1. Association Rules
   - Introduction
   - Definitions

2. Apriori algorithm
   - Finding association rules
   - Algorithm
   - Computational Cost

3. FP-Growth
   - FP-tree

4. Measures of interestingness

5. Twitter - City Connections
1. Association Rules

2. Apriori algorithm

3. FP-Growth

4. Measures of interestingness

5. Twitter - City Connections
A **Binary** database is a database where each row is composed of binary attributes

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

From this database frequent patterns of occurrence can be discovered
Association rules

This kind of databases appear for example on transactional data (market basket analysis)
An association rule represents cooccurrence of events in the database

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Chips, Beer, Yogourt, Eggs</td>
<td>{Flour} → {Eggs}</td>
</tr>
<tr>
<td>2</td>
<td>Flour, Beer, Eggs, Butter,</td>
<td>{Beer} → {Chips}</td>
</tr>
<tr>
<td>3</td>
<td>Bread, Ham, Eggs, Butter, Milk</td>
<td>{Bacon} → {Eggs}</td>
</tr>
<tr>
<td>4</td>
<td>Flour, Eggs, Butter, Chocolate</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Beer, Chips, Bacon, Eggs</td>
<td></td>
</tr>
</tbody>
</table>
Definitions (I)

- We define $R$ as the set of attributes of the database.
- We define $X$ as a subset of attributes from $R$.
- We say that $X$ is a pattern from DB if there is any row where all the attributes of $X$ are 1.
- We define the support (frequency) of a pattern $X$ as the function:

$$fr(X) = \frac{|M(X, r)|}{|r|}$$

- Where $|M(X, r)|$ is the number of times that $X$ appears in the DB and $|r|$ is the size of the BD.
Definitions (II)

- Given a minimum support \((min\_sup \in [0, 1])\), we say that the pattern \(X\) is frequent (frequent itemset) if:

\[
fr(X) \geq min\_sup
\]

- \(\mathcal{F}(r, min\_sup)\) is the set of patterns that are frequent in:

\[
\mathcal{F}(r, min\_sup) = \{X \subseteq R / fr(X) \geq min\_sup\}
\]
Definitions (III)

- Given a BD with $R$ attributes and $X$ and $Y$ attribute subsets, with $X \cap Y = \emptyset$, an **association rule** is the expression:

  $$X \Rightarrow Y$$

- $\text{conf}(X \Rightarrow Y)$ is the **confidence** of an association rule, computed as:

  $$\text{conf}(X \Rightarrow Y) = \frac{\text{fr}(X \cup Y)}{\text{fr}(X)}$$

- We consider a minimum value of confidence ($\text{min}_\text{conf} \in [0, 1]$)
1. Association Rules

2. Apriori algorithm

3. FP-Growth

4. Measures of interestingness

5. Twitter - City Connections
Finding association rules

- Given a minimum confidence and a minimum support, a association rule ($X \Rightarrow Y$) exists in a DB if:

$$\left( fr(X \cup Y) \geq \text{min\_sup} \right) \land \left( \text{conf}(X \Rightarrow Y) \geq \text{min\_conf} \right)$$

- Trivial approach:
  - It is enough to find all frequent subsets from $R$
  - We have to explore $\forall X \forall Y ((X \subseteq R) \land (Y \subseteq X))$ the association rule $(X - Y) \Rightarrow Y$

- There are $2^{|R|}$ candidates
Space of itemsets
Properties of frequent sets

- We need ways to prune the search space
- Given $X$ and $Y$ with $Y \subseteq X$:
  - If $fr(Y) \geq fr(X)$, if $X$ is frequent, $Y$ also is frequent
  - If any subset $Y$ from $X$ is not frequent then $X$ is not frequent
- This is known as the **anti-monotone** property of support
- A feasible exploration approach is
  - Find the frequent sets starting from size 1 and increasing its size
  - Prune the candidates including infrequent itemsets
Pruning of itemsets

The Apriori algorithm is used to find association rules in a dataset. It involves generating candidate itemsets and pruning those that are not frequent. The diagram illustrates the process of pruning itemsets. Each node represents an itemset, and the edges show the superset relationships. The red nodes represent pruned itemsets, and the green nodes represent infrequent itemsets. The pruning process eliminates itemsets that cannot be part of any frequent itemset, optimizing the search for frequent itemsets efficiently.
Finding frequent sets

- \( \mathcal{F}_l(r, \text{min}\_sup) \) is the set of frequent sets from \( R \) of size \( l \).

- Given a set of patterns of length \( l \), the only frequent set candidates of length \( l + 1 \) will be those which all subsets are in the frequent sets of length \( l \).

\[
C(\mathcal{F}_{l+1}(r)) = \{ X \subseteq R / |X| = l + 1 \land \\
\forall Y (Y \subseteq X \land |Y| = l \Rightarrow Y \in \mathcal{F}_l(r)) \}
\]

- The computation of association rules can be done iteratively starting with the smallest frequent subsets until no more candidates are obtained.
**Algorithm**: Apriori \((R, \text{min}\_\text{sup}, \text{min}\_\text{conf})\)

- **Algorithm**
- **Input**: \(C, \text{CCan}, \text{CTemp}: \text{set of frequent subsets} \)
- **Output**: \(\text{RAs}: \text{Set of association rules}, \ L: \text{integer} \)

1. **Initialization**
   - \(L = 1\)
   - \(\text{CCan} = \text{Frequent\_sets}\_1(R, \text{fr\_min})\)
   - \(\text{RAs} = \emptyset\)

2. **Loop**
   - **while** \(\text{CCan} \neq \emptyset\) **do**
     - \(L = L + 1\)
     - \(\text{CTemp} = \text{Candidate\_sets}(L, R, \text{CCan}) = C(\mathcal{F}_{l+1}(r))\)
     - \(C = \text{Frequent\_sets}(\text{CTemp}, \text{min}\_\text{sup})\)
     - \(\text{RAs} = \text{RAs} \cup \text{Confidence\_rules}(C, \text{min}\_\text{conf})\)
     - \(\text{CCan} = C\)

3. **End**
Initial set of itemsets
Itemsets size=1

L=4

L=3

L=2

L=1
Itemsets size = 2
Apriori algorithm

Algorithm

Itemsets size=3

L=4

L=3

L=2

L=1

ABC

ACD

A

C

D

AD

CD

AC
Final frequent itemsets

L=4

L=3

L=2

L=1

A
C
D

AC
AD
CD

ACD
Effect of Support

- The specific value used as $min\_sup$ has effect on the computational cost of the search
  - If it is too high, only a few patterns will appear (we could miss interesting rare occurrences)
  - If it is too low the computational cost will be too high (too many associations will appear)
- Unfortunately the threshold value can not be known beforehand
- Sometimes multiple minimum supports are needed (different for different groups of items)
Other factors affecting computational cost

- Number of different items
  - The more items there are the more space is needed to count its support
- Size of the database
  - The algorithm has to perform multiple passes to count the support
- Average transaction width
  - Affects the maximum length of frequent itemsets
Other pruning strategies

- Some itemsets are redundant because they have identical support as their supersets
- We could focus only on those itemsets, reducing the number of candidates
  - **Maximal frequent itemsets**: A frequent itemset for which none of its immediate supersets are frequent
  - **Closed frequent itemsets**: A frequent itemset for which none of its immediate supersets have the same support, and its support is larger than \textit{minsupport}

Maximal frequent itemsets $\subseteq$ Closed frequent itemsets $\subseteq$ Frequent itemsets
1. Association Rules
2. Apriori algorithm
3. FP-Growth
4. Measures of interestingness
5. Twitter - City Connections
Other approaches

- Despite of pruning and search strategies for Association Rules candidate generation is expensive.
- Other strategies allow to extract association rules using specialized data structures.
FP-Growth

Han, Pei, Yin, Mao *Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach* (2004)

- The problem of the *apriori* approach is that the number of candidates to explore in order to find long patterns can be very large.
- This approach tries to obtain patterns from transaction databases without candidate generation.
- It is based on a specialized data structure (*FP-Tree*) that summarizes the frequency of the patterns in the DB.
- The patterns are explored incrementally adding prefixes without candidate generation.
The goal of this structure is to avoid to query the DB to compute the frequency of patterns.

It assumes that there is an order among the elements of a transaction.

This order allows to obtain common prefixes from the transactions.

The transactions with common prefixes are merged in the structure maintaining the frequency of each subprefix.
FP-Tree - Algorithm

1. Compute the frequency of each individual item in the DB
2. Create the tree root (empty prefix)
3. For each transaction from the DB
   1. Pick the frequent items
   2. Order the items by their original frequency
   3. Iterate for each item, inserting it in the tree
      - If the node has a descendant equal to the actual item increase its frequency
      - Otherwise, a new node is created
## FP-Tree - Example

<table>
<thead>
<tr>
<th>BD</th>
<th>Transactions</th>
<th>Item</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a, e, f</td>
<td>a</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>a, c, b, f, g</td>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>d</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>b, a, e, d</td>
<td>e</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>f</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>d, e, a, b, c</td>
<td>g</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>c, d, f, b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>c, f, a, d</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BD</th>
<th>Transactions (fr=4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a, f</td>
</tr>
<tr>
<td>2</td>
<td>a, c, b, f</td>
</tr>
<tr>
<td>3</td>
<td>a, b, d</td>
</tr>
<tr>
<td>4</td>
<td>a, b, c, d</td>
</tr>
<tr>
<td>5</td>
<td>b, c, d</td>
</tr>
<tr>
<td>6</td>
<td>a, c, d, f</td>
</tr>
</tbody>
</table>
FP-Tree - Example

\[ \lambda \]

\[ \text{Trans}(a,f) \]

\[ \begin{array}{cc}
5 & a \\
4 & b \\
4 & c \\
4 & d \\
4 & f \\
\end{array} \]

\[ a^1 \]

\[ f^1 \]
FP-Tree - Example

Trans(a,b,c,f)

λ

5  a
4  b
4  c
4  d
4  f
FP-Tree - Example

Trans(a,b,d)

\[ \lambda \]

5 \hspace{1cm} a
4 \hspace{1cm} b
4 \hspace{1cm} c
4 \hspace{1cm} d
4 \hspace{1cm} f

\( f^1 \)
\( b^2 \)
\( a^3 \)
\( \lambda \)
\( \text{Trans}(a,b,d) \)
FP-Tree - Example

\[
\begin{array}{c|c}
5 & a^4 \\
4 & b^3 \\
4 & c^2 \\
4 & d^1 \\
4 & f^1 \\
\end{array}
\]

\[\text{Trans}(a,b,cd)\]
FP-Tree - Example

Trans(b,c,d,f)

a^4
b^3
c^2
d^1
f^1

5 a
4 b
4 c
4 d
4 f
FP-Tree - Example

FP-Tree - Example
The FP-Growth Algorithm

1. Given an item, select all the paths that contain the item (*prefix paths*)

2. Convert all the paths that contain the item into a *conditional* FP-tree:
   - Update the counts along the path using the frequency of the selected item
   - Truncate the paths removing the nodes for the item
   - Eliminate from the paths the items that are no longer frequent (if any)

3. For all the items previous in order that are frequent in the conditional FP-tree
   - Count the prefix as frequent
   - Recursively find the frequent items for that prefix
FP-Growth - Example

Extract patterns with suffix \( d \)
FP-Growth - Example

Keep only the paths that contain d (minsupport = 2)

```
<table>
<thead>
<tr>
<th></th>
<th>a:10</th>
<th>b:7</th>
<th>c:4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>b:5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d:2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Javier Béjar (CS - MAI)
Generate the **conditional FP-tree** for \( d \) (updating the counts)
FP-Growth - Example

Prune the path eliminating \( d \) (is no longer needed)
FP-Growth - Example

Solve the problem for the predecesors of \( d \), in this case \( b \) and \( c \) (we are looking if \( bd \) and \( cd \) are frequent)

If we continue the algorithm, the patterns extracted would be \{ [d], [bd], [cd], [abd], [bcd] \}
1. Association Rules
2. Apriori algorithm
3. FP-Growth
4. Measures of interestingness
5. Twitter - City Connections
Given a rule, its interestingness can be computed from a contingency table:

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>(\neg Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>(f_{11})</td>
<td>(f_{10})</td>
</tr>
<tr>
<td>(\neg X)</td>
<td>(f_{01})</td>
<td>(f_{00})</td>
</tr>
<tr>
<td></td>
<td>(f_{+1})</td>
<td>(f_{+0})</td>
</tr>
</tbody>
</table>

- \(f_{11}\); support of \(X\) and \(Y\)
- \(f_{10}\); support of \(X\) and \(\neg Y\)
- \(f_{01}\); support of \(\neg X\) and \(Y\)
- \(f_{01}\); support of \(\neg X\) and \(\neg Y\)
Other measures are based on probabilistic independence and correlation.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift/Interest factor</td>
<td>$N \times \frac{f_{11}}{f_{1+} \times f_{+1}}$</td>
</tr>
<tr>
<td>All Confidence</td>
<td>$\min\left(\frac{f_{11}}{f_{1+}}, \frac{f_{11}}{f_{+1}}\right)$</td>
</tr>
<tr>
<td>Max Confidence</td>
<td>$\max\left(\frac{f_{11}}{f_{1+}}, \frac{f_{11}}{f_{+1}}\right)$</td>
</tr>
<tr>
<td>Kulczynski</td>
<td>$\frac{1}{2}\left(\frac{f_{11}}{f_{1+}} + \frac{f_{11}}{f_{+1}}\right)$</td>
</tr>
<tr>
<td>Cosine Measure</td>
<td>$\frac{f_{11}}{\sqrt{f_{1+}f_{+1}}}$</td>
</tr>
</tbody>
</table>
Properties of a measure of interestingness

- Properties that a measure must hold
  - If $A$ and $B$ are statistically independent then $M(A, B) = 0$
  - $M(A, B)$ increases monotonically with $P(A, B)$ when $P(A)$ and $P(B)$ remain unchanged
  - $M(A, B)$ decreases monotonically with $P(A)$ (or $P(B)$) when $P(A, B)$ and $P(B)$ (or $P(A)$) remain unchanged

- Other properties
  - Symmetry ($M(A, B) = M(B, A)$)
  - Invariant to row/column scaling
  - Invariant to inversions ($f_{11} \leftrightarrow f_{00}, f_{10} \leftrightarrow f_{01}$)
  - Null addition invariant (not affected if $f_{00}$ is increased)
This Python Notebook has examples using the Authors dataset for Apriori and FP-Growth

- Association Rules Notebook ([click here](#) to go to the url)

If you have downloaded the code from the repository you will be able to play with the notebooks ([run jupyter notebook](#) to open the notebooks)
1. Association Rules
2. Apriori algorithm
3. FP-Growth
4. Measures of interestingness

5. Twitter - City Connections
The dataset consists on tweets with a time stamp and longitude and latitude coordinates.

Data has been collected for several months (2.5 million tweets) in the Barcelona area (30 km × 30 km).

The goal is to perform spatio-temporal analysis of the behavior of people in a geographical area.

We are interested in patterns that show connections among different parts of the city.
One approach is to consider a tweet as an event performed by a user.
The events of a user in a single day can be seen as a transaction (market basket).
The attributes of a transaction are the time and the position of the user.
To discover city connections can be solved by discovering frequent itemsets.
Considering all possible values for position and time makes impossible to discover frequent patterns.

In order to obtain a suitable representation for frequent itemsets discovery we need to discretize the attributes.

For time:
- Consider different meaningful hour groups (morning, evening, night, ...)
- Groups of hours

For coordinates:
- Equally spaced grid (granularity?)
- Clustering (what clustering algorithm?)
We pick to discretize time in intervals of 8 hours (0-7, 8-16, 13-23)

Two possibilities for coordinates:
- An equally spaced 300 × 300 grid (100m × 100m)
- Clusters obtained by the leader algorithm (radius 100m)

The first option makes that a transaction has a determined number of attributes (300 × 300 × 3), but no all the possibilities will appear

For second the option the number of attributes will depend on the densities in the coordinates
Generate the database of transactions
Discard the transactions that have only value for 1 attribute
Decide for a value for the support parameters
Use FP-Grow because of the size of the dataset and the number of possible attributes
Only present maximal itemsets to reduce the number or redundant patterns
Twitter - City Connections

Grid Discretization 100m $\sup=50$
Twitter - City Connections - results

Leader Clustering Discretization 100m sup=50