Tutorial: Machine Learning

The objective of this tutorial is to classify images by using machine learning techniques. Some images, as illustrated in Figure 1 of the Kimia database will be used [1, 2].

Figure 1: The different processes will be applied on images from the Kimia database. Here are some examples.

(a) bird (b) camel (c) ray (d) turtle

1 Feature extraction

The image database used in this tutorial is composed of 18 classes. Each class contains 12 images. All these 216 images come from the Kimia database. In order to classify these images, we first have to extract some features of each binary image.

1. For each image in the database, extract a number of different features, denoted *nbFeatures*.
2. Organize these features in an array of *nbFeatures* columns and 216 lines (the shape is thus $(216, nbFeatures)$). In this way, each line i represents the different features of the image i.

You can use the Matlab function regionprops to extract some features such as area, eccentricity, perimeter, solidity...

You can use the Python function skimage.measure.regionprops to extract the same geometrical features. In order to load all image, from the directory, you can use this for loop and use the glob module in order to load all files.

```
import glob
rep = 'images_Kimia216/'
classes = ['bird', 'bone', 'brick', 'camel', 'car', '
    ↪ children',
           'classic', 'elephant', 'face', 'fork', '
              ↪ fountain',
           'glass', 'hammer', 'heart', 'key', 'misk', '
              ↪ ray', 'turtle']
nbClasses = len(classes)
nbImages = 12

# The features are manually computed
properties = np.zeros((nbClasses*nbImages, 9))
target = np.zeros(nbClasses * nbImages)
index = 0
for ind_c, c in enumerate(classes):
    filelist = glob.glob(rep+c+'*')
    for filename in filelist:
        print(filename)
```

2 Image classification

In order to classify the images, we are going to use neural networks. Pattern recognition networks are feedforward networks that can be trained to classify inputs according to target classes. The inputs are the features of each image. The target data are the labels, indicating the class of each image.

2.1 Construction of the array of properties

- Build the target data, representing the class of each image. The target data for pattern recognition networks should consist of one vector containing the index of the target class. In our example, the target data will be a vector of 216 elements.
- The database will be divided into a training set (75%) and a test set (25%).

Use `from sklearn.model_selection import train_test_split` to perform this partition into training/test sets.

2.2 Training and classification

- Run the training task and classify the test images.
- Show the classification confusion matrix as well as the overall performance.

Look at the Matlab function patternnet to make the classification. Look at the field *divideParam* of your pattern network for performing the division.

Look at the Python module sklearn.neural_network.MLPClassifier to make the classification. You may also use SVM classifier. Notice that your data may be normalized (use from sklearn.preprocessing import StandardScaler) before performing the classification.

- Try to run the same process again (starting from the traint/test split). What can you conclude?
- Try to change the parameters of the network to improve the classification performance.

References

[1] http://vision.lems.brown.edu/content/available-software-and-databases. 1

[2] Daniel Sharvit, Jacky Chan, Huseyin Tek, and Benjamin B Kimia. Symmetry-based indexing of image databases. In *Content-Based Access of Image and Video Libraries, 1998. Proceedings. IEEE Workshop on*, pages 56–62. IEEE, 1998. 1