Association rules are a key concept in data mining, used to discover interesting relationships, patterns, and correlations among a set of items in large datasets. They are commonly used in market basket analysis to identify products that frequently co-occur in transactions.

Components of Association Rules

An association rule is usually expressed in the form: A→BA \rightarrow BA→B where AAA and BBB are itemsets (sets of items). The rule suggests that if itemset AAA occurs in a transaction, then itemset BBB is likely to occur in the same transaction.

Key Metrics

To evaluate the strength and usefulness of association rules, several metrics are used:

1. **Support**: This measures how frequently the itemsets appear in the dataset.

   Support(A→B)=Number of transactions containing A∪BTotal number of transactions\text{Support}(A \rightarrow B) = \frac{\text{Number of transactions containing } A \cup B}{\text{Total number of transactions}}Support(A→B)=Total number of transactionsNumber of transactions containing A∪B

2. **Confidence**: This measures how often items in BBB appear in transactions that contain AAA.

   Confidence(A→B)=Number of transactions containing A∪BNumber of transactions containing A\text{Confidence}(A \rightarrow B) = \frac{\text{Number of transactions containing } A \cup B}{\text{Number of transactions containing } A}Confidence(A→B)=Number of transactions containing ANumber of transactions containing A∪B

3. **Lift**: This measures the strength of the rule over the random co-occurrence of AAA and BBB. A lift value greater than 1 indicates a positive correlation between AAA and BBB.

   Lift(A→B)=Confidence(A→B)Support(B)\text{Lift}(A \rightarrow B) = \frac{\text{Confidence}(A \rightarrow B)}{\text{Support}(B)}Lift(A→B)=Support(B)Confidence(A→B)

Example

Consider a retail store with the following transaction data:

Transaction ID	Items
1	Bread, Milk
2	Bread, Diapers, Beer

Transaction ID	Items
3	Milk, Diapers, Beer, Cola
4	Bread, Milk, Diapers, Beer
5	Bread, Milk, Cola

An example of an association rule from this data could be: Bread→Milk\text{Bread} \rightarrow \text{Milk}Bread→Milk

- **Support**: This rule appears in 3 out of 5 transactions, so the support is 35=0.6\frac{3}{5} = 0.653=0.6.
- **Confidence**: Out of the 4 transactions that contain Bread, 3 also contain Milk, so the confidence is 34=0.75\frac{3}{4} = 0.7543=0.75.
- **Lift**: If Milk appears in 4 out of 5 transactions, the support of Milk is 0.80.80.8. The lift is 0.750.8=0.9375\frac{0.75}{0.8} = 0.93750.80.75=0.9375.

Applications

Association rules are used in various domains, including:

- **Market Basket Analysis**: Identifying product bundles to improve cross-selling strategies.
- **Fraud Detection**: Finding patterns that indicate fraudulent transactions.
- **Medical Diagnosis**: Discovering relationships between symptoms and diseases.
- **Web Usage Mining**: Understanding user navigation patterns on websites.

FP-Growth Algo

FP-Growth (Frequent Pattern Growth) is a popular algorithm for mining frequent itemsets and association rules in unsupervised learning. Here's a detailed explanation, including its purpose, suitable data types, examples, and a practical implementation with visualization.

Purpose of FP-Growth

FP-Growth is used for:

1. **Finding Frequent Itemsets:** It identifies itemsets that appear frequently together in a dataset.
2. **Generating Association Rules:** These rules describe how the occurrence of one itemset influences the occurrence of another.

Suitable Data Types

FP-Growth is most suitable for transactional data, where each transaction is a set of items. Examples of such data include:

- Market Basket Analysis (e.g., retail transactions)
- Web Clickstream Analysis
- Medical Diagnosis (e.g., symptoms leading to a disease)
- Text Mining (e.g., word co-occurrences in documents)

Examples

1. **Market Basket Analysis:** Identifying products often bought together, like bread and butter.
2. **Web Usage Mining:** Finding patterns in web page visits, such as users who visit the homepage often also visit the product page.
3. **Medical Diagnosis:** Determining which symptoms frequently co-occur in patients.

Implementation and Visualization

We'll use a real-life dataset, the "Groceries" dataset, to demonstrate the FP-Growth algorithm. This dataset contains transactions of items purchased by customers.

Step 1: Load the Dataset

First, we'll load and visualize the dataset.

```
import pandas as pd


# Load the Groceries dataset

url = 'https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/groceries.csv'

groceries = pd.read_csv(url)

groceries.head()
```

Step 2: Preprocess the Data

We'll preprocess the data to fit the format required for FP-Growth.

```
# Convert the dataset into a list of transactions

transactions = groceries.values.tolist()
```

Step 3: Apply FP-Growth

We'll use the `mlxtend` library for the FP-Growth algorithm.

```
from mlxtend.preprocessing import TransactionEncoder
```

```
from mlxtend.frequent_patterns import fpgrowth, association_rules


# Transform the transactions into a one-hot encoded DataFrame
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_ary, columns=te.columns_)


# Apply the FP-Growth algorithm
frequent_itemsets = fpgrowth(df, min_support=0.01, use_colnames=True)
frequent_itemsets.head()
```

Step 4: Generate Association Rules

We can now generate association rules from the frequent itemsets.

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

Step 5: Visualization

Finally, we'll visualize the frequent itemsets and association rules.

```
import matplotlib.pyplot as plt
import networkx as nx


# Visualize the frequent itemsets
plt.bar(frequent_itemsets['itemsets'].astype(str), frequent_itemsets['support'])
plt.xticks(rotation=90)
plt.ylabel('Support')
plt.xlabel('Itemsets')
plt.title('Frequent Itemsets')
plt.show()
```

Visualize the association rules

G = nx.from_pandas_edgelist(rules, 'antecedents', 'consequents', edge_attr=True)

plt.figure(figsize=(12, 8))

pos = nx.spring_layout(G)

nx.draw(G, pos, with_labels=True, node_size=3000, node_color="skyblue", font_size=12, font_color="black", font_weight="bold")

edge_labels = nx.get_edge_attributes(G, 'lift')

nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)

plt.title('Association Rules')

plt.show()

Code Explanation

- **import pandas as pd**: Imports the pandas library, which is used for data manipulation and analysis.
- **url = 'https://raw.githubusercontent.com/stedy/Machine-Learning-with-R-datasets/master/groceries.csv'**: Stores the URL of the dataset in a variable.
- **groceries = pd.read_csv(url)**: Reads the CSV file from the URL and loads it into a pandas DataFrame named `groceries`.
- **groceries.head()**: Displays the first few rows of the DataFrame to get an overview of the data.

transactions = groceries.values.tolist(): Converts the DataFrame into a list of lists (each inner list represents a transaction).

- **from mlxtend.preprocessing import TransactionEncoder**: Imports the TransactionEncoder class from the mlxtend library, which helps in encoding the transactions.
- **from mlxtend.frequent_patterns import fpgrowth, association_rules**: Imports the fpgrowth function for finding frequent itemsets and the association_rules function for generating association rules.
- **te = TransactionEncoder()**: Creates an instance of the TransactionEncoder.
- **te_ary = te.fit(transactions).transform(transactions)**: Fits the TransactionEncoder to the data and transforms the transactions into a one-hot encoded NumPy array.
- **df = pd.DataFrame(te_ary, columns=te.columns_)**: Converts the one-hot encoded array into a pandas DataFrame.
- **frequent_itemsets = fpgrowth(df, min_support=0.01, use_colnames=True)**: Applies the FP-Growth algorithm to find frequent itemsets with a minimum support of 0.01 and uses column names.
- **frequent_itemsets.head()**: Displays the first few frequent itemsets.

- **rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)**: Generates association rules from the frequent itemsets using the lift metric and a minimum threshold of 1.
- **rules.head()**: Displays the first few association rules.

1. **import matplotlib.pyplot as plt**: Imports the matplotlib library for plotting graphs.
2. **import networkx as nx**: Imports the networkx library for creating and manipulating complex networks.

Visualizing Frequent Itemsets

3. **plt.bar(frequent_itemsets['itemsets'].astype(str), frequent_itemsets['support'])**: Creates a bar chart of the frequent itemsets and their support values.
4. **plt.xticks(rotation=90)**: Rotates the x-axis labels by 90 degrees for better readability.
5. **plt.ylabel('Support')**: Sets the label for the y-axis as 'Support'.
6. **plt.xlabel('Itemsets')**: Sets the label for the x-axis as 'Itemsets'.
7. **plt.title('Frequent Itemsets')**: Sets the title of the chart as 'Frequent Itemsets'.
8. **plt.show()**: Displays the bar chart.

Visualizing Association Rules

9. **G = nx.from_pandas_edgelist(rules, 'antecedents', 'consequents', edge_attr=True)**: Creates a network graph from the association rules, where antecedents are connected to consequents.
10. **plt.figure(figsize=(12, 8))**: Sets the figure size for the plot.
11. **pos = nx.spring_layout(G)**: Computes the layout for the graph using a spring layout.
12. **nx.draw(G, pos, with_labels=True, node_size=3000, node_color="skyblue", font_size=12, font_color="black", font_weight="bold")**: Draws the graph with nodes, labels, and specified aesthetics.
13. **edge_labels = nx.get_edge_attributes(G, 'lift')**: Retrieves the edge attributes for the 'lift' metric.
14. **nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)**: Draws the edge labels (lift values) on the graph.
15. **plt.title('Association Rules')**: Sets the title of the graph as 'Association Rules'.
16. **plt.show()**: Displays the network graph.