Artificial Intelligence and Data Analytics for Engineers
Exercise 7

Introduction

Solving exercises is not mandatory, and exercises will not be graded or corrected by the lecture team. However, we strongly advise you to do as many exercises as possible to prepare for the final exam. Some of the tasks will be discussed with a presentation of the solutions during the exercise session on Thursday. If you want to do exercises at home, we suggest installing Anaconda (https://www.anaconda.com/distribution/ - you will want to download the Python 3.7 version).

In case you have any questions, feel free to send us an e-Mail to: aidae@ima-ifu.rwth-aachen.de.

Task 0) Setting up your Python environment

Since you created a Python environment the first week, we only need to enable the environment from now on. This needs to be done **every time** you open a new terminal, for instance after a system restart or if you accidentally closed the Terminal. Open a terminal and enter:

```
source activate aidae
```

Now you are ready to start programming in Python. Remember to ensure that spyder is installed in your environment (`conda install spyder`) and launched from the terminal where you activated the environment.

Task 1) Association Rule Mining

In this task you will use the Apriori Algorithm to identify rules describing associations between product purchases from a retail store. For the task you are given a dataset which can be downloaded from https://drive.google.com/file/d/1y5DYn0dGoSbC22xowBq2d4po6h1JxcTQ/view?usp=sharing. The dataset contains 7500 transactions from the retail store and lists line-wise purchases (maximum 20 items) as CSV, e.g. `[‘vegetables’, ‘spaghetti’, ‘green tea’]`. For the task we will use the following library: https://pypi.org/project/apyori/

a) Import the library.
b) Import the dataset using the appropriate pandas `read_csv` function and explore it.

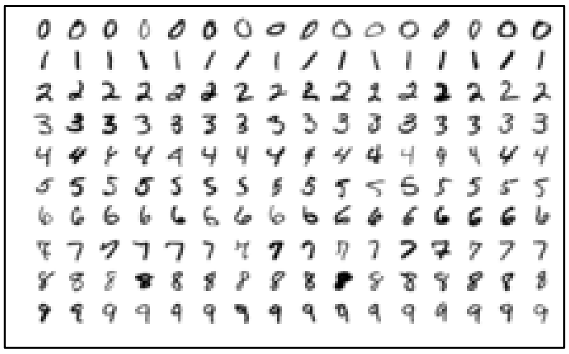
c) The algorithm requires the dataset to be in a specific format, so we'll have to do some pre-processing of it: You have to transform the data into a list (outer list) of lists (inner list). Each inner list contains a transaction. Hint: Use a for loop to iterate over the dataset.
d) Apply the Apriori algorithm and store the result into a variable, e.g. `association_rules`. It requires certain parameters:
   - `input` (i.e. your list of lists)
   - `min_support`
   - `min_confidence`
   - `min_lift`
   - `min_length`

   Use the following values: `min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2`

e) Convert the rules into a Python list using the following method `list(association_rules)`. Again, store the result, e.g. in `association_results`.
f) Explore the result list from the previous task
g) Implement the following snippet and interpret the result

```
for item in association_rules:

    # first index of the inner list

    # Contains base item and add item

    pair = item[0]

    items = [x for x in pair]

    print("Rule: " + items[0] + " -> " + items[1])

    #second index of the inner list

    print("Support: " + str(item[1]))

    #third index of the list located at 0th

    #of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))

    print("Lift: " + str(item[2][0][3]))

    print("=====================================")
```

Task 2) Unsupervised Learning for Dimensionality Reduction



In this task, you will use unsupervised learning to implement a dimensionality reduction from one of our previous lectures (“PCA”). For this task we’ll use the MNIST database of handwritten digits. It has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. Each image is 28x28 pixels, thus, each observation has 784 dimensions.

a) Import the dataset “`MNIST original`” from the datasets potion of sklearn (`sklearn.datasets.fetch_mldata`)

b) Split the data using the `train_test_split` method
```
X, y = mnist.data / 255., mnist.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```
What did the previous commands achieve?

c) Import the following libraries (which will be used for visualization):
```
from time import time
import numpy as np
from sklearn import manifold
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from matplotlib import offsetbox
```

d) Prepare the visualization and try to explain the purpose of the following lines:
```
def plot_embedding(X,y, title=None):
    x_min, x_max = np.min(X, 0), np.max(X, 0)
    X = (X - x_min) / (x_max - x_min)
    plt.figure()
    ax = plt.subplot(111)
    for i in range(X.shape[0]):
        plt.text(X[i, 0], X[i, 1], str(y[i]),
                 color=plt.cm.Set1(y[i] / 10.),
                 fontdict={'weight': 'bold', 'size': 9})
    plt.xticks([]), plt.yticks([])
    if title is not None:
        plt.title(title)
indexes=np.random.choice(len(X_train),size=300)
```

e) Train an embedding model (PCA, umap) and plot the result. Try to explain the purpose of each code line and the resulting visualization.