G. Montavon
Institute of Computer Science
Department of Mathematics and Computer Science
Freie Universität Berlin

Exercise Sheet 1

In the following, we explore different ways of accessing data, including reading CSV files, querying databases, and applying preprocessing and plotting techniques to the available data. The cell below imports some libraries that are required to complete the tasks. Note that you need to install additional python libraries such as `cv2`, `torch`, `torchvision`, `matolotlib` and `sqlite3`. Some of these libraries will also be needed for the next exercise sheets.

In [1]:
```
%matplotlib inline
import numpy,scipy,scipy.spatial
import torch
import torchvision,torchvision.transforms
import sqlite3
import matplotlib
from matplotlib import pyplot as plt
from PIL import Image
```

Exercise 1: Loading CSV Data (15+15 P)

In this exercise, we investigate the usage of the function `numpy.genfromtxt` to load several datasets from the UCI repository. These datasets are provided in the form of csv files in the folder `csvdata` of the homework.

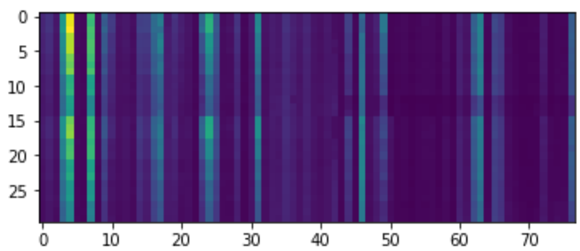
(a) Using the method `numpy.genfromtxt`, load the dataset contained in the file `Wholesale customers data.csv`. In this dataset, instances (rows) are retailers, and features (columns) represent how much these retailers spend for different categories of products. Once the dataset is loaded, compute the average and median spending (over instances) for each category of products.

In [2]:
```
# ------------------------------------
# TODO: Replace by your code
# ------------------------------------
import solutions
solutions.task1a()
# ------------------------------------
```

```
Fresh                 12000.298   8504.000
Milk                   5796.266   3627.000
Grocery                7951.277   4755.500
Frozen                 3071.932   1526.000
Detergents_Paper       2881.493    816.500
Delicatessen           1524.870    965.500
```

(b) Using the method `numpy.genfromtxt`, load the dataset contained in the file `CortexNuclear.csv`, and use the library `matplotlib` to produce an image plot that visualizes the dataset, specifically visualize the first 30 instances that do not contain any missing value.

In [3]:
```
# ------------------------------------
# TODO: Replace by your code
# ------------------------------------
import solutions
solutions.task1b()
# ------------------------------------
```



Exercise 2: Querying a Database (20+20 P)

In the following, we will use the sqlite3 package to connect to a database, and perform various join operations. The sqlite3 package enables you to connect to a database and to perform various queries. We will consider the chinook database, which simulates data from a music store, relating music tracks, artists, invoices, customers, etc. Connect to the database. The database can also be downloaded from the link https://www.sqlitetutorial.net/sqlite-sample-database/ (https://www.sqlitetutorial.net/sqlite-sample-database/) .

In [4]:
```
db = sqlite3.connect('chinook.db')
```

The database has the following schema

We first consider a simple query on this database. The query is formulated in the SQL language and retrieves the duration of tracks found in that database. Once the results of the query have been obtained, we perform a very basic data analysis: computing the mean track duration.

In [5]:
```
cursor = db.cursor()
query = "SELECT Milliseconds FROM tracks;"
results = numpy.array(cursor.execute(query).fetchall())
mean = results.mean()/1000.0
print(f"{'Average track duration':25s} {mean:8.3f}")
```

```
Average track duration     393.599
```

Now, we would like to perform more complex SQL queries. For a tutorial on SQL, see for example, https://www.sqltutorial.org/ (https://www.sqltutorial.org/). In particular, look at Section 6 which discusses the SQL operation "INNER JOIN" and that is useful for generating outputs involving multiple tables.

(a) Apply a SQL query that extracts a table containing for all tracks their genre and their track length. Then, write code that computes for each genre (sorted alphabetically the average track length).

In [6]:
```
# ----------------------------------
# TODO: Replace by your code
# ----------------------------------
import solutions
solutions.task2a(db)
# ----------------------------------
```

```
Alternative          264.059
Alternative & Punk   234.354
Blues                270.360
Bossa Nova           219.590
Classical            293.868
Comedy              1585.264
Drama               2575.284
Easy Listening       189.164
Electronica/Dance    302.986
Heavy Metal          297.453
Hip Hop/Rap          178.176
Jazz                 291.755
Latin                232.859
Metal                309.749
Opera                174.813
Pop                  229.034
R&B/Soul             220.067
Reggae               247.178
Rock                 283.910
Rock And Roll        134.644
Sci Fi & Fantasy    2911.783
Science Fiction     2625.549
Soundtrack           244.371
TV Shows            2145.041
World                224.924
```

We would like to analyze the preference for music genres in different countries.

(b) Apply a SQL query that extracts for each invoice the country of the customer and the genre of the track the customer has purchased. Then, print in the form of a table the number of purchases for each country and genre.

In [7]:
```python
# ------------------------------------
# TODO: Replace by your code
# ------------------------------------
import solutions
solutions.task2b(db)
# ------------------------------------
```

```
               ArgAusAusBelBraCanChiCzeDenFinFraGerHunIndIreItaNetNorPolPorSpaSweUSAUni
Alternative      0  0  0  0  0  0  0  0  0  0  4  1  0  0  0  0  0  4  0  0  0  0  5  0
Alternative & P  9  0  0 14  7 36  2  9  4  2 31 13  3 11  4 11  5  2  7  5  4  6 50  9
Blues            0  1  0  0  6  4  2  1  0  0  2 14  2  3  1  4  4  0  1  1  0  0 15  0
Bossa Nova       0  0  0  0  0  7  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  7  0
Classical        0  0  2  0  6  5  1  0  0  0 10  0  0  2  0  2  0  5  0  0  0  0  8  0
Comedy           0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  8  0
Drama            0  0  1  0  0  2  4  6  0  0  4  1  2  0  1  0  0  2  0  0  0  0  6  0
Easy Listening   2  0  0  0  0  0  3  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  3  0
Electronica/Dan  0  0  0  0  0  4  0  2  0  0  2  0  1  0  0  0  0  2  0  1  0  0  0  0
Heavy Metal      0  3  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  2  0  0  4  0
Hip Hop/Rap      0  0  0  0  2  5  0  0  0  0  2  0  0  0  0  0  0  0  0  0  1  0  4  3
Jazz             2  0  2  0  0 13  0  3  0  2 11  2  0 10  3  0  0  0  1  2  2  1 22  4
Latin            8  2  2  0 53 60  8  9  4  9 26 18  3  9  7  2  3  5  7 13  4 12 91 31
Metal            7  8  7  1 15 40  3  6  6  2 20 25  5  8  3  1  2  0  0 11  3  7 64 20
Pop              0  0  1  0  3  0  0  4  3  0  2  1  0  0  0  0  3  0  0  4  0  0  5  2
R&B/Soul         0  0  4  2  3  5  0  2  0  0  0  0  5  4  0  0  0  0  0  2  0  0 12  2
Reggae           0  2  0  0  6  7  0  0  0  1  1  0  0  0  0  0  0  0  0  0  2  0  6  5
Rock             9 22 15 21 81107  9 25 21 18 65 62 11 25 12 18 18 17 22 31 22 10157 37
Rock And Roll    0  0  0  0  0  2  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  3  0
Sci Fi & Fantas  0  0  0  0  2  0  1  0  0  4  2  0  1  1  0  0  3  0  0  1  0  0  5  0
Science Fiction  0  0  0  0  0  0  2  1  0  0  0  2  0  0  0  0  0  0  0  0  0  0  1  0
Soundtrack       1  0  0  0  4  0  1  0  0  0  5  5  0  0  0  0  0  0  0  0  0  0  4  0
TV Shows         0  0  4  0  0  1  2  8  0  0  1  3  4  1  7  0  0  0  0  1  0  1 14  0
World            0  0  0  0  2  6  0  0  0  0  0  0  0  0  0  0  0  1  0  2  0  1  0  1
```

Exercise 3: Representing Images (15+15 P)

Images are high-dimensional data. High-level concepts contained in these images are hard to extract directly from the raw pixel representation. In the following, we investigate the benefit of preprocessing the data with a neural network. Specifically, we want to see if representations built by the neural network enable to produce meaningful similarities or dissimilarities between images. We consider for this exercise a set ot 10 images. The first 5 images depict geraniums, and the last 5 images depict ferrari cars.

In [8]:
```python
images = [Image.open(f'imagedata/{i}.jpg') for i in range(10)]
```

Although the two groups are clearly distinct from a human point of view, we will show that distances computed on pixel values (i.e. treating an image as vector storing the multiple RGB pixel values) does do not enable such distinction.

(a) Compute a matrix of pairwise Euclidean distances between images (images are resized to 100 x 100 for this task).

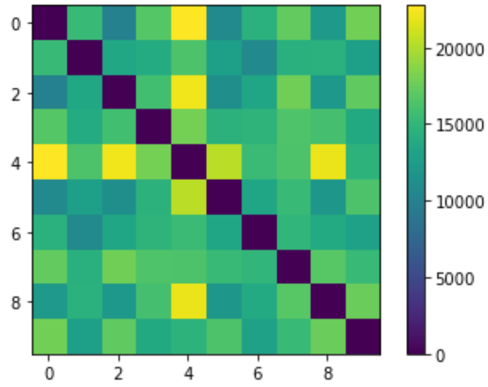
In [9]:
```python
resizedimages = [numpy.array(img.resize((100,100))).flatten() for img in images]

# ------------------------------------
# TODO: Replace by your code
# ------------------------------------
import solutions
D = solutions.task3a(resizedimages)
# ------------------------------------
```

The distance matrix can then be displayed using `matplotlib` .

In [10]:
```
plt.imshow(D)
plt.colorbar()
```

Out[10]: <matplotlib.colorbar.Colorbar at 0x7f2ec3de8280>



We cannot see clear similarities (low distances) within each image group. This suggests that the pixel representation does not encode well the concepts we are interested in.

To address this limitation, we consider a state-of-the-art neural network called `densenet161` and available pretrained in the `torchvision` libary. This neural network is composed of a feature extractor and a classification head. The feature extractor transforms image data (given as a `torch` tensor) into a tensor of activations where concepts are easier to predict.

(b) Compute the distance matrix between the images represented at the output of the densenet features extractor.

Hints: (1) The input images need to be converted to a torch tensor and the normalized using the function provided below before being fed to the neural network. (2) Note that the tensor at the output of the network can vary in shape due to the varying size of the input images. This can be addressed by applying the mean operation over the two dimensions representing the horizontal and vertical components.

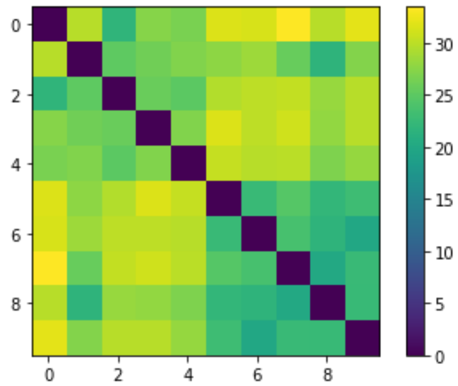
In [11]:
```
model = torchvision.models.densenet161(pretrained=True).features
model.eval();

def normalize(x):
    x = x - torch.Tensor([0.485, 0.456, 0.406]).reshape(1,-1,1,1)
    x = x / torch.Tensor([0.229, 0.224, 0.225]).reshape(1,-1,1,1)
    return x

# ------------------------------------
# TODO: Replace by your code
# ------------------------------------
import solutions
D = solutions.task3b(images,model,normalize)
# ------------------------------------

plt.imshow(D)
plt.colorbar()
```

Out[11]: <matplotlib.colorbar.Colorbar at 0x7f2ec441f9a0>



We observe that distances now form a block structure, where the first 5 images are clearly mutually similar, and similarly for the last 5 images.