Association Rule Learning

Apriori

Apriori Intuition

What are the three essential relations between the support, confidence and lift?

Given two movies *M*1 and *M*2, here are the three essential relations to remember:

Relation between the support and the confidence:

confidence(*M*1 → *M*2) = support(*M*1*,M*2) support(*M*1)

Relation between the lift and the support:

support(*M*1*,M*2)

lift(*M*1 → *M*2) = support(*M*1) × support(*M*2)

Relation between the lift and the confidence (consequence of the two previous equations):

lift(*M*1 → *M*2) = confidence(*M*1 → *M*2) support(*M*2)

Are the confidence and lift symmetrical functions?

Given the three equations of the previous question, we can easily see that:

Confidence is non symmetrical:

confidence(*M*1 → *M*2) 6= confidence(*M*2 → *M*1)

Lift is symmetrical: lift(*M*1 → *M*2) = lift(*M*2 → *M*1)

Apriori in Python

Important question: is R better than Python for Association Rule Learning?

Yes, absolutely. The R package is much more optimized and easy to use. But for R haters we made sure to provide a Python solution. However if you don’t mind, we highly recommend to go with R.

I have trouble finding the rules in Python. How can I see them more explicitly?

Sorry about that, the package version changed after I recorded the Lecture. But you can see them by just adding the following at the end of your Python code (in a new section right after the "Visualising the results" section):

|  |
| --- |
| the\_rules = []  for result in results:  the\_rules.append({’rule’: ’,’.join(result.items),  ’support’:result.support, ’confidence’:result.ordered\_statistics[0].confidence,  ’lift’:result.ordered\_statistics[0].lift})  df = pd.DataFrame(the\_rules, columns = [’rule’, ’support’, ’confidence’, ’lift’]) |

5

After execution of this code, you should see a DataFrame called ’df’ appearing in the ’Variable Explorer’ with all the required information.

Eclat

Eclat Intuition

When should we use Eclat rather than Apriori?

The only advantage of Eclat compared to Apriori is that it is simpler and faster to use. However if you need to run a deep analysis of your market basket, then you should definitely go for Apriori.

Eclat in Python

Could you provide an Eclat implementation in Python?

Credit to Marcello Morchio, a student of this course who kindly shared his implementation:

|  |
| --- |
| *# Eclat, by Marcello Morchio, December 4th 2017*  *# Importing the libraries* import numpy as np  import matplotlib.pyplot as plt  import pandas as pd |

5

|  |
| --- |
| *# Importing the dataset*  dataset = pd.read\_csv(’Market\_Basket\_Optimisation.csv’, header = None ) transactions = [[str(dataset.values[i,j]) for j in range(0, dataset.shape[1])  if str(dataset.values[i,j]) != ’nan’] for i in range(0, dataset.shape[0])]  *# Generate a list of unique items in the transactions* items = list() for t in transactions:  for x in t:  if(not x in items):  items.append(x)  *# Generate a list of pairs of items with relevant support value*  *# [[(item\_a, item\_b) , support\_value]]*  *# support\_value is initialized to 0 for all pairs* eclat = list() for i in range(0, len(items)):  for j in range(i+1, len(items)):  eclat.append([(items[i],items[j]),0])  *# Compute support value for each pair by looking for transactions with both items* for p in eclat:  for t in transactions:  if(p[0][0] in t) and (p[0][1] in t):  p[1] += 1  p[1] = p[1]/len(transactions)  *# Converts eclat in sorted DataFrame to be visualized in variable explorer* eclatDataFrame = pd.DataFrame(eclat,  columns = [’rule’,’support’]).sort\_values(by = ’support’, ascending = False) |

10

15

20

25

30

35