Introduction to *In-Silico* Learning

Data Mining Algorithms

Giacomo Bergami

Use Case

Clash Royale (1/2)

- Clash Royale is a Real Time Strategy game combining collectible card games, tower defense, and multiplayer online battle arena.
- Prior to each battle, players construct a deck of eight cards which they use to attack and defend against their opponent's cards.
- At the start of each game, both players begin with four randomly chosen cards from their deck of eight.

Clash Royale (2/2)

- The API selects all the 1-versus-1 battles.
- From the API, we have no information concerning the randomly selected cards, but just the whole deck of 8.
- For each battle, we know who was the winner.
- We want to implement a recommendation system suggesting the best cards that might bring to an easy victory.

Working Plan

- In order to provide good suggestions, we need to mine only the card decks that are always winning in our dataset.
- From these cards, return the most frequent subset of card patterns.
- Use the most frequent subset of card patterns to generate good association rules.

http://github.com/kekepins/clash-royal-analytics/

Introduction

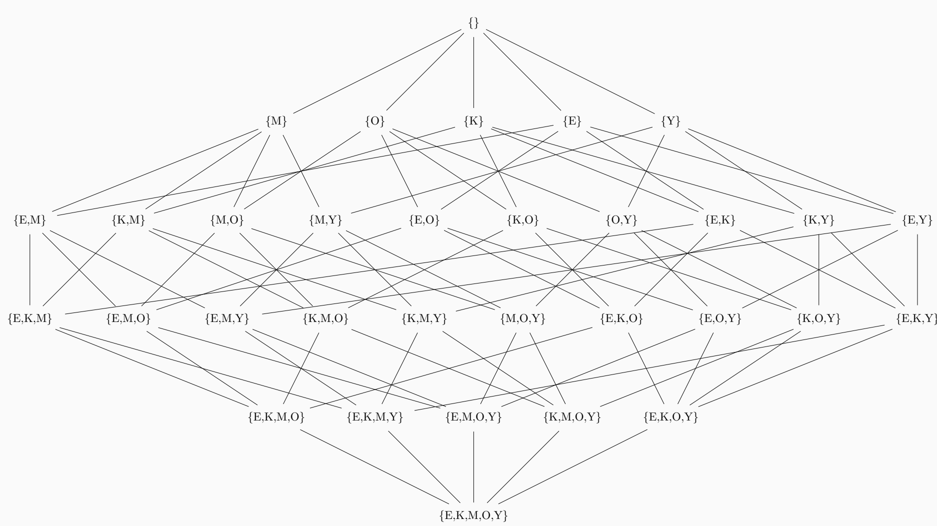
Enumerating Data Mining Algorithm

- Mining and Learning are both search processes
- Therefore, by using a trivial generate-and-test approach we can define the following

Algorithm 1 Returning hypotheses when $\mathcal{L}_h$ is known

```
function ENUMERATE(Q, D, L_h)
    for each h ∈ L_h do
        if Q(h, D) then yield h
        end if
    end for
end function
```


Frequent Itemset's Search Space



Given the set of all the items $\{E, K, M, O, Y\} = \mathcal{L}_e$, the set of all the possible itemsets is $\wp(\{E, K, M, O, Y\}) = \mathcal{L}_h$. We can efficiently traverse it if we use $\supseteq$ as the $\preceq$ relationship to visit the sets.

Why Association Analysis?

This kind of analysis permits to hilight when two events are correlated to each other:

- In market basket analysis, we want to check which are the products that the customer is more likely to buy given the elements that has already bought.
- In earth science, we want to associate patterns of events happening between biosphere, lithosphere, hydrosphere and atmosphere.
- In healthcare, comorbidity could be studied through event correlation.

Mining Association Rules.

Two step approach:

1. Frequent Itemsets Generation
   - Generate all the itemsets whose support is greater than a minsupport threshold.
2. Rule Generation
   - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent Itemset Generation

Definition

- The set of all the possible items (elements, events) is I.
- An observation of a sequence of events is called transacrions. T is the set of all the observed transactions.
  - I can be defined as $I = \bigcup_{t \in T} t$
- The set set of all the possible transactions is $2^I \equiv \wp(I)$, which is a poset $(2^I, \subseteq)$ that could be represented as a lattice. This is the set of all the possible itemsets.
- For each itemset, we could evaluate its support number, that is the number of distinct transactions containing the considered itemset:

$$\sigma(X) = |\,\{\, t \in T \mid X \subseteq t \,\}\,|$$

FP-Growth vs. A Priori

Why do we explain FP-Growth instead the simplest A Priori for the **frequent itemset generation**?

- It uses a divide and conquer strategy for increasing the size of the frequent itemsets
- The FP-Tree efficiently evalutes the **support number**
- It provies a compact representation of all the databases' transactions.
- If the *compaction factor* of the data is high (there are a lot of common subsequences), this algorithm outperforms A Priory by several orders of magnitude.

FP-Growth: *a)* creating the FP-Tree

The FP-Tree maps each transition in the database T as a path in the tree.

- The more the paths overlap, more is the compression we could achieve
- This allows to represent the data structure in main memory without performing multiple I/O (or random accesses) to the database.
- Instead of generating a complete lattice of all the possible frequent solutions, we generate a tree with a support number information.

Example (1/13)

TID	Itemsets
1	$\{M, O, N, K, E, Y\}$
2	$\{D, O, N, K, E, Y\}$
3	$\{M, A, K, E\}$
4	$\{M, U, C, K, Y\}$
5	$\{C, O, O, K, I, E\}$

Database T of 5 itemsets

Example (2/13)

TID	Itemsets
1	$\{M, O, N, K, E, Y\}$
2	$\{D, O, N, K, E, Y\}$
3	$\{M, A, K, E\}$
4	$\{M, U, C, K, Y\}$
5	$\{C, O, O, K, I, E\}$

Item (I)	$\sigma(\cdot)$
M	3
O	3
N	2
K	5
E	4
Y	3
D	1
A	1
U	1
C	2
I	1

Counting the total number of item occurences

Example (3/13)

minsupport=3

TID	Itemsets
1	$\{M, O, N, K, E, Y\}$
2	$\{D, O, N, K, E, Y\}$
3	$\{M, A, K, E\}$
4	$\{M, U, C, K, Y\}$
5	$\{C, O, O, K, I, E\}$

Item (I)	$\sigma(\cdot)$
M	3
O	3
K	5
E	4
Y	3

We want to consider only itemsets that occur at least 3 times

Example (4/13)

TID	Itemsets
1	$\{M, O, N, K, E, Y\}$
2	$\{D, O, N, K, E, Y\}$
3	$\{M, A, K, E\}$
4	$\{M, U, C, K, Y\}$
5	$\{C, O, O, K, I, E\}$

Item (I)	$\sigma(\cdot)$
K	5
E	4
M	3
O	3
Y	3

Sorting the I table by descending support order

- With the whole set of $I = 11$, the total number of all the possible itemsets is $2^{11} = 2048$ (impossible to plot!)
- After pruning the initial space search to $I = 5$, we reduce the visit to $2^5 = 32$.

Example (5/13)

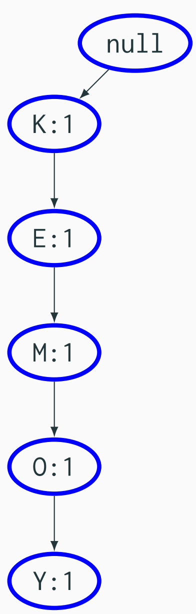
Item (I)	$\sigma(\cdot)$
K	5
E	4
M	3
O	3
Y	3

TID	Itemsets
1	$\{K, E, M, O, Y\}$
2	$\{K, E, O, Y\}$
3	$\{K, E, M\}$
4	$\{K, M, Y\}$
5	$\{K, E, O\}$

Rewriting the itemset using the same order from I

Example (6/13)

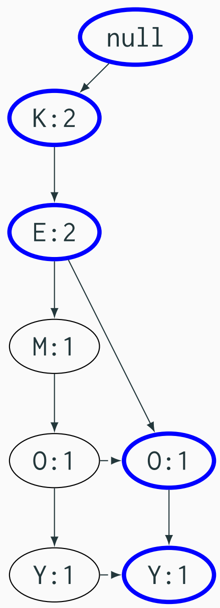
{K,E,M,O,Y}



- Starting from the `null` root, we start creating a path following the order given by I
- Each item in the I table will contain a pointer to the first occurence of the item in the first path.
- The nodes in blue will represent the newly inserted or updated ones.

Example (7/13)

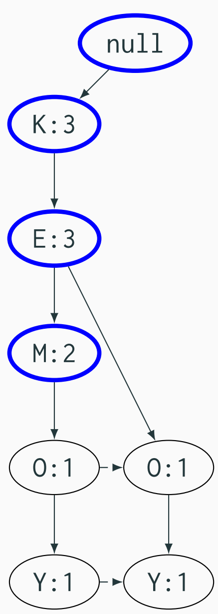
{K,E,O,Y}



- Increment the already visited nodes.
- Create a new branch if there does not exist a path immediately providing the next itemset
- Point the last path occurrence of an item to the next one for traversing ease (dashed lines).

Example (8/13)

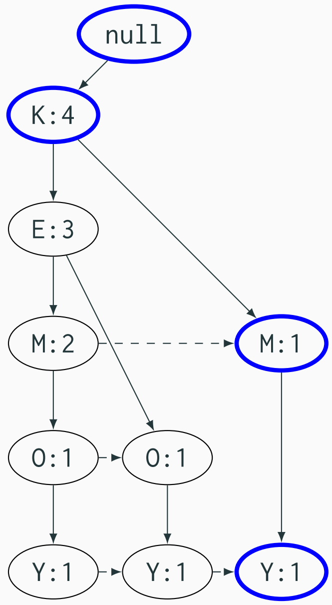
{K,E,M}



- At this stage, we're just incrementing the elements, given that there already exists a path containing the elements of choice.

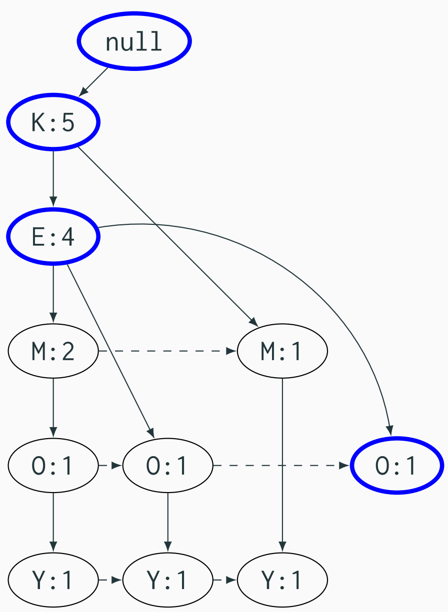
Example (9/13)

{K,M,Y}



Example (10/13)

{K,E,O}



FP-Growth: *b)* Frequent Itemset Generation

After completing the generation of the tree, we want to use such datastructure to minimize the DB access for generating the frequent itemsets.

- Given that the longest set contains all the elements, the worst case scenario for creating such trie is $O(|I|^2)$

FP-Growth: *b)* Frequent Itemset Generation

Run the algorithm FP-FreqIt(x, τ, R, I) with the current item x and R initialized as empty over the current FP-Tree τ.

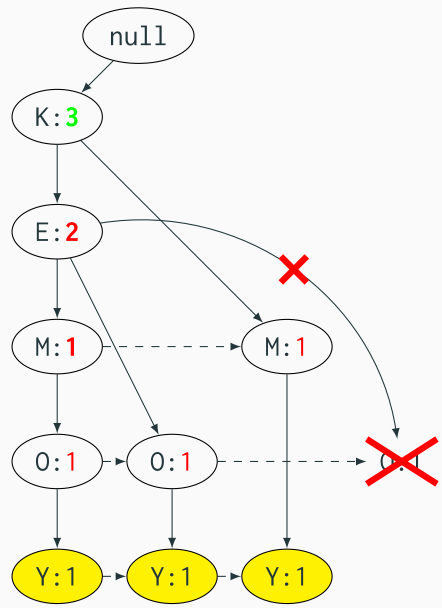
FP-FreqIt(x, τ, R, I'):

1. Insert x in the result set R, R *passed by reference*.
2. Create a PrefixSubtree$_x$ containing only the subpaths in T from the root to the nodes labelled with the last character c in x, so that all the leaves are labelled c.
3. Restructure the pruned tree as follows:
   3.1 For each node, replace its support count as the sum of the supports of all the descendant leaves.
   3.2 Update $\sigma(x)$ in the support table by summing the support of the x-labelled nodes in the subtree.
   3.3 Remove from σ and I' all the nodes that are x or do not pass the minimum support threshold in σ (*passed by copy*).
   3.4 Run FP-FreqIt$(xy, \text{PrefixSubtree}_x, R, I')$ for each node $y \in I'$.

Example (11/13)

FP-FreqIt$(Y, \tau, \{Y\}, I)$

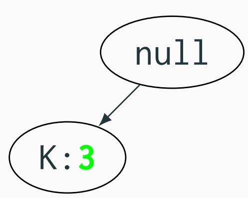
Item (I)	$\sigma(\cdot)$
K	3
E	2
M	2
O	2
Y	3



Example (12/13)

FP-FreqIt$(YK, \text{PrefixSubtree}_Y, \{Y, YK\}, I')$

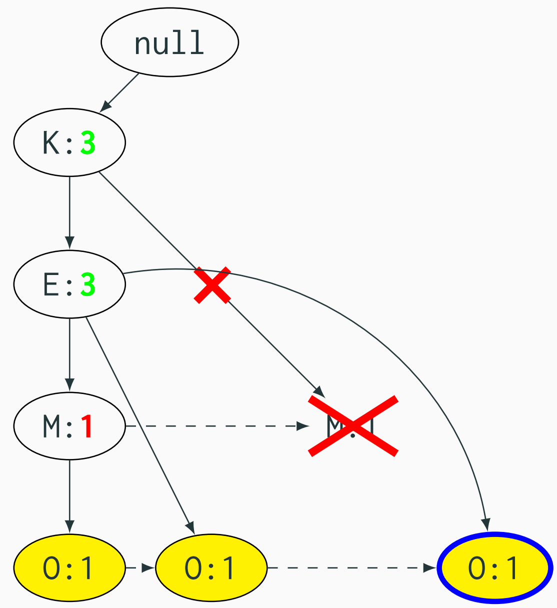
Item (I')	$\sigma(\cdot)$
K	3



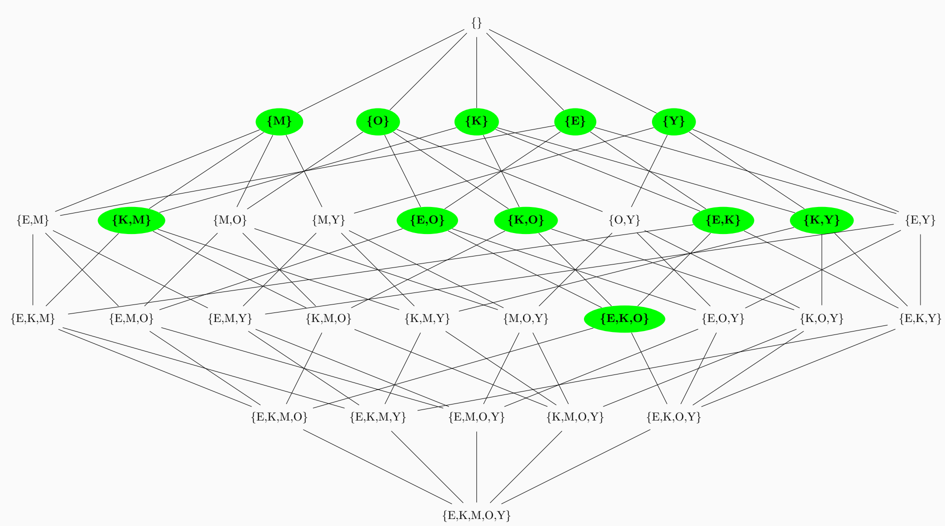
Example (13/13)

FP-FreqIt$(O, \tau, \{Y, YK, O\}, I)$

Item (I)	$\sigma(\cdot)$
K	3
E	3
M	1
O	3



Visiting $\mathcal{L}_h$ (1/2)



- In the worst case scenario, we will compute a `PrefixSubtree` for each node in the lattice.
- Given that min-support is anti-monotonic, we can prune specializations!

Visiting $\mathcal{L}_h$ (2/2)

Let us now analyse some advantages of the FP-Tree:

- It allows a Divide and Conquer strategy to parallelize the algorithm (generation of different `PrefixTrees`).
- If we need to expand the nodes without any additional information on the lattice structure, we will be forced to visit the same element multiple times.
- By duplicating the σ information into the tree, we can then update it while pruning `PrefixTree` without scanning the whole T.

Association Rule Mining

Definition

- An association rule $X \to Y$ shall be interpreted as "if X appears, then it is also likely that Y appears, too". Moreover, $X \cap Y = \emptyset$.
- Those rules are generated from the 2^I elements. This implies that for any $Z \in 2^I$ we can generate a set of association rules $X \to Y$ where $Z = X \cup Y$.

Metrics

We could evaluate any rule $X \to Y$ mainly through two measures:

- support: how many times it could be applied in the dataset (*high support: should apply to a large amount of cases*):

$$s(X \to Y) = \frac{\sigma(X \cup Y)}{|T|}$$

- confidence: the frequency of Y when also X appears (*high confidence: should be often correct*):

$$c(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

- lift: compares the rule confidence with the null hypothesis's confidence (*high lift: indicates the rule is not just a coincidence*):

$$\ell(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)\sigma(Y)}$$

Drawback of Confidence

Association Rule: Tea $\rightarrow$ Coffee

	Coffee	¬Coffee	
Tea	15	5	=20
¬Tea	75	5	=80
	=90	=10	

- $c(\text{Tea} \rightarrow \text{Coffee}) = \frac{15}{20} = 0.75$ but $\mathbb{P}(\mathit{Coffe}) = 0.9$
- Albeit in this case confidence is hight, if trivially interpreted might be misleading!
- $c(\neg\text{Tea} \rightarrow \text{Coffee}) = 0.9375$

Statistical Independence

Given a population of 1000 students, where:

- 600 know how to swim (S)
- 700 know how to bike (B)
- 420 know how to do both (S,B)

$\mathbb{P}(SB) = \frac{420}{1000} = 0.42 \qquad \mathbb{P}(S)\mathbb{P}(B) = \frac{600}{1000}\frac{700}{1000} = 0.42$

- $\mathbb{P}(SB) = \mathbb{P}(S)\mathbb{P}(B)$: statistical independence
- $\mathbb{P}(SB) > \mathbb{P}(S)\mathbb{P}(B)$: positively correlated
- $\mathbb{P}(SB) < \mathbb{P}(S)\mathbb{P}(B)$: negatively correlated

Lift

Association Rule: Tea $\rightarrow$ Coffee

As we will now see, the lift will tell us that this rule is negatively determined, and therefore it is not relevant for our use-case.

	Coffee	$\neg$Coffee	
Tea	15	5	=20
$\neg$Tea	75	5	=80
	=90	=10	

- $c(\text{Tea} \rightarrow \text{Coffee}) = \frac{15}{20} = 0.75$ but $\mathbb{P}(\textit{Coffe}) = 0.9$
- $\textit{lift}(\text{Tea} \rightarrow \text{Coffee}) = \frac{\mathbb{P}(\text{Coffee}|\text{Tea})}{\mathbb{P}(\text{Coffee})} = 0.8\overline{3}$
- $\textit{lift}(\neg\text{Tea} \rightarrow \text{Coffee}) = \frac{\mathbb{P}(\text{Coffee}|\neg\text{Tea})}{\mathbb{P}(\text{Coffee})} = \approx 1.041\overline{6}$

Pattern Evaluation

- Association rule algorithms tend to produce too many rules:
  - many of them are uninteresting or redundant
  - Redundant if $\{A, B, C\} \rightarrow \{D\}$ and $\{A, B\} \rightarrow \{D\}$ have the same support and confidence.
- Interestingness measures can be used to prune and rank the derived rules
- In the original formulation of association rules, support and confidence are the only measures that are used.

Heuristic General-to-Specific Algorithm

- FP-Tree growth uses a general-to-specific algorithm using min support as an anti-monotinic quality criterion.
- There exist many interesting mining and learning tasks for which the quality criterion is neither monotonic nor anti-monotonic.
  - In these cases, it is too inefficient to perform a complete search space:
  - We need to use a branch-and-bound technique.

Heuristic Algorithm: Implementation

Algorithm 2 Branch-and-Bound hypotheses search

function $(\mathcal{Q}, D, \mathcal{L}_h)$
    Queue:=$\{\top\}$ *Th*:=$\emptyset$
    while *Queue*$\neq \emptyset$ **do**
        h:=pop(*Queue*)
        if $\mathcal{Q}(h, D)$ **then**
            Th:=*Th*$\cup\{h\}$
        else
            Queue:=*Queue*$\cup \{ d \in \mathcal{L}_h \mid d \preceq h \}$
        end if
        prune(*Queue*)
    end while
    return *Th*
end function

Heuristic Algorithm for Association Rule Mining

- Run the frequent itemset algorithm, from which we will obtain all the possible σ measures that we are going to use for lift.
- For each frequent itemset X, generate a lattice having a top element $\top = X \to \emptyset$.
- Specialize each hypothesis h by moving the elements on the premises to the consequences one at a time.
- Use both as a `prune` and as a quality strategy $\mathcal{Q}(h, D)$ the *min lift* criterion: *lift*$(h) > 1$.

Further Work

- Given that the *lift* is monotonic, implement a specific to general algorithm working as follows:
    1. Add to the queue all the leaves $X \to Y$ where $|X| = 1$ and $X \cup Y$ is a frequent itemset.
    2. If $X \to Y$ satisfies the minimum lift of 1, then add the current hypothesis h to the result set.
    3. Otherwise, for all the rules $Z \to T$ s.t. $\neg(X \to Y \preceq Z \to T)$ add $X \cap Z \to Y \cup T$ to the queue.