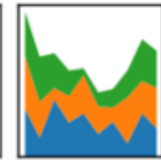
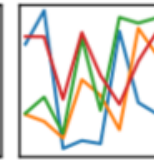




pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



MLXTEND

python

```
In [22]: rules = association_rules(frequent_itemsets, metric="Lift", min_threshold=1)
rules
```

```
Out[22]:
```

	antecedents	consequents	support	confidence	lift
0	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE)	0.170918	0.597025	3.545907
1	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.168367	0.606061	3.545907
2	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN SPACEBOY)	0.168367	0.530303	3.849607
3	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN CIRCUS PARADE)	0.137755	0.648148	3.849607
4	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.170918	0.611940	4.442233
5	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.137755	0.756258	4.442233

pbpython.com

Introduction

There are many data analysis tools available to the python analyst and it can be challenging to know which ones to use in a particular situation. A useful (but somewhat overlooked) technique is called association analysis which attempts to find common patterns of items in large data sets. One specific application is often called market basket analysis. The most commonly cited example of market basket analysis is the so-called "beer and diapers" case. The basic story is that a large retailer was able to mine their transaction data and find an unexpected purchase pattern of individuals that were buying beer and baby diapers at the same time.

Unfortunately this story is most likely a data urban legend (<http://www.dssresources.com/newsletters/66.php>). However, it is an illustrative (and entertaining) example of the types of insights that can be gained by mining transactional data.

While these types of associations are normally used for looking at sales transactions; the basic analysis can be applied to other situations like click stream tracking, spare parts ordering and online recommendation engines - just to name a few.

If you have some basic understanding of the python data science world, your first inclination would be to look at scikit-learn for a ready-made algorithm. However, scikit-learn does not support this algorithm. Fortunately, the very useful MLxtend (<http://rasbt.github.io/mlxtend/>) library by Sebastian Raschka has a an implementation of the Apriori algorithm (http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/) for extracting frequent item sets for further analysis.

The rest of this article will walk through an example of using this library to analyze a relatively large online retail (<http://archive.ics.uci.edu/ml/datasets/Online+Retail>) data set and try to find interesting purchase combinations. By the end of this article, you should be familiar enough with the basic approach to apply it to your own data sets.

Why Association Analysis?

In today's world, there are many complex ways to analyze data (clustering, regression, Neural Networks, Random Forests, SVM, etc.). The challenge with many of these approaches is that they can be difficult to tune, challenging to interpret and require quite a bit of data prep and feature engineering to get good results. In other words, they can be very powerful but require a lot of knowledge to implement properly.

Association analysis is relatively light on the math concepts and easy to explain to non-technical people. In addition, it is an unsupervised learning tool that looks for hidden patterns so there is limited need for data prep and feature engineering. It is a good start for certain cases of data exploration and can point the way for a deeper dive into the data using other approaches.

As an added bonus, the python implementation in MLxtend should be very familiar to anyone that has exposure to scikit-learn and pandas. For all these reasons, I think it is a useful tool to be familiar with and can help you with your data analysis problems.

One quick note - technically, market basket analysis is just one application of association analysis. In this post though, I will use association analysis and market basket analysis interchangeably.

Association Analysis 101

There are a couple of terms used in association analysis that are important to understand. This chapter (<http://www-users.cs.umn.edu/~kumar/dmbook/ch6.pdf>) in Introduction to Data Mining (<http://www-users.cs.umn.edu/~kumar/dmbook/index.php>) is a great reference for those interested in the math behind these definitions and the details of the algorithm implementation.

Association rules are normally written like this: {Diapers} -> {Beer} which means that there is a strong relationship between customers that purchased diapers and also purchased beer in the same transaction.

In the above example, the {Diaper} is the **antecedent** and the {Beer} is the **consequent**. Both antecedents and consequents can have multiple items. In other words, {Diaper, Gum} -> {Beer, Chips} is a valid rule.

Support is the relative frequency that the rules show up. In many instances, you may want to look for high support in order to make sure it is a useful relationship. However, there may be instances where a low support is useful if you are trying to find “hidden” relationships.

Confidence is a measure of the reliability of the rule. A confidence of .5 in the above example would mean that in 50% of the cases where Diaper and Gum were purchased, the purchase also included Beer and Chips. For product recommendation, a 50% confidence may be perfectly acceptable but in a medical situation, this level may not be high enough.

Lift is the ratio of the observed support to that expected if the two rules were independent (see wikipedia (https://en.wikipedia.org/wiki/Association_rule_learning)). The basic rule of thumb is that a lift value close to 1 means the rules were completely independent. Lift values > 1 are generally more “interesting” and could be indicative of a useful rule pattern.

One final note, related to the data. This analysis requires that all the data for a transaction be included in 1 row and the items should be 1-hot encoded. The MLxtend documentation (http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/) example is useful:

	Apple	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogurt
0	0	0	0	1	0	1	1	1	1	0	1
1	0	0	1	1	0	1	0	1	1	0	1

	Apple	Corn	Dill	Eggs	Ice cream	Kidney Beans	Milk	Nutmeg	Onion	Unicorn	Yogurt
2	1	0	0	1	0	1	1	0	0	0	0
3	0	1	0	0	0	1	1	0	0	1	1
4	0	1	0	1	1	1	0	0	1	0	0

The specific data (<http://archive.ics.uci.edu/ml/datasets/Online+Retail>) for this article comes from the UCI Machine Learning Repository and represents transactional data from a UK retailer from 2010-2011. This mostly represents sales to wholesalers so it is slightly different from consumer purchase patterns but is still a useful case study.

Let's Code

MLxtend can be installed using pip, so make sure that is done before trying to execute any of the code below. Once it is installed, the code below shows how to get it up and running. I have made the notebook

(https://github.com/chris1610/pbpython/blob/master/notebooks/Market_Basket_Intro.ipynb) available so feel free to follow along with the examples below.

Get our pandas and MLxtend code imported and read the data:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

df = pd.read_excel('http://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx (http://ar
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

There is a little cleanup, we need to do. First, some of the descriptions have spaces that need to be removed. We'll also drop the rows that don't have invoice numbers and remove the credit transactions (those with invoice numbers containing C).

```
df['Description'] = df['Description'].str.strip()
df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)
df['InvoiceNo'] = df['InvoiceNo'].astype('str')
df = df[~df['InvoiceNo'].str.contains('C')]
```

After the cleanup, we need to consolidate the items into 1 transaction per row with each product 1 hot encoded (<http://pbpython.com/categorical-encoding.html>). For the sake of keeping the data set small, I'm only looking at sales for France. However, in additional code below, I will compare these results to sales from Germany. Further country comparisons would be interesting to investigate.

```
basket = (df[df['Country'] == "France"]
          .groupby(['InvoiceNo', 'Description'])['Quantity']
          .sum().unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
```

Here's what the first few columns look like (note, I added some numbers to the columns to illustrate the concept - the actual data in this example is all 0's):

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY
InvoiceNo								
536370	11.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
536852	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0
536974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
537065	0.0	0.0	0.0	0.0	0.0	7.0	0.0	0.0
537463	0.0	0.0	9.0	0.0	0.0	0.0	0.0	0.0

There are a lot of zeros in the data but we also need to make sure any positive values are converted to a 1 and anything less the 0 is set to 0. This step will complete the one hot encoding of the data and remove the postage column (since that charge is not one we wish to explore):

```
def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
basket_sets.drop('POSTAGE', inplace=True, axis=1)
```

Now that the data is structured properly, we can generate frequent item sets that have a support of at least 7% (this number was chosen so that I could get enough useful examples):

```
frequent_itemsets = apriori(basket_sets, min_support=0.07, use_colnames=True)
```

The final step is to generate the rules with their corresponding support, confidence and lift:

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.head()
```

	antecedants	consequents	support	confidence	lift
0	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN CIRCUS PARADE)	0.170918	0.597015	3.545907
1	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.168367	0.606061	3.545907
2	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN SPACEBOY)	0.168367	0.530303	3.849607
3	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN CIRCUS PARADE)	0.137755	0.648148	3.849607
4	(PLASTERS IN TIN WOODLAND ANIMALS)	(PLASTERS IN TIN SPACEBOY)	0.170918	0.611940	4.442233

That's all there is to it! Build the frequent items using `apriori` then build the rules with `association_rules` .

Now, the tricky part is figuring out what this tells us. For instance, we can see that there are quite a few rules with a high lift value which means that it occurs more frequently than would be expected given the number of transaction and product combinations. We can also see several where the confidence is high as well. This part of the analysis is where the domain knowledge will come in handy. Since I do not have that, I'll just look for a couple of illustrative examples.

We can filter the dataframe using standard pandas code. In this case, look for a large lift (6) and high confidence (.8):

```
rules[ (rules['lift'] >= 6) &
       (rules['confidence'] >= 0.8) ]
```

	antecedants	consequents	support	confidence	lift
8	(SET/6 RED SPOTTY PAPER CUPS)	(SET/6 RED SPOTTY PAPER PLATES)	0.137755	0.888889	6.968889
9	(SET/6 RED SPOTTY PAPER PLATES)	(SET/6 RED SPOTTY PAPER CUPS)	0.127551	0.960000	6.968889

	antecedants	consequents	support	confidence	lift
10	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.096939	0.815789	8.642959
11	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.094388	0.837838	8.642959
16	(SET/6 RED SPOTTY PAPER CUPS, SET/6 RED SPOTTY...	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.122449	0.812500	6.125000
17	(SET/6 RED SPOTTY PAPER CUPS, SET/20 RED RETRO...	(SET/6 RED SPOTTY PAPER PLATES)	0.102041	0.975000	7.644000
18	(SET/6 RED SPOTTY PAPER PLATES, SET/20 RED RET...	(SET/6 RED SPOTTY PAPER CUPS)	0.102041	0.975000	7.077778
22	(SET/6 RED SPOTTY PAPER PLATES)	(SET/20 RED RETROSPOT PAPER NAPKINS)	0.127551	0.800000	6.030769

In looking at the rules, it seems that the green and red alarm clocks are purchased together and the red paper cups, napkins and plates are purchased together in a manner that is higher than the overall probability would suggest.

At this point, you may want to look at how much opportunity there is to use the popularity of one product to drive sales of another. For instance, we can see that we sell 340 Green Alarm clocks but only 316 Red Alarm Clocks so maybe we can drive more Red Alarm Clock sales through recommendations?

```

basket['ALARM CLOCK BAKELIKE GREEN'].sum()

340.0

basket['ALARM CLOCK BAKELIKE RED'].sum()

316.0

```

What is also interesting is to see how the combinations vary by country of purchase. Let's check out what some popular combinations might be in Germany:


```

basket2 = (df[df['Country'] == "Germany"]
           .groupby(['InvoiceNo', 'Description'])['Quantity']
           .sum().unstack().reset_index().fillna(0)
           .set_index('InvoiceNo'))

basket_sets2 = basket2.applymap(encode_units)
basket_sets2.drop('POSTAGE', inplace=True, axis=1)
frequent_itemsets2 = apriori(basket_sets2, min_support=0.05, use_colnames=True)
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)

rules2[ (rules2['lift'] >= 4) &
        (rules2['confidence'] >= 0.5)]

```

	antecedants	consequents	support	confidence	lift
7	(PLASTERS IN TIN SPACEBOY)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.107221	0.571429	4.145125
9	(PLASTERS IN TIN CIRCUS PARADE)	(PLASTERS IN TIN WOODLAND ANIMALS)	0.115974	0.584906	4.242887
10	(RED RETROSPOT CHARLOTTE BAG)	(WOODLAND CHARLOTTE BAG)	0.070022	0.843750	6.648168

It seems that in addition to David Hasselhoff, Germans love Plasters in Tin Spaceboy and Woodland Animals.

In all seriousness, an analyst that has familiarity with the data would probably have a dozen different questions that this type of analysis could drive. I did not replicate this analysis for additional countries or customer combos but the overall process would be relatively simple given the basic pandas code shown above.



Conclusion

The really nice aspect of association analysis is that it is easy to run and relatively easy to interpret. If you did not have access to MLxtend and this association analysis, it would be exceedingly difficult to find these patterns using basic Excel analysis. With python and MLxtend, the analysis process is relatively straightforward and since you are in python, you have access to all the additional visualization techniques and data analysis tools in the python ecosystem.


Finally, I encourage you to check out the rest of the MLxtend library. If you are doing any work in sci-kit learn it is helpful to be familiar with MLxtend and how it could augment some of the existing tools in your data science toolkit.

← How Accurately Can Prophet Project Website Traffic? (<http://pbpython.com/prophet-accuracy.html>)

Pandas Grouper and Agg Functions Explained → (<http://pbpython.com/pandas-grouper-agg.html>)

Tags  pandas (<http://pbpython.com/tag/pandas.html>)  mlxtend (<http://pbpython.com/tag/mlxtend.html>)

Tweet

 Share

Vote on Hacker News (<https://news.ycombinator.com/submit>)

Share


  3 points

Comments

58 Comments pbpython.com

 1 Login ▾

 Recommend 6

 Share

Sort by Best ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Jarad Collier • 7 months ago

This is a phenomenal post! I coded a R solution because the apriori algorithm isn't extensively ported in Python it seems. I wish mlxtend had some more of the features that R does such as: remove redundant rules, plot a network flow graph (example on this page: <http://www.kdnuggets.com/20....> MLxtend, although minimal, is the only one I know about so far. Anyone know any others with more depth to do market basket analysis like R ?