

# 1 Règles d'association

Scikit learn est très pauvre pour la recherche de patterns. Il existe cependant quelques bibliothèques qui ont été développées. La recherche de patterns nécessite souvent des algorithmes très performants pour trouver les résultats. Il est donc courant de devoir passer par une API vers un code développé en C++ par exemple. Nous présentons ici deux approches différentes.

La première utilise une bibliothèque développée en Python qui permet de pouvoir extraire des itemsets et des règles d'association. Il s'agit de la bibliothèque MLxtend : <http://rasbt.github.io/mlxtend/> (<http://rasbt.github.io/mlxtend/>).

La seconde utilise une implémentation d'un algorithme de recherche de règles d'association développé en Java. Il s'agit de SPMF. SPMF propose un très grand choix d'algorithmes pour extraire des règles d'association, des motifs séquentiels, des sous graphes. Il propose également un grand nombre de jeu de données. Il est très utilisé dans la communauté Data Mining : <http://www.philippe-fournier-viger.com/spmf/> (<http://www.philippe-fournier-viger.com/spmf/>).

## Rappel de la problématique de la recherche de règles d'association

Soit  $I = \{i_1, i_2, \dots, i_m\}$  un ensemble d'items. Soit  $T = \{t_1, t_2, \dots, t_n\}$  un ensemble de transactions, telles que  $t_i$  soit un sous-ensemble de  $I$  (i.e.  $t_i \subseteq I$ ). Une règle d'association s'exprime sous la forme :  $X \rightarrow Y$ , où  $X \in T$ ,  $Y \in T$ , et  $X \cap Y \neq \emptyset$ .

Initialement la problématique de la recherche de règles d'association était définie par : A partir d'une base de données de transactions, l'objectif est d'extraire l'ensemble des règles qui sont telles que le support de la règle est  $\geq$  support\_minimal et la confiance (confidence) est  $\geq$  confiance\_minimale où support\_minimal confiance\_minimale sont deux paramètres définis par l'utilisateur final. Cependant au cours des années différentes mesures ont été proposées notamment pour remplacer la confiance.

Le Support d'une règle  $X \rightarrow Y$ , noté  $Support(X, Y)$  indique la fréquence de la règle dans les transactions, i.e. combien de fois les items  $X$  et  $Y$  apparaissent ensemble, i.e  $Support(X \cup Y)$ .

La Confidence d'une règle  $X \rightarrow Y$  est définie par :  $Confidence(X \rightarrow Y) = p(Y|X) = \frac{support(X, Y)}{support(X)}$ . Range =  $[0, 1]$

Différentes mesures ont été proposées dans la littérature.

$$Lift(X \rightarrow Y) = \frac{p(Y|X)}{p(Y)} = \frac{support(X, Y)}{support(X) \times support(Y)} = \frac{Confidence(X \rightarrow Y)}{support(X)}. \text{ Range } [0, \infty]$$

Si des règles ont un lift de 1 cela voudrait dire que la probabilité de l'antécédent et du conséquent sont indépendantes l'une de l'autre. Quand deux événements sont indépendants il est donc difficile d'en tirer une règle. Si le lift est supérieur à 1 cela montre que les deux parties sont dépendantes et qu'une règle est peut être utile pour prédire le conséquent à partir de l'antécédent.

$$Leverage(X \rightarrow Y) = support(X, Y) - support(X) \times support(Y). \text{ Range } = [-1, 1]$$

Une valeur de leverage nulle indique que  $X$  et  $Y$  sont indépendantes.

$$Conviction(X \rightarrow Y) = \frac{1 - support(Y)}{1 - Confidence(X \rightarrow Y)}. \text{ Range } [-1, 1]$$

Une valeur de conviction de 1 indique que les événements sont indépendants.

## 1.1 Utilisation de MLxtend

Dans l'exemple nous considérons l'exercice fait en cours. Les données sont organisées de la manière suivante :

Chaque ligne correspond à un client, les items achetés sont précisés.

In [1]:

```
1  ▼ basket = [['Biscuit', 'Creme', 'Couches',  
2             'Pain', 'Confiture'],  
3             ['Oeufs', 'Salade', 'Pain'],  
4  ▼         ['Pain', 'Biscuit', 'Couches',  
5             'Lait', 'Camembert', 'Biere', 'Poudre de cacao'],  
6  ▼         ['Biere', 'Lait', 'Creme',  
7             'Pain', 'Salade', 'Couches', 'Poudre de cacao'],  
8  ▼         ['Camembert', 'Confiture',  
9             'Poudre de cacao', 'Lait'],  
10 ▼         ['Lait', 'Biere', 'Perrier',  
11            'Oeufs', 'Couches'],  
12 ▼         ['Biscuit', 'Creme', 'Oeufs',  
13            'Pain', 'Couches', 'Lait',  
14            'Poudre de cacao', 'Confiture'],  
15 ▼         ['Camembert', 'Biere', 'Perrier',  
16            'Oeufs', 'Salade', 'Confiture', 'Couches'],  
17 ▼         ['Salade', 'Confiture', 'Pain', 'Oeufs',  
18            'Creme', 'Perrier', 'Biscuit']]  
19
```

executed in 25ms, finished 14:00:53 2020-02-04

Il faut à présent transformer les données pour qu'elles correspondent à une matrice où les items sont les colonnes, les clients les lignes et faire apparaître un 0 ou un 1 lorsqu'un client a acheté ou pas un item.

mlxtend propose une fonction TransactionEncoder pour réaliser cette opération.

In [2]:

```
1  
2  import pandas as pd  
3  from mlxtend.preprocessing import TransactionEncoder  
4  
5  te = TransactionEncoder()  
6  te_ary = te.fit(basket).transform(basket)  
7  df = pd.DataFrame(te_ary, columns=te.columns_)  
8
```

executed in 2.33s, finished 14:00:55 2020-02-04

Pour connaître l'ensemble des itemsets fréquents il suffit d'appeler l'algorithme apriori.

In [3]:

```
1 from mlxtend.frequent_patterns import apriori
2
3 frequent_itemsets = apriori(df, min_support=0.3, use_colnames=True)
4
5 display(frequent_itemsets.head(10))
```

executed in 80ms, finished 14:00:55 2020-02-04

	support	itemsets
0	0.444444	(Biere)
1	0.444444	(Biscuit)
2	0.333333	(Camembert)
3	0.555556	(Confiture)
4	0.666667	(Couches)
5	0.444444	(Creme)
6	0.555556	(Lait)
7	0.555556	(Oeufs)
8	0.666667	(Pain)
9	0.333333	(Perrier)

L'obtention des règles se fait par la fonction association rules. Il est possible de spécifier différentes metrique (Cf. remarque précédente).

In [4]:

```

1  from mlxtend.frequent_patterns import association_rules
2  print ( 'Utilisation de la confiance\n')
3  rules = association_rules(frequent_itemsets,
4                           metric="confidence",
5                           min_threshold=0.2)
6
7  display(rules.sample(10))

```

executed in 52ms, finished 14:00:55 2020-02-04

Utilisation de la confiance

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
135	(Lait, Couches)	(Poudre de cacao, Pain)	0.444444	0.333333	0.333333	0.75	2.250	0.185185
133	(Pain, Lait)	(Poudre de cacao, Couches)	0.333333	0.333333	0.333333	1.00	3.000	0.222222
6	(Biscuit)	(Couches)	0.444444	0.666667	0.333333	0.75	1.125	0.037037
47	(Biere, Couches)	(Lait)	0.444444	0.555556	0.333333	0.75	1.350	0.086420
88	(Pain, Lait)	(Couches)	0.333333	0.666667	0.333333	1.00	1.500	0.111111
139	(Couches)	(Poudre de cacao, Pain, Lait)	0.666667	0.333333	0.333333	0.50	1.500	0.111111
68	(Pain)	(Biscuit, Couches)	0.666667	0.333333	0.333333	0.50	1.500	0.111111
86	(Couches)	(Pain, Creme)	0.666667	0.444444	0.333333	0.50	1.125	0.037037
138	(Lait)	(Poudre de cacao, Pain, Couches)	0.555556	0.333333	0.333333	0.60	1.800	0.148148
60	(Confiture, Pain)	(Biscuit)	0.333333	0.444444	0.333333	1.00	2.250	0.185185

In [5]:

```

1 print ('Utilisation du lift\n')
2 rules = association_rules(frequent_itemsets,
3                           metric="lift")
4
5 display(rules.head(10))

```

executed in 57ms, finished 14:00:55 2020-02-04

Utilisation du lift

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
0	(Couches)	(Biere)	0.666667	0.444444	0.444444	0.666667	1.5000	0.148148
1	(Biere)	(Couches)	0.444444	0.666667	0.444444	1.000000	1.5000	0.148148
2	(Lait)	(Biere)	0.555556	0.444444	0.333333	0.600000	1.3500	0.086420
3	(Biere)	(Lait)	0.444444	0.555556	0.333333	0.750000	1.3500	0.086420
4	(Biscuit)	(Confiture)	0.444444	0.555556	0.333333	0.750000	1.3500	0.086420
5	(Confiture)	(Biscuit)	0.555556	0.444444	0.333333	0.600000	1.3500	0.086420
6	(Biscuit)	(Couches)	0.444444	0.666667	0.333333	0.750000	1.1250	0.037037
7	(Couches)	(Biscuit)	0.666667	0.444444	0.333333	0.500000	1.1250	0.037037
8	(Biscuit)	(Creme)	0.444444	0.444444	0.333333	0.750000	1.6875	0.135802
9	(Creme)	(Biscuit)	0.444444	0.444444	0.333333	0.750000	1.6875	0.135802

Le code suivant permet de pouvoir ajouter une colonne qui contient la taille de la partie antecedent pour ne pouvoir afficher que les règles qui ont un certain nombre d'items dans la partie antécédent.

In [6]:

```

1
2 rules["antecedent_len"] = rules["antecedents"].apply(lambda x: len(x))
3

```

executed in 14ms, finished 14:00:55 2020-02-04

Affichage des règles dont la partie antécédent est supérieure à 1.

In [7]:

```
1 display (rules.loc[(rules["antecedent_len"]>1)].head(10))
```

executed in 55ms, finished 14:00:55 2020-02-04

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
46	(Lait, Couches)	(Biere)	0.444444	0.444444	0.333333	0.75	1.6875	0.135802
47	(Biere, Couches)	(Lait)	0.444444	0.555556	0.333333	0.75	1.3500	0.086420
48	(Lait, Biere)	(Couches)	0.333333	0.666667	0.333333	1.00	1.5000	0.111111
52	(Biscuit, Confiture)	(Creme)	0.333333	0.444444	0.333333	1.00	2.2500	0.185185
53	(Biscuit, Creme)	(Confiture)	0.333333	0.555556	0.333333	1.00	1.8000	0.148148
54	(Confiture, Creme)	(Biscuit)	0.333333	0.444444	0.333333	1.00	2.2500	0.185185
58	(Biscuit, Confiture)	(Pain)	0.333333	0.666667	0.333333	1.00	1.5000	0.111111
59	(Biscuit, Pain)	(Confiture)	0.444444	0.555556	0.333333	0.75	1.3500	0.086420
60	(Confiture, Pain)	(Biscuit)	0.333333	0.444444	0.333333	1.00	2.2500	0.185185
64	(Biscuit, Pain)	(Couches)	0.444444	0.666667	0.333333	0.75	1.1250	0.037037

## 1.2 Utilisation d'une API extérieure (SPMF)

Le code java de SPMF est disponible ici :

<http://www.philippe-fournier-viger.com/spmf/index.php?link=download.php> (<http://www.philippe-fournier-viger.com/spmf/index.php?link=download.php>)

Il faut le télécharger et sauvegarder le fichier jar dans le répertoire courant.

Comme vous pouvez le constater il existe un très grand nombre d'algorithmes de recherche de règles d'association disponibles. Nous allons utiliser ici FP\_Growth.

Pour obtenir les itemsets : FPGrowth\_itemsets

Pour obtenir les règles d'association : FPGrowth\_association\_rules

Pour obtenir les règles d'association avec le lift : FPGrowth\_association\_rules\_with\_lift

Dans l'exemple nous utiliserons le dernier. Se reporter à la page de documentation pour connaître les différents paramètres.

L'appel se fait simplement via :

```
java -jar spmf.jar run FPGrowth_association_rules_with_lift NomduFichierFormatSPMF Sortie.txt support%
confiance% valeurlift"
```

Dans l'exemple nous prenons le fichier contextIGB.txt, support 50%, confiance 90%, lift 1.2.

In [8]:

```
1 import os
2
3 os.system("java -jar spmf.jar run FPGrowth_association_rules_with_lift Datas
4
5
6
```

executed in 940ms, finished 14:00:56 2020-02-04

Out[8]:

0

In [9]:

```
1  ▾ #Affichage des résultats
2  f=open("output.txt")
3  ▾ for ln in f:
4      display (ln)
5  f.close()
```

executed in 27ms, finished 14:00:56 2020-02-04

```
'1 ==> 5 #SUP: 4 #CONF: 1.0 #LIFT: 1.2\n'
'1 2 ==> 5 #SUP: 4 #CONF: 1.0 #LIFT: 1.2\n'
'4 5 ==> 1 #SUP: 3 #CONF: 1.0 #LIFT: 1.5\n'
'1 4 ==> 5 #SUP: 3 #CONF: 1.0 #LIFT: 1.2\n'
'2 4 5 ==> 1 #SUP: 3 #CONF: 1.0 #LIFT: 1.5\n'
'1 2 4 ==> 5 #SUP: 3 #CONF: 1.0 #LIFT: 1.2\n'
```