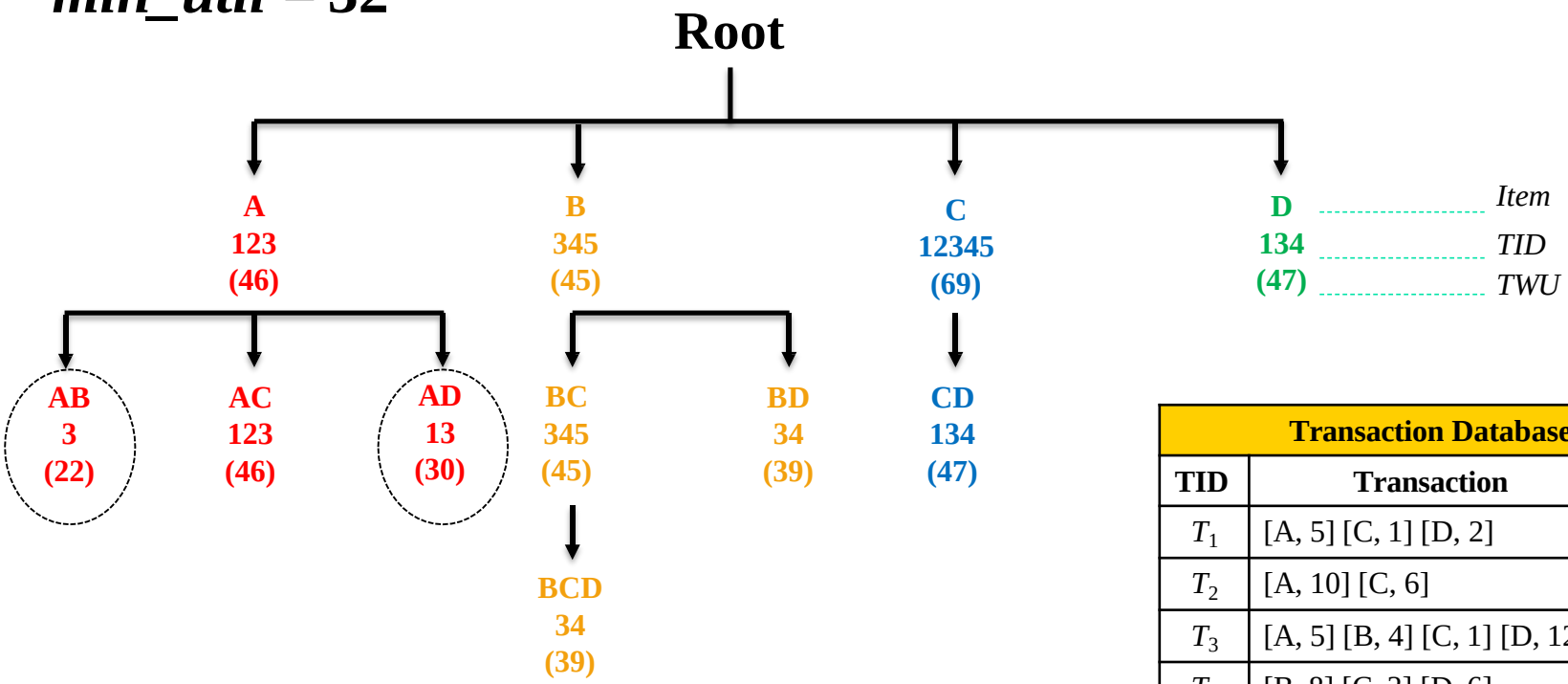
***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# Two-Phase Algorithm using Vertical Data Representation

***min_util* = 32**




| Transaction Database | | |
|---|---|---|
| **TID** | **Transaction** | **TU** |
| $T_1$ | [A, 5] [C, 1] [D, 2] | 8 |
| $T_2$ | [A, 10] [C, 6] | 16 |
| $T_3$ | [A, 5] [B, 4] [C, 1] [D, 12] | 22 |
| $T_4$ | [B, 8] [C, 3] [D, 6] | 17 |
| $T_5$ | [B, 4] [C, 2] | 6 |

# 結論

- 傳統頻繁項目集探勘缺點
- 效益型樣探勘目的
- 效益型樣探勘技術
- 效益型樣應用
- 效益型樣探勘軟體操作

# Reference


- [1] Ying Liu, Weikeng Liao, A. Choudhary. (2005). “ A two-phase algorithm for fast discovery of high utility itemsets “. PAKDD'05: Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining, pp. 689–695.
- [2] C. F. Ahmed, S. K. Tanbeer, B. Jeong and Y. Lee. (2009). “Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases,” IEEE Transactions on Knowledge and Data Engineering, 21(12):1708-1721.
- [3] V. S. Tseng, C. Wu, B. Shie and P. S. Yu. (2010). “UP-Growth: An Efficient Algorithm for High Utility Itemset Mining,” in Proc. of ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 253–262.
- [4] V. S. Tseng, B. Shie, C. Wu and P. S. Yu. (2013). “Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases,” IEEE Transactions on Knowledge and Data Engineering, 25(8):1772-1786.
- [5] Mengchi Liu, Junfeng Qu, (2012). “Mining high utility itemsets without candidate generation” CIKM '12: Proceedings of the 21st ACM international conference on Information and knowledge management, Pages 55–64.
- [6] P. Fournier-Viger, C. Wu, S. Zida and V. S. Tseng, (2014). “FHM: Faster High-Utility Itemset Mining Using Estimated Utility Co-occurrence Pruning,” in Proc. of Int’l Symposium on Methodologies for Intelligent Systems, pp. 83-92.

# 教師資訊

- 姓名：吳政瑋 (小吳老師)
- 現職：宜大資工助理教授
- 學歷：成功大學資工博士
- 研究興趣：資料探勘、人工智慧、AIoT應用
- 通訊方式
  - 電子信箱：wucw@niu.edu.tw
  - 校內電話: (03)9317331
  - Line: silvemoonfox
  - Office: 格致大樓E405室
  - 數位學習園區
- 實驗室：ＡＩ與資料科學實驗室
  - https://sites.google.com/view/cwwuadslab/

# 意見交流

歡迎提供意見與指導!!

您的寶貴意見將使本系更進步!!