Case-based Reasoning System

Yefferson A. Marín Cantero
Artificial intelligence course
Systems Engineering Program
Universidad Tecnológica de Bolívar
Cartagena de Indias, D.T. y C - Bolívar
1p-2020

Definitions

A **Case-based reasoning** (CBR) is a paradigm of artificial intelligence and cognitive science that models the reasoning process as primarily memory based. Case-based reasoning systems solve new problems by retrieving stored 'cases' describing similar prior problem-solving episodes and adapting their solutions to fit new needs.

Case-based reasoning has been formalized (https://www.idi.ntnu.no/emner/tdt4171/papers/AamodtPlaza94.pdf) for purposes of computer reasoning as a four-step process:

- **1. Retrieve** Given a target problem, retrieve from memory cases relevant to solving it. A case consists of a problem, its solution, and, typically, annotations about how the solution was derived.
- **2. Reuse:** Map the solution from the previous case to the target problem. This may involve adapting the solution as needed to fit the new situation.
- **3. Revise:** Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise.
- **4. Retain:** After the solution has been successfully adapted to the target problem, store the resulting experience as a new case in memory.

Introduction

This algorithm is done using the following requirements:

- Python 3.* (https://python.org)
- Pandas (https://pandas.pydata.org/)
- Numpy (https://numpy.org)
- Scipy spatial distance (https://docs.scipy.org/doc/scipy/reference/spatial.distance.html#module-scipy.spatial.distance)
- Matplotlib (https://matplotlib.org)
- Seaborn (https://docs.scipy.org/doc/scipy/reference/spatial.distance.html#module-scipy.spatial.distance)

And its taking a **library cases** stored in input/library.csv (input/library.csv) to get the Case-based reasoning from test **problem cases** stored in input/cases.csv (input/cases.csv).

The purpose is designing a system that fits the test cases into the base library cases in order to find the most appropriate solution.

Steps

1. Retrieve

Library and problem cases

First it's required to retrieve the base cases or the library of relevant cases with their data (used to compute) and its solutions. Also we need to retrieve our test problem cases.

2. Reuse

One-hot encoding

Machine learning algorithms cannot work with categorical data directly, this must be converted to numbers. This technique is called one-hot encoding and is a representation of categorical variables as binary vectors.

We must transform our library and problem cases by firstly requiring that the categorical values be mapped to integer values, and then, representing each integer value as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

Mahalanobis distance

Having mapped both (library and problem cases), we should define a similarity comparison method, the best match is the Mahalanobis distance which is an effective multivariate distance metric that measures the distance between a point (P) and a distribution (D). It is an extremely useful metric having, excellent applications in multivariate anomaly detection, classification on highly imbalanced datasets and one-class classification.

Mahalanobis distance is widely used in cluster analysis and classification techniques, as a multi-dimensional generalization of the idea of measuring how many standard deviations away P is from the mean of D.

$$D(\vec{u}, \vec{v}) = \sqrt{(\vec{u} - \vec{u})V^{-1}(\vec{u} - \vec{v})^T}$$

Where $\vec{u}$ and $\vec{v}$ are arrays, and V^{-1} The inverse of the covariance matrix.

Covariance matrix

A covariance matrix (also known as auto-covariance matrix, dispersion matrix, variance matrix, or variance–covariance matrix) is a square matrix giving the covariance between each pair of elements of a given random vector. In the matrix diagonal there are variances, i.e., the covariance of each element with itself. Intuitively, the covariance matrix generalizes the notion of variance to multiple dimensions.

3. Revise

After compare the shortest distances, we get the solution based on proximity calculated (similarity) with library cases.

4. Retain

Implementation

Initially we import the needed libraries, as it follows:

In [1]:
```
import numpy as np
import pandas as pd
from scipy.spatial import distance
import matplotlib.pyplot as plt
import seaborn as sn
```

Library and problem cases

The we get our **library cases** stored in [input/library.csv (input/library.csv)](input/library.csv) and the test **problem cases** stored in [input/cases.csv (input/cases.csv)](input/cases.csv).

In [2]:
```
# Get the input .csv library and problem cases
# {pandas.DataFrame}
library, cases = pd.read_csv('input/library.csv'), pd.read_csv('input/cases.csv')
```

In [3]:
```
library
```

Out[3]:

	Outlook	Temperature	Humidity	Windy	Play
0	Sunny	Hot	High	False	No
1	Sunny	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Rainy	Mild	High	False	Yes
4	Rainy	Cool	Normal	False	Yes
5	Rainy	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Sunny	Mild	High	False	No
8	Sunny	Cool	Normal	False	Yes
9	Rainy	Mild	Normal	False	Yes
10	Sunny	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Rainy	Mild	High	True	No

In [4]:
```
cases
```

Out[4]:

	Outlook	Temperature	Humidity	Windy
0	Sunny	Mild	Normal	False
1	Rainy	Cool	Normal	False
2	Overcast	Cool	High	False
3	Sunny	Cool	High	True
4	Rainy	Hot	High	True
5	Rainy	Cool	High	True

At this point, we can verify which kind of data its represented:

In [5]:
```
library.dtypes
```

Out[5]:
```
Outlook          object
 Temperature     object
 Humidity        object
 Windy           object
 Play            object
dtype: object
```

In [6]:
```
cases.dtypes
```

Out[6]:
```
Outlook          object
 Temperature     object
 Humidity        object
 Windy           object
dtype: object
```

Base & Initial one-hot encoding

As we verified, our data is categorical, so we are going to convert them using one-hot encoding method:

In [7]:
```
# Select columns from library to use as base cases, except solutions
base = library.iloc[:, range(library.shape[1] - 1)]      # Exclude last column

# Initial One-hot encoding
base = pd.get_dummies(base)
problems = pd.get_dummies(cases)
```

Our initial library cases (base) and case/problems to evaluate, with this technique will lock as it follows:

In [8]:
```
base
```

Out[8]:

	Outlook_Overcast	Outlook_Rainy	Outlook_Sunny	Temperature_Cool	Temperature_Hot	Temperature_Mild	Humidity_High	Humidity_Normal	Windy_False	Windy_True
0	0	0	1	0	1	0	1	0	1	0
1	0	0	1	0	1	0	1	0	0	1
2	1	0	0	0	1	0	1	0	1	0
3	0	1	0	0	0	1	1	0	1	0
4	0	1	0	1	0	0	0	1	1	0
5	0	1	0	1	0	0	0	1	0	1
6	1	0	0	1	0	0	0	1	0	1
7	0	0	1	0	0	1	1	0	1	0
8	0	0	1	1	0	0	0	1	1	0
9	0	1	0	0	0	1	0	1	1	0
10	0	0	1	0	0	1	0	1	0	1
11	1	0	0	0	0	1	1	0	0	1
12	1	0	0	0	1	0	0	1	1	0
13	0	1	0	0	0	1	1	0	0	1

In [9]:
```
problems
```

Out[9]:

	Outlook_Overcast	Outlook_Rainy	Outlook_Sunny	Temperature_Cool	Temperature_Hot	Temperature_Mild	Humidity_High	Humidity_Normal	Windy_False	Windy_True
0	0	0	1	0	0	1	0	1	1	0
1	0	1	0	1	0	0	0	1	1	0
2	1	0	0	1	0	0	1	0	1	0
3	0	0	1	1	0	0	1	0	0	1
4	0	1	0	0	1	0	1	0	0	1
5	0	1	0	1	0	0	1	0	0	1

Calculate

Our main code can be divided in the following steps:

1. Calculate *inverse covariance matrix* for the *base* cases.
2. Get the *case* to evaluate.
3. Calculate *mahalanobis distance* using *case*, *base* and *inverse covariance matrix*.
4. *Minimum distances calculated* will be stored.
5. *Minimum distance calculated index* will be used to solve the problem, by using the *index* solution in *base* cases.
6. Append solution to the *library*, to use it in future cases, and store other relevant data (eg. *covariance heat maps*).
7. If there are more cases, it evaluates, getting the new *base* (one-hot) encoded.

In [10]:
```
# Move through all problem cases
for i in range(problems.shape[0]):
    # Get inverse covariance matrix for the base cases
    covariance_matrix = base.cov()                                     # Covariance
    inverse_covariance_matrix = np.linalg.pinv(covariance_matrix)      # Inverse

    # Get case row to evaluate
    case_row = problems.loc[i, :]

    # Empty distances array to store mahalanobis distances obtained comparing each library cases
    distances = np.zeros(base.shape[0])

    # For each base cases rows
    for j in range(base.shape[0]):
        # Get base case row
        base_row = base.loc[j, :]

        # Calculate mahalanobis distance between case row and base cases, and store it
        distances[j] = distance.mahalanobis(case_row, base_row, inverse_covariance_matrix)

    # Returns the index (row) of the minimum value in distances calculated
    min_distance_row = np.argmin(distances)

    # Get solution based on index of found minimum distance, and append it to main library
    # From cases, append library 'similar' solution
    case = np.append(cases.iloc[i, :], library.iloc[min_distance_row, -1])

    # Print
    print(f'> For case/problem {i}: {cases.iloc[i, :].to_numpy()}, solution is {case[-1]}')

    # Store
    # Get as operable pandas Series
    case = pd.Series(case, index = library.columns)          # Case with Solution
    library = library.append(case, ignore_index = True)      # Append to library

    # Save 'covariance heat map (biased)' output as file
    sn.heatmap(np.cov(base, bias = True), annot = True, fmt = 'g')
    plt.gcf().set_size_inches(12, 6)
    plt.title(f'Covariance Heat map #{i} \n Library cases stored {j} - Base to solve problem {i}')
    plt.savefig(f'output/covariance_heat_map_{i}.png', bbox_inches='tight')
    plt.close()

    # Reuse
    base = library.iloc[:, range(library.shape[1] - 1)]      # Exclude last column (solution)
    base = pd.get_dummies(base)                              # Get new one-hot encoded base

# Save 'library' output as file
library.to_csv('output/library.csv', index = False)
```

```
> For case/problem 0: ['Sunny' ' Mild' ' Normal' ' False'], solution is  Yes
> For case/problem 1: ['Rainy' ' Cool' ' Normal' ' False'], solution is  Yes
> For case/problem 2: ['Overcast' ' Cool' ' High' ' False'], solution is  Yes
> For case/problem 3: ['Sunny' ' Cool' ' High' ' True'], solution is  Yes
> For case/problem 4: ['Rainy' ' Hot' ' High' ' True'], solution is  No
> For case/problem 5: ['Rainy' ' Cool' ' High' ' True'], solution is  No
```

Results

Finally, we can output our library plus cases/problems solved.

In [11]:
```
library
```

Out[11]:

	Outlook	Temperature	Humidity	Windy	Play
0	Sunny	Hot	High	False	No
1	Sunny	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Rainy	Mild	High	False	Yes
4	Rainy	Cool	Normal	False	Yes
5	Rainy	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Sunny	Mild	High	False	No
8	Sunny	Cool	Normal	False	Yes
9	Rainy	Mild	Normal	False	Yes
10	Sunny	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Rainy	Mild	High	True	No
14	Sunny	Mild	Normal	False	Yes
15	Rainy	Cool	Normal	False	Yes
16	Overcast	Cool	High	False	Yes
17	Sunny	Cool	High	True	Yes
18	Rainy	Hot	High	True	No
19	Rainy	Cool	High	True	No

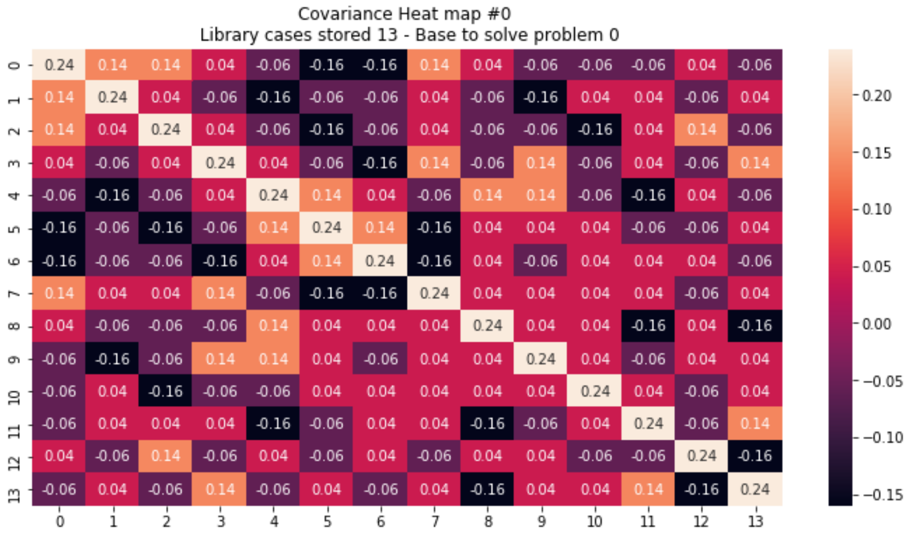
Generated files

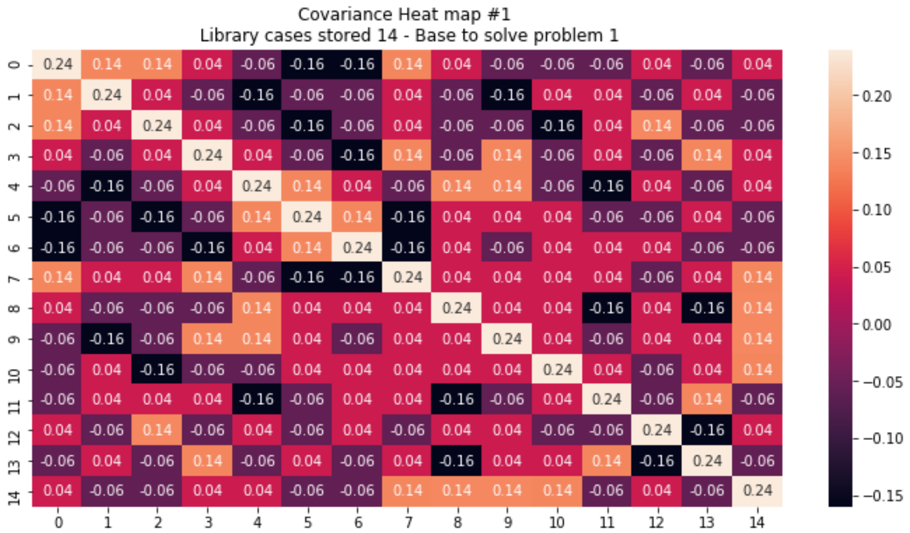
Library

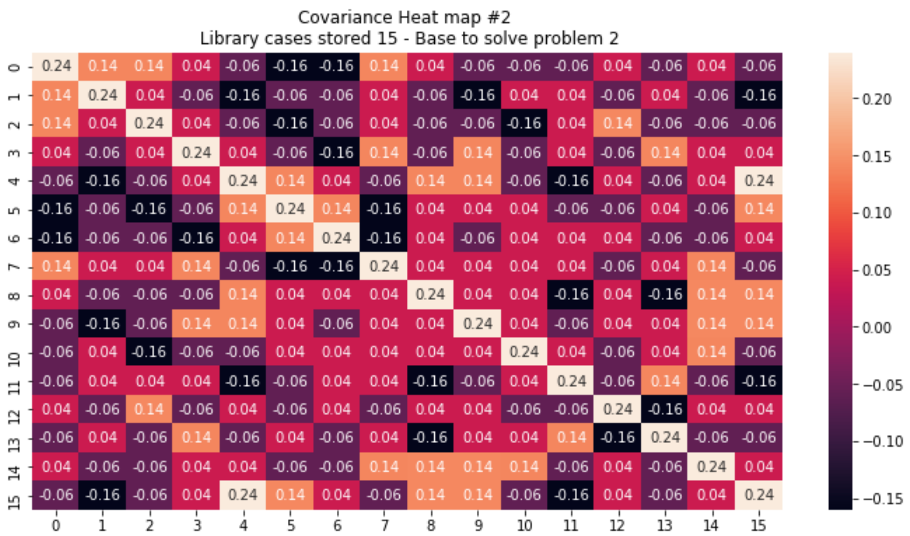
Initial library plus cases/problems solved, will be available at [output/library.csv (output/library.csv)](output/library.csv).

Heat maps

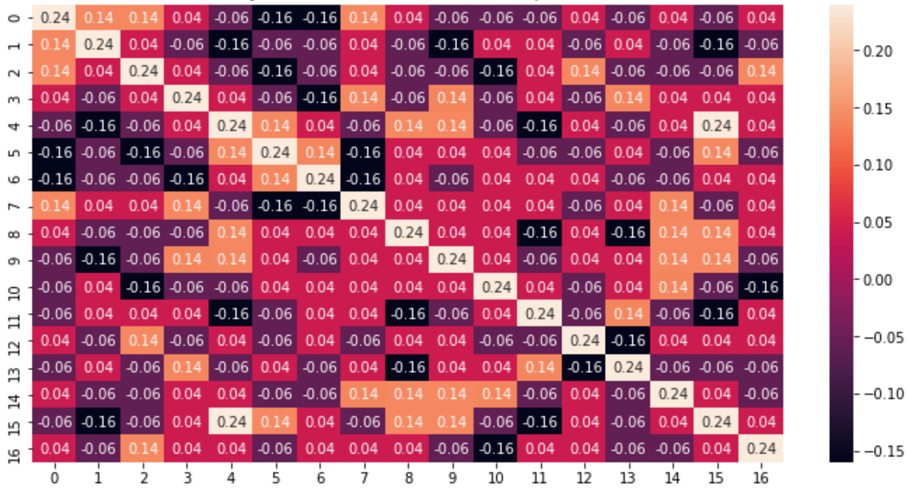
An image for each Covariance Heat map, using library cases stored (base) at each iteration to solve a specific problem, will be available at [output/heatmap_x.png (output)](output)



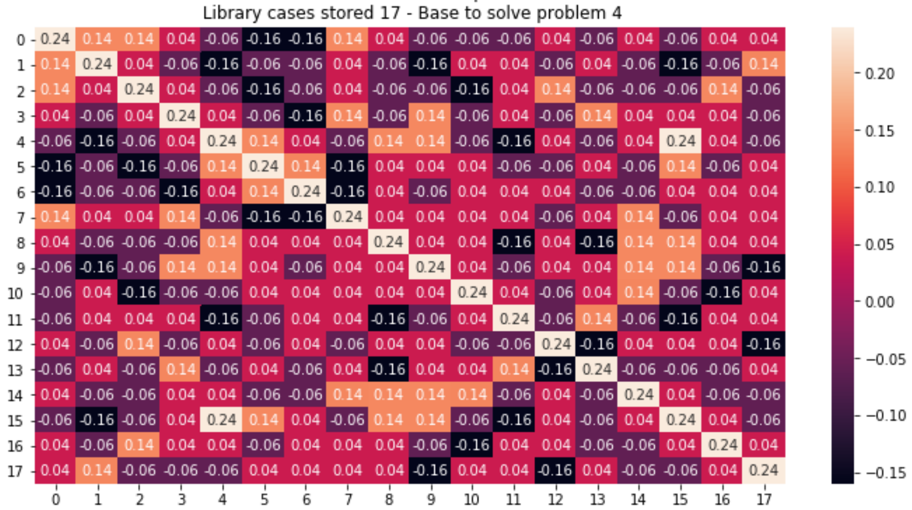




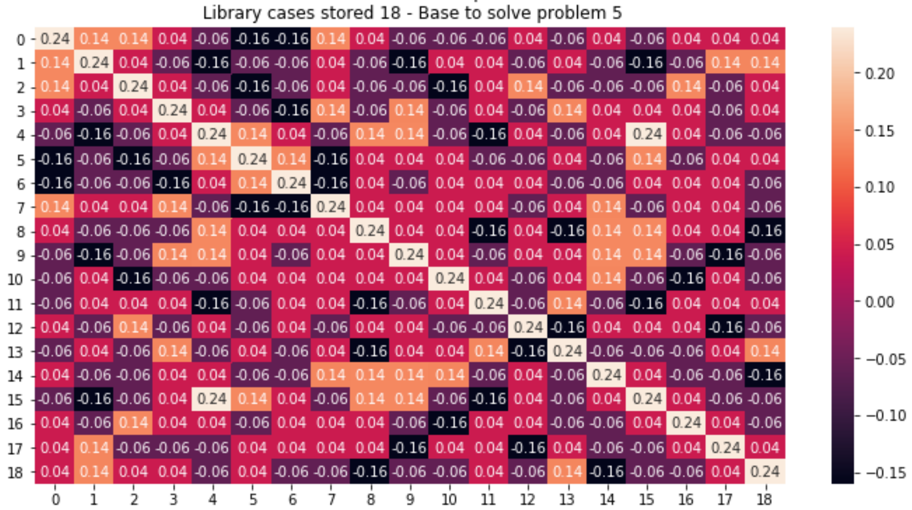
Covariance Heat map #3
Library cases stored 16 - Base to solve problem 3



Covariance Heat map #4
Library cases stored 17 - Base to solve problem 4



Covariance Heat map #5
Library cases stored 18 - Base to solve problem 5



Source code

Repository

All code has been deployed at *Github* and its available at [yammadev/cbrs (https://github.com/yammadev/cbrs)](https://github.com/yammadev/cbrs).

Testing

Executable using `py cbrs.py` will output (and also generate files):

```
> Initial Library

     Outlook  Temperature  Humidity   Windy  Play
0      Sunny          Hot      High   False    No
1      Sunny          Hot      High    True    No
2   Overcast          Hot      High   False   Yes
3      Rainy         Mild      High   False   Yes
4      Rainy         Cool    Normal   False   Yes
5      Rainy         Cool    Normal    True    No
6   Overcast         Cool    Normal    True   Yes
7      Sunny         Mild      High   False    No
8      Sunny         Cool    Normal   False   Yes
9      Rainy         Mild    Normal   False   Yes
10     Sunny         Mild    Normal    True   Yes
11  Overcast         Mild      High    True   Yes
12  Overcast          Hot    Normal   False   Yes
13     Rainy         Mild      High    True    No

> Calculating

> For case/problem 0: ['Sunny' ' Mild' ' Normal' ' False'], solution is  Yes
> For case/problem 1: ['Rainy' ' Cool' ' Normal' ' False'], solution is  Yes
> For case/problem 2: ['Overcast' ' Cool' ' High' ' False'], solution is  Yes
> For case/problem 3: ['Sunny' ' Cool' ' High' ' True'], solution is  Yes
> For case/problem 4: ['Rainy' ' Hot' ' High' ' True'], solution is  No
> For case/problem 5: ['Rainy' ' Cool' ' High' ' True'], solution is  No

> Output library

     Outlook  Temperature  Humidity   Windy  Play
0      Sunny          Hot      High   False    No
1      Sunny          Hot      High    True    No
2   Overcast          Hot      High   False   Yes
3      Rainy         Mild      High   False   Yes
4      Rainy         Cool    Normal   False   Yes
5      Rainy         Cool    Normal    True    No
6   Overcast         Cool    Normal    True   Yes
7      Sunny         Mild      High   False    No
8      Sunny         Cool    Normal   False   Yes
9      Rainy         Mild    Normal   False   Yes
10     Sunny         Mild    Normal    True   Yes
11  Overcast         Mild      High    True   Yes
12  Overcast          Hot    Normal   False   Yes
13     Rainy         Mild      High    True    No
14     Sunny         Mild    Normal   False   Yes
15     Rainy         Cool    Normal   False   Yes
16  Overcast         Cool      High   False   Yes
17     Sunny         Cool      High    True   Yes
18     Rainy          Hot      High    True    No
19     Rainy         Cool      High    True    No
```