COMPSCI 361 - Tutorial 5

Shuxiang Zhang

Semester 1, Week 5, 2020

Adapted from 2019 slides by Jordan Douglas

Introduction

- An unsupervised learning task.
- **Input:** a set of items I (binary attributes) and a set of transactions D (instances).
- **Output:** a set of rules A$\rightarrow$ B which state that when A is true, B is likely to be true, where A, B $\subseteq$ I.
- Each transaction in D contains a subset of the items in I.
- These rules are associations/correlations only and do not imply a causal link.

Running example 1: steam games

ID	The_Elder_Scrolls_V_Skyrim	Fallout_4	Spore	Fallout_New_Vegas	Left_4_Dead_2	Path_of_Exile	Poly_Bridge	Left_4_Dead	Team
151603712	1	1	1	1	1	1	1	1	
187131847	0	0	0	0	0	0	0	0	
59945701	1	0	0	1	1	0	0	0	
53875128	1	0	1	1	1	0	0	1	
234941318	0	0	0	0	0	0	0	0	
140954425	0	0	0	0	0	0	0	0	
26122540	0	0	0	0	0	0	0	0	
176410694	0	0	0	0	0	0	0	0	
197278511	0	0	0	0	0	0	0	0	
150128162	0	0	0	0	0	0	0	0	
197455089	0	0	0	0	0	0	0	0	
63024728	0	0	0	0	0	0	0	0	
297811211	0	0	0	0	0	0	0	0	
76933274	0	0	0	0	0	0	0	0	

- Each row is a transaction (ie. a steam user account).
- Each column is a is a binary attribute describing if the user has the product in their library.
- Processed data is on Canvas, original data available here.

Running example 2: Game of Thrones episodes

	A	B	C	D	E	F	G	H	I	J	K	L	
1	Episode	Tyrion_Lannister	Jon_Snow	Daenerys_Targaryen	Cersei_Lannister	Sansa_Stark	Arya_Stark	Jaime_Lannister	Jorah_Mormont	Samwell_Tarly	Theon_Greyjoy	Lord_Varys	Dav
2	1	1	1	1	1	1	1	1	1	0	1	0	
3	2	1	1	1	1	1	1	1	1	0	1	0	
4	3	1	1	1	1	1	1	1	1	0	0	1	
5	4	1	1	1	1	1	1	1	1	1	1	1	
6	5	1	0	0	1	1	1	1	0	0	1	1	
7	6	1	0	1	1	1	1	0	1	0	1	0	
8	7	0	1	1	1	0	0	1	1	1	1	1	
9	8	1	1	1	1	1	1	0	1	1	1	1	
10	9	1	1	1	1	1	1	1	1	1	1	1	
11	10	1	1	1	1	1	1	1	1	1	1	1	
12	11	1	1	1	1	1	1	1	1	1	1	1	
13	12	1	1	1	1	0	1	0	1	1	1	1	
14	13	1	1	0	1	1	1	0	0	1	1	1	
15	14	1	0	1	0	1	1	0	1	0	0	0	
16	15	1	1	1	1	0	1	0	1	1	1	0	

- Each row is an episode (73 episodes).
- Each column whether the character is in the episode (only top 100 characters included).
- Processed data on Canvas, original data by Jeffrey Lancaster.

Implications

A	B	A→B	B→A
0	0	1	1
0	1	1	0
1	0	0	1
1	1	1	1

- For A to imply B, it must be the case that if A is true then B is also true.

Implications: example 1

User	CS:GO	DOTA 2	TF2
1	1	0	1
2	0	1	0
3	1	1	1
4	1	0	1
5	0	1	1
6	0	0	0
7	0	1	1
8	0	0	0
9	1	0	1

- What implications do you observe in the above data?

Implications: example 1

- CS:GO→TF2.
- If they play CS:GO, they are likely to play TF2.

Implications: example 2

Episode	Jon_Snow	Tormund_Giantsbane	Catelyn_Stark
21	1	1	1
22	1	1	1
23	1	1	1
24	0	0	1
25	1	1	1
26	1	1	1
27	1	1	1
28	0	0	0
29	1	1	1
30	1	0	0

- What implications do you observe in the above data?

Implications: example 2

- Tormund_Giantsbane→Jon_Snow.
- If Tormund is in the episode, then so is Jon_Snow.

Support and Confidence

- Given a rule A $\rightarrow$ B.
- **Support:** fraction of transactions which contain A and B, i.e. $P(A \wedge B)$.
- **Confidence:** how often items B appear in transactions that contain A, i.e. $P(A|B) = \frac{P(A \wedge B)}{P(A)}$.
- The goal is to find rules which have high support and high confidence.

Support and confidence: example 1

User	CS:GO	DOTA 2	TF2
1	1	0	1
2	0	1	0
3	1	1	1
4	1	0	1
5	0	1	1
6	0	0	0
7	0	1	1
8	0	0	0
9	1	0	1

- CS:GO $\rightarrow$ TF2. What is the support and confidence of this rule?
- DOTA 2 $\rightarrow$ TF2. What is the support and confidence of this rule?

Support and confidence: example 1

- CS:GO $\rightarrow$ TF2.
- **Support:** $P(CS:GO \wedge TF2) = \frac{4}{9}$.
- **Confidence:** $P(TF2|CS:GO) = \frac{4}{4} = 1$.

- DOTA2 $\rightarrow$ TF2.
- **Support:** $P(DOTA2 \wedge TF2) = \frac{3}{9}$.
- **Confidence:** $P(TF2|DOTA2) = \frac{3}{4} = 0.75$.

- The first rule has more support and more confidence.

Support and confidence: example 1

Game(s)	Confidence
CoD: Modern Warfare 2 → CoD: Modern Warfare 2 - Multiplayer	0.96
CS: Condition Zero → CS	0.94
Garry's Mod and DOTA 2 → TF 2	0.94
Garry's Mod and Left 4 Dead 2 → TF 2	0.94
Terraria and Left 4 Dead 2 → TF 2	0.93
Spiral Knights → TF 2	0.92
DOTA 2 and Terraria → TF 2	0.92
Left 4 Dead 2 and Half-Life 2 → TF 2	0.92
DOTA 2 and Portal 2 → TF 2	0.92
Left 4 Dead 2 and Portal → TF 2	0.92
Garry's Mod and CS: Source → TF 2	0.92
Alien Swarm and Left 4 Dead 2 → TF 2	0.92
Portal 2 and Left 4 Dead 2 → TF 2	0.91
Garry's Mod → TF 2	0.91
CoD: Modern Warfare 2 - Multiplayer → CoD: Modern Warfare 2	0.90

Table 3: Top 15 association rules, note TF 2's prevalence.

- Sifa, Rafet, Anders Drachen, and Christian Bauckhage. "Large-scale cross-game player behavior analysis on steam." Borderlands 2 (2015): 46-378.

Itemsets

- An itemset is a set of one or more items ie. attributes.
- If there are d items there are 2^d possible itemsets.
- Eg. for d $=$ 3 the itemsets are $\{\{\}, \{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}\}$.
- A frequent itemset is an itemset whose support is greater than some threshold.
- An itemset if maximal frequent if none of its immediate supersets are frequent.
- As the number of items in a set increases, the support is non-increasing ie. $P(\{0, 1, 2\}) \leqslant P(\{0, 1\})$

Itemsets

- The task is to find itemsets which have:
  - Support greater than threshold ϵ_S.
  - Confidence greater than threshold ϵ_C.
- But without exhaustively examining all possible 2^d itemsets.
- We will cover two algorithms to achieve this goal.

Introduction

- An algorithm to find frequent itemsets (with support threshold ϵ_S) eg. $\{\{A, B\}, \{B, C, D\}\}$.
- A breadth-first search algorithm that prunes the tree when support drops too low.
- Runtime of (2^d) where d is the total number of items.
- The rules which have high confidence (greater than ϵ_C) can be extracted from these item sets eg. $\{\{C \wedge D \rightarrow B\}\}$.

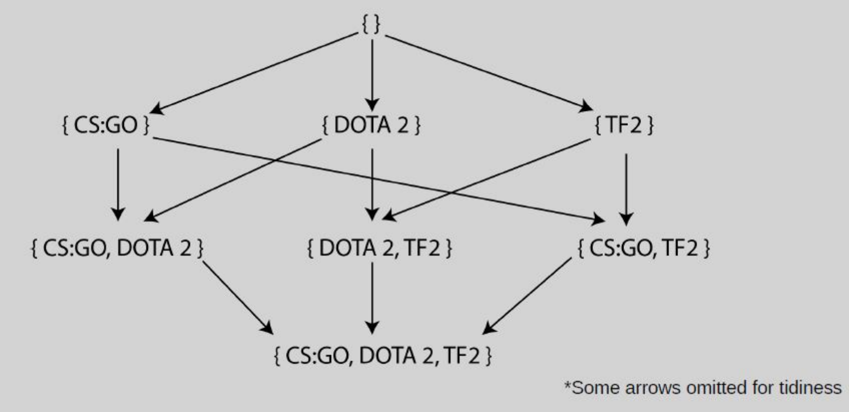
apriori algorithm: example 1

User	CS:GO	DOTA 2	TF2
1	1	0	1
2	0	1	0
3	1	1	1
4	1	0	1
5	0	1	1
6	0	0	0
7	0	1	1
8	0	0	0
9	1	0	1

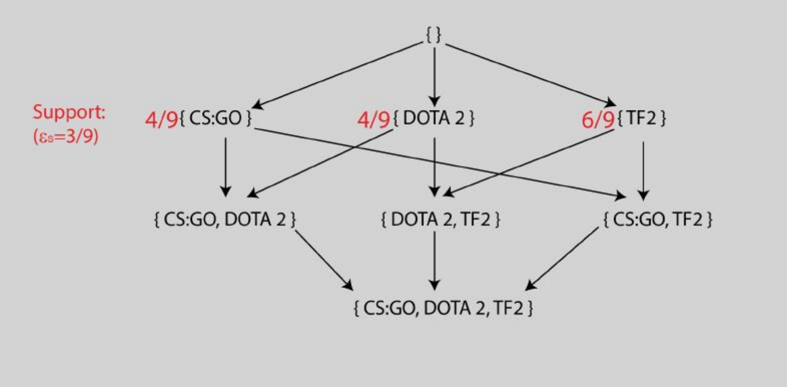
apriori algorithm: example 1

- Apply the apriori algorithm to the Steam user data to find all rules which have:
  - Support $\geqslant \epsilon_S = \frac{1}{3}$.
  - Confidence $\geqslant \epsilon_C = 0.8$.

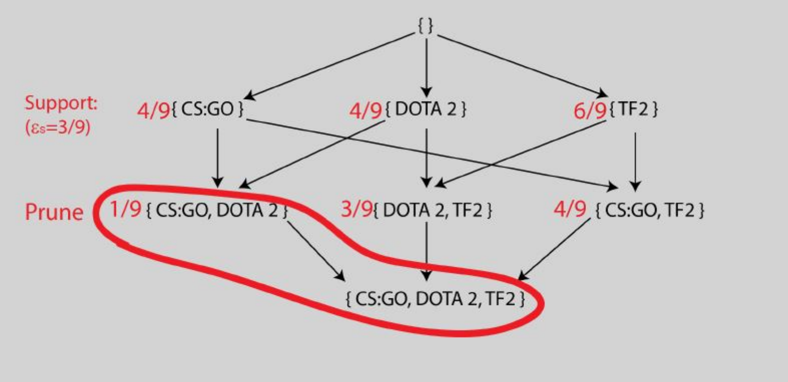
apriori algorithm: example 1



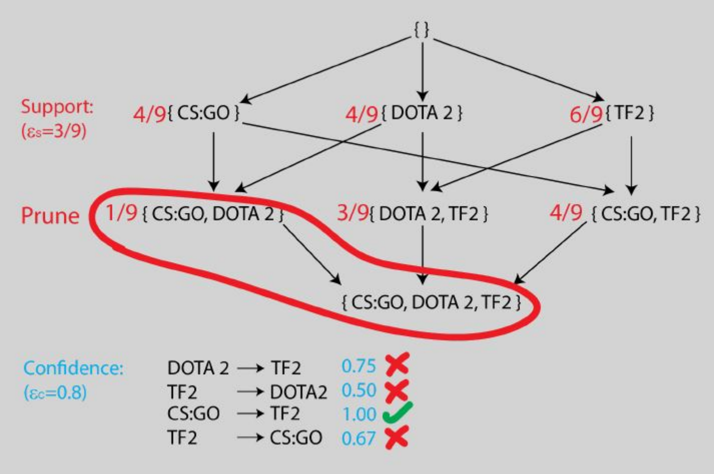
apriori algorithm: example 1



apriori algorithm: example 1



apriori algorithm: example 1



apriori algorithm: example 1

- The following itemsets are frequent: $\{\{CS:GO\},\{DOTA2\},\{TF2\},\{DOTA2,TF2\},\{CS:GO,TF2\}\}$.
- The following itemsets are maximal frequent: $\{\{DOTA2,TF2\},\{CS:GO,TF2\}\}$.
- The following rules derived from frequent itemsets have high confidence: $\{\{CS:GO \rightarrow TF2\}\}$.

apriori algorithm: example 2

- Apply the apriori algorithm to the Game of Thrones episode data to find all rules which have:
  - Support $\geqslant \epsilon_S = \frac{7}{10}$.
  - Confidence $\geqslant \epsilon_C = \frac{10}{10}$.

apriori algorithm: example 2

Episode	Jon_Snow	Tormund_Giantsbane	Catelyn_Stark	Daenerys_Targaryen
21	1	1	1	1
22	1	1	1	1
23	1	1	1	0
24	0	0	1	1
25	1	1	1	1
26	1	1	1	1
27	1	1	1	0
28	0	0	0	1
29	1	1	1	1
30	1	0	0	1

apriori algorithm: example 2

- The following itemsets are frequent: $\{\{J\}, \{T\}, \{C\}, \{D\}, \{J, T\}, \{J, C\}, \{T, C\}, \{J, T, C\}\}$.
- The following itemset is maximal frequent: $\{\{J, T, C\}\}$.
- The following rules derived from frequent itemsets have high confidence: $\{\{T \to J\}, \{T \to C\}, \{T \to J \wedge C\}\}$.

apriori algorithm: example 2

- Load data into Weka (eg. steam.csv or episodes.csv from Canvas). Ensure every attribute is nominal and remove ID.
- Go to the associate tab.
- Select the apriori algorithm and open its settings.
- "lowerBoundMinSupport" is $\epsilon_S = 0.01$, and "minMetric" is $\epsilon_C = 0.7$.
- To prevent association with "false" values set "treatZeroAsMissing" to true.
- Press "Start".

Apriori Algorithm in Weka

Associator output

```
              Left_4_Dead
              Team_Fortress_2
              Tomb_Raider
              The_Banner_Saga
              Dead_Island_Epidemic
              BioShock_Infinite
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.01 (100 instances)
Minimum metric <confidence>: 0.7
Number of cycles performed: 20

Generated sets of large itemsets:

Size of set of large itemsets L(1): 10

Size of set of large itemsets L(2): 17

Size of set of large itemsets L(3): 5

Best rules found:

 1. Left_4_Dead=1 Team_Fortress_2=1 142 ==> Left_4_Dead_2=1 129    <conf:(0.91)> lift:(11.92) lev:(0.01) [118] conv:(9.37)
 2. Left_4_Dead=1 231 ==> Left_4_Dead_2=1 193    <conf:(0.84)> lift:(10.96) lev:(0.02) [175] conv:(5.47)
 3. Fallout_New_Vegas=1 Left_4_Dead_2=1 132 ==> The_Elder_Scrolls_V_Skyrim=1 104    <conf:(0.79)> lift:(13.54) lev:(0.01) [96] conv:(4.29)
 4. Team_Fortress_2=1 BioShock_Infinite=1 137 ==> Left_4_Dead_2=1 105    <conf:(0.77)> lift:(10.06) lev:(0.01) [94] conv:(3.84)
 5. Left_4_Dead_2=1 BioShock_Infinite=1 143 ==> The_Elder_Scrolls_V_Skyrim=1 107    <conf:(0.75)> lift:(12.86) lev:(0.01) [98] conv:(3.64)
 6. Left_4_Dead_2=1 BioShock_Infinite=1 143 ==> Team_Fortress_2=1 105    <conf:(0.73)> lift:(3.92) lev:(0.01) [78] conv:(2.98)
 7. The_Elder_Scrolls_V_Skyrim=1 Left_4_Dead_2=1 239 ==> Team_Fortress_2=1 169    <conf:(0.71)> lift:(3.78) lev:(0.01) [124] conv:(2.74)
```

Confidence problems

- If itemsets A and B are very common then $A \rightarrow B$ and $B \rightarrow A$ probably has high confidence.
- But this doesn't really mean much. Confidence is not a very informative metric.
- is the confidence of $A \rightarrow B$ divided by the support of B.
- **Lift**$(A \rightarrow B) = \frac{P(B|A)}{P(B)}$.
- **Lift** downweights a rule when B is frequent.

FP Growth

- Step 1: build a compact data structure: the **FP-tree**.
- Step 2: extract frequent itemsets directly from the tree.
- After that, rules with high confidence can be extracted from the frequent itemsets.
- The tree exploits repetitivity in the data.
- (FP = frequent pattern).

FP Growth in Weka

- 1. Load data into Weka (eg. steam.csv or episodes.csv from Canvas). Ensure every attribute is nominal + remove ID attribute.
- 2. Go to the associate tab.
- 3. Select the FPGrowth algorithm and open its settings.
- 4. "lowerBoundMinSupport" is $\epsilon_S = 0.01$, and "minMetric" is $\epsilon_C = 0.7$.
- 5. Press "Start".
- 6. Compare the runtimes between this and Apriori algorithm.

Apriori algorithm vs FP growth

- Both algorithms return the same output: the frequent itemsets, without performing exhaustive search.
- Apriori prunes the graph when support drops below a threshold.
- But apriori still needs to create every generate every candidate itemset (unless pruned).
- FP growth compresses the items into a compact structure.
- The FP tree is usually smaller than the uncompressed data -¿ faster to find frequent itemsets.
- But the FP tree is expensive to build and can consume a lot of memory.

Anaconda vs Jupyter notebook

- **Anaconda** is a free and open source distribution of the Python and R programming languages for scientific computing and data science related applications, which comes with some commonly used Python and R packages and also includes Jupyter notebook.
- **Anaconda** can be downloaded from here.
- **Jupyter notebook** is an interactive IDE for several programming languages such as Python and R, which allows you to create and share documents that contain live code, equations, visualizations and all.
- There are a number of different ways of openning **Jupyter notebook**. A detailed tutorial on this is available here.