IT VEDANT

TOPIC: ASSOCIATION RULE

1. What is Association Rule Learning?

Ans: Association rules are "if-then" statements, that help to show the probability of relationships between data items, within large data sets in various types of databases. Association rule mining has a number of applications and is widely used to help discover sales correlations in transactional data or in medical data sets.

Use cases for association rules

In data science, association rules are used to find correlations and co-occurrences between data sets. They are ideally used to explain patterns in data from seemingly independent information repositories, such as relational databases and transactional databases. The act of using association rules is sometimes referred to as "association rule mining" or "mining associations."

Below are a few real-world use cases for association rules:

- **Medicine.** Doctors can use association rules to help diagnose patients. There are many variables to consider when making a diagnosis, as many diseases share symptoms. By using association rules and machine learning-fueled data analysis, doctors can determine the conditional probability of a given illness by comparing symptom relationships in the data from past cases. As new diagnoses get made, machine learning models can adapt the rules to reflect the updated data.
- **Retail.** Retailers can collect data about purchasing patterns, recording purchase data as item barcodes are scanned by point-of-sale systems. Machine learning models can look for co-occurrence in this data to determine which products are most likely to be purchased together. The retailer can then adjust marketing and sales strategy to take advantage of this information.
- **User experience (UX) design.** Developers can collect data on how consumers use a website they create. They can then use associations in the data to optimize the website user interface -- by analyzing where users tend to click and what maximizes the chance that they engage with a call to action, for example.
- **Entertainment.** Services like Netflix and Spotify can use association rules to fuel their content recommendation engines. Machine learning models analyze past user behavior data for frequent patterns, develop association rules and use those rules to recommend content that a user is likely to engage with, or organize content in a way that is likely to put the most interesting content for a given user first.

How association rules work

Association rule mining, at a basic level, involves the use of machine learning models to analyze data for patterns, or co-occurrences, in a database. It identifies frequent if-then associations, which themselves are the *association rules*.

An association rule has two parts: an antecedent (if) and a consequent (then). An antecedent is an item found within the data. A consequent is an item found in combination with the antecedent.

Association rules are created by searching data for frequent if-then patterns and using the criteria *support* and *confidence* to identify the most important relationships. Support is an indication of how frequently the items appear in the data. Confidence indicates the number of times the if-then statements are found true. A third metric, called *lift*, can be used to compare confidence with expected confidence, or how many times an if-then statement is expected to be found true.

Association rules are calculated from *itemsets*, which are made up of two or more items. If rules are built from analyzing all the possible itemsets, there could be so many rules that the rules hold little meaning. With that, association rules are typically created from rules well-represented in data.

Measures of the effectiveness of association rules

The strength of a given association rule is measured by two main parameters: support and confidence. Support refers to how often a given rule appears in the database being mined. Confidence refers to the amount of times a given rule turns out to be true in practice. A rule may show a strong correlation in a data set because it appears very often but may occur far less when applied. This would be a case of high support, but low confidence.

Conversely, a rule might not particularly stand out in a data set, but continued analysis shows that it occurs very frequently. This would be a case of high confidence and low support. Using these measures helps analysts separate causation from correlation and allows them to properly value a given rule.

A third value parameter, known as the lift value, is the ratio of confidence to support. If the lift value is a negative value, then there is a negative correlation between datapoints. If the value is positive, there is a positive correlation, and if the ratio equals 1, then there is no correlation.

Association rule algorithms

Popular algorithms that use association rules include AIS, SETM, Apriori and variations of the latter.

With the AIS algorithm, itemsets are generated and counted as it scans the data. In transaction data, the AIS algorithm determines which large itemsets contained a transaction, and new candidate itemsets are created by extending the large itemsets with other items in the transaction data.

The SETM algorithm also generates candidate itemsets as it scans a database, but this algorithm accounts for the itemsets at the end of its scan. New candidate itemsets are generated the same way as with the AIS algorithm, but the transaction ID of the generating transaction is saved with the candidate itemset in a sequential data structure. At the end of the pass, the support count of candidate itemsets is created by aggregating the sequential structure. The downside of both the AIS and SETM algorithms is that each one can generate and count many small candidate itemsets, according to published materials from Dr. Saed Sayad, author of *Real Time Data Mining*.

With the Apriori algorithm, candidate itemsets are generated using only the large itemsets of the previous pass. The large itemset of the previous pass is joined with itself to generate all itemsets with a size that's larger by one. Each generated itemset with a subset that is not large is then deleted. The remaining itemsets are the candidates. The Apriori algorithm considers any subset of a frequent itemset to also be a frequent itemset. With this approach, the algorithm reduces the number of candidates being considered by only exploring the itemsets whose support count is greater than the minimum support count, according to Sayad.

Uses of association rules in data mining

In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important part in customer analytics, market basket analysis, product clustering, catalog design and store layout.

Programmers use association rules to build programs capable of machine learning. Machine learning is a type of artificial intelligence (AI) that seeks to build programs with the ability to become more efficient without being explicitly programmed.

Examples of association rules in data mining

A classic example of association rule mining refers to a relationship between diapers and beers. The example, which seems to be fictional, claims that men who go to a store to buy diapers are also likely to buy beer. Data that would point to that might look like this:

A supermarket has 200,000 customer transactions. About 4,000 transactions, or about 2% of the total number of transactions, include the purchase of diapers. About 5,500 transactions (2.75%) include the purchase of beer. Of those, about 3,500 transactions, 1.75%, include both the purchase of diapers and beer. Based on the percentages, that large number should be much lower. However, the fact that about 87.5% of diaper purchases include the purchase of beer indicates a link between diapers and beer.

History

While the concepts behind association rules can be traced back earlier, association rule mining was defined in the 1990s, when computer scientists Rakesh Agrawal, Tomasz Imieliński and Arun Swami developed an algorithm-based way to find relationships between items using point-of-sale (POS) systems. Applying the algorithms to supermarkets, the scientists were able to discover links between different items purchased, called *association rules*, and ultimately use that information to predict the likelihood of different products being purchased together.

For retailers, association rule mining offered a way to better understand customer purchase behaviors. Because of its retail origins, association rule mining is often referred to as *market basket analysis*.

As advances in data science, AI and machine learning, have occurred since the original use case for association rules -- and more devices generate data -- association rules can be used in wider breadth of use cases. More data is being generated, meaning more applications for association rules. AI and machine learning allow for larger and more complex data sets to be analyzed and mined for association rules.

2. Explain the following in context of Association Rule Learning:

a. Support

b. Confidence

c. Lift

d. Conviction

Ans: Association Rule Learning

As briefly mentioned in the introduction, association rule learning is a rule-based machine learning method for discovering interesting relations between variables in large databases. Let's use a simple supermarket shopping basket analysis to explain how the association rules are found.

	Item 1	Item 2	Item 3
Shopper 1	Eggs	Bacon	Soup
Shopper 2	Eggs	Bacon	Apple
Shopper 3	Eggs	Bacon	Apple
Shopper 4	Soup	Bacon	Banana
Shopper 5	Banana	Butter	-
Shopper 6	Butter	-	-

Supermarket purchase list by shoppers. Image by author.

Assume we analyze the above transaction data to find frequently bought items and determine if they are often purchased together. To help us find the answers, we will make use of the following 4 metrics:

- Support
- Confidence
- Lift
- Conviction

Support

The first step for us and the algorithm is to find frequently bought items. It is a straightforward calculation that is based on frequency:

```
Support(A) = Transactions(A) / Total Transactions
```

So in our example:

```
Support(Eggs) = 3/6 = 1/2 = 0.5
Support(Bacon) = 4/6 = 2/3 = 0.667
Support(Apple) = 2/6 = 1/3 = 0.333
...
Support(Eggs&Bacon) = 3/6 = 0.5
...
```

Here we can set our first constraint by telling the algorithm the minimum support level we want to explore, which is useful when working with large datasets. We typically want to focus computing resources to search for associations between frequently bought items while discounting the infrequent ones.

For the sake of our example, let's **set minimum support to 0.5**, which leaves us to work with Eggs and Bacon for the rest of this example.

Important: while Support(Eggs) and Support(Bacon) individually satisfy our minimum support constraint, it is crucial to understand that we also need the combination of them (Eggs&Bacon) to pass this constraint. Otherwise, we would not have a single item pairing to progress forward to create association rules.

Confidence

Now that we have identified frequently bought items let's calculate confidence. This will tell us how confident (based on our data) we can be that an item will be purchased, given that another item has been purchased.

```
Confidence(A→B) = Probability(A & B) / Support(A)Note, confidence is the same as what is also known as conditional probability in statistics:
P(B|A) = P(A & B) / P(A) Please beware of the notation. The above two
```

equeations are equivalent, although the notations are in different order: ***(A→B)*** *is the same as* ***(B|A).***

So, let's calculate confidence for our example:

Confidence(Eggs→Bacon) = P(Eggs & Bacon) / Support(Eggs) = (3/6) / (3/6) = 1**Confidence(Bacon→Eggs)** = P(Eggs & Bacon) / Support(Bacon) = (3/6) / (2/3) = 3/4 = 0.75

The above tells us that whenever eggs are bought, bacon is also bought 100% of the time. Also, whenever bacon is bought, eggs are bought 75% of the time.

Lift

Given that different items are bought at different frequencies, how do we know that eggs and bacon really do have a strong association, and how do we measure it? You will be glad to hear that we have a way to evaluate this objectively using **lift**.

There are multiple ways to express the formula to calculate lift. Let me first show what the formulas look like, and then I will describe an intuitive way for you to think about it.

Lift(A→B) = Probability(A & B) / (Support(A) * Support(B))You should be able to spot that we can simplify this formula by replacing P(A&B)/Sup(A) with Confidence(A→B). Hence, we have:**Lift(A→B)** = Confidence(A→B) / Support(B)

Let's calculate lift for our associated items:

Lift(Eggs→Bacon) = Confidence(Eggs→Bacon) / Support(Bacon) = 1 / (2/3) = 1.5**Lift(Bacon→Eggs)** = Confidence(Bacon→Eggs) / Support(Eggs) = (3/4) / (1/2) = 1.5

Lift for the two items is equal to 1.5. Note, lift>1 means that the two items are more likely to be bought together, while lift<1 means that the two items are more likely to be bought separately. Finally, lift=1 means that there is no association between the two items.

An intuitive way to understand this would be to first think about the probability of eggs being bought: `P(Eggs)=Support(Eggs)=0.5` as 3 out of 6 shoppers bought eggs.

Then think about the probability of eggs being bought whenever bacon was bought: `P(Eggs|Bacon)=Confidence(Bacon->Eggs)=0.75` since out of the 4 shoppers that bought bacon, 3 of them also bought eggs.

Now, lift is simply a measure that tells us whether the probability of buying eggs increases or decreases given the purchase of bacon. Since the probability of buying eggs in such a scenario goes up from 0.5 to 0.75, we see a positive lift of 1.5 times (0.75/0.5=1.5). This means you are 1.5 times (i.e., 50%) more likely to buy eggs if you have already put bacon into your basket.

See if your supermarket places these two items nearby. 😜

Conviction

Conviction is another way of measuring association, although it is a bit harder to get your head around. It compares the probability that A appears without B if they were independent with the actual frequency of the appearance of A without B. Let's take a look at the general formula first:

```
Conviction(A→B) = (1 - Support(B)) / (1 - Confidence(A→B))
```

In our example, this would be:

```
Conviction(Eggs→Bacon) = (1 - Sup(Bacon) / (1 - Conf(Eggs→Bacon)) = (1 - 2/3) / (1 - 1) = (1/3) / 0 = infinityConviction(Bacon→Eggs) = (1 - Sup(Eggs) / (1 - Conf(Bacon→Eggs)) = (1 - 1/2) / (1 - 3/4) = (1/2) / (1/4) = 2
```

As you can see, we had a division by 0 when calculating conviction for (Eggs→Bacon) and this is because we do not have a single instance of eggs being bought without bacon (confidence=100%).

In general, high confidence for A→B with low support for item B would yield a high conviction.

In contrast to lift, conviction is a directed measure. Hence, while lift is the same for both (Eggs→Bacon) and (Bacon→Eggs), conviction is different between the two, with Conv(Eggs→Bacon) being much higher. Thus, you can use conviction to evaluate the directional relationship between your items.

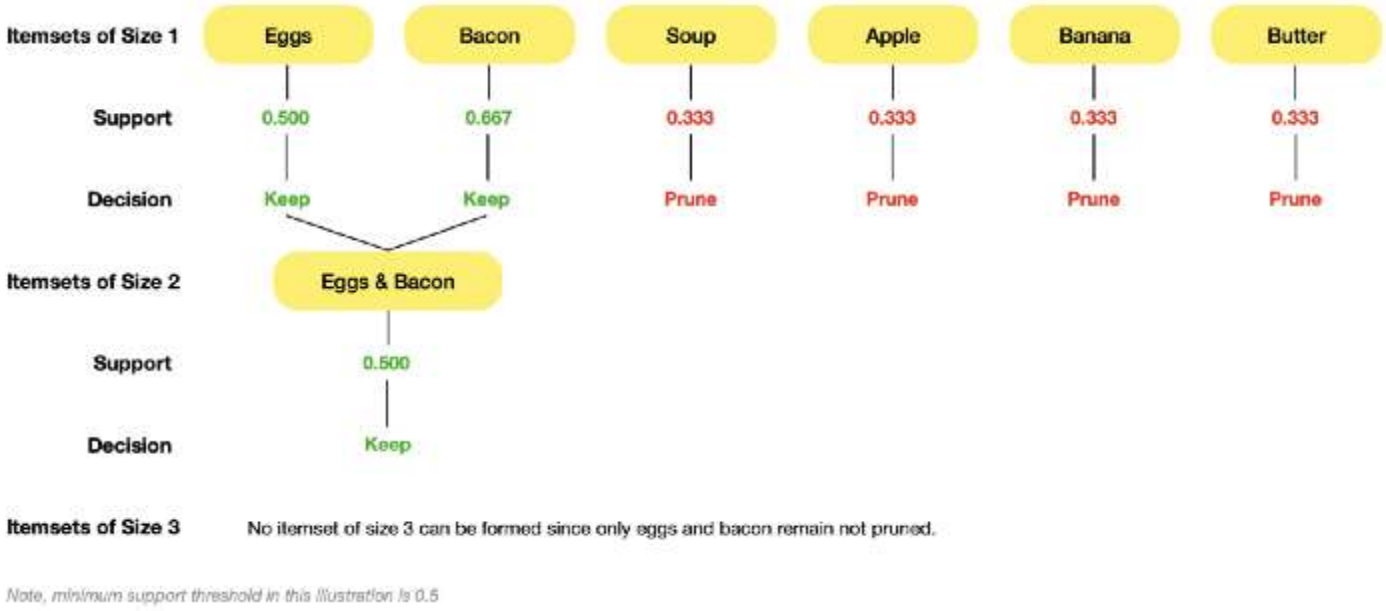
Finally, similar to lift, conviction=1 means that items are not associated, while conviction>1 indicates the relationship between the items (the higher the value, the stronger the relationship).

Apriori algorithm

Apriori is a pretty straightforward algorithm that performs the following sequence of calculations:

1. Calculate support for itemsets of size 1.
2. Apply the minimum support threshold and prune itemsets that do not meet the threshold.
3. Move on to itemsets of size 2 and repeat steps one and two.
4. Continue the same process until no additional itemsets satisfying the minimum threshold can be found.

To make the process more visual, here is a diagram that illustrates what the algorithm does:



Process tree of Apriori algorithm. Image by author.

As you can see, most itemsets in this example got pruned since they did not meet the minimum support threshold of 0.5.

It is important to realize that by setting a lower minimum support threshold we would produce many more itemsets of size 2. To be precise, with a minimum support threshold of 0.3, none of the itemsets of size 1 would get pruned giving us a total of 15 itemsets of size 2 (5+4+3+2+1=15).

This is not an issue when we have a small dataset, but it could become a bottleneck if you are working with a large dataset. E.g., 1,000 items can create as many as 499,500 item pairs. Hence, choose your minimum support threshold carefully.