SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 3: November 11–November 15, 2019

Exercises consist of both pen&paper and computer assignments. Pen&paper questions are solved at home before exercises, while computer assignments are solved during exercise hours. The computer assignments are marked by **python** and Pen&paper questions by **pen&paper**

1. **pen&paper** *Design an LDA classifier manually.*

   A dataset consists of two classes, whose distributions are assumed Gaussian, and whose sample covariances and means are the following:

$$\boldsymbol{\mu}_0 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \qquad \boldsymbol{\mu}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$
$$\mathbf{C}_0 = \begin{pmatrix} 2 & 0.2 \\ 0.2 & 0.5 \end{pmatrix} \qquad \mathbf{C}_1 = \begin{pmatrix} 3 & -1 \\ -1 & 1 \end{pmatrix}$$

   Calculate the projection vector $\mathbf{w}$. In order to be fully manual, invert the 2×2 matrix using the rule

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

2. **pen&paper** *Compute the threshold and classify.*

   In the lecture slides, we proposed to define the threshold T as the mean of the projected class means: $T = \frac{1}{2}\mathbf{w}^T(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0)$. This approach does not take into account the fact that the two classes have different spreads, and the threshold should probably not be exactly at the center.

   A more appropriate approach computes the projection of the multivariate Gaussians and sets the threshold as we did in detection theory. The projected Gaussians are univariate normal: $\mathcal{N}(\mathbf{w}^T\boldsymbol{\mu}_1, \mathbf{w}^T\mathbf{C}_1\mathbf{w})$ and $\mathcal{N}(\mathbf{w}^T\boldsymbol{\mu}_2, \mathbf{w}^T\mathbf{C}_2\mathbf{w})$. Formulate the classification problem as a likelihood ratio test and choose the threshold based on that.

   Which class will be predicted for sample $\mathbf{x} = (1, 2)$?

3. **python** *Load a dataset of images split to training and testing.*

   We will train a classifier to classify hand written digits. Scikit-learn provides a number of sample datasets. Load the `digits`-dataset as follows.

```
from sklearn.datasets import load_digits
digits = load_digits()
```

The result is a `dict` structure that can be accessed using *keys*. Find all keywords of the dict with `print(digits.keys())`. The interesting ones for us are: `'images'`,`'data'` and `'target'`.

Plot the first image of the 1797 numbers like this.

```
import matplotlib.pyplot as plt
plt.gray()
plt.imshow(digits.images[0])
plt.show()
```

Check that this corresponds to the label `digits.target[0]`.

The images are vectorized as rows in the matrix `digits.data`, whose size is 1797×64 (1797 images of size 8×8).

Split the data to training and testing sets, such that the training set consists of 80% and test set 20% of the data. Use `sklearn.cross_validation.train_test_split` to do this and create variables `x_train`, `y_train`, `x_test`, `y_test`.

Create a list of four classifiers with their default parameters:

- `sklearn.neighbors.KNeighborsClassifier`
- `sklearn.discriminant_analysis.LinearDiscriminantAnalysis`
- `sklearn.svm.SVC`
- `sklearn.linear_model.LogisticRegression`

Train each classifier in a for loop and assess the accuracy in the test set using `sklearn.metrics.accuracy_score`. Which one is the best?

Run the code again. Do you get the same result? Why not?

4. **python** *Train classifiers for the GTSRB task.*

In this exercise we will extract image features for categorization of traffic signs. Download the following file:

`http://www.cs.tut.fi/courses/SGN-41007/GTSRB_subset.zip`

If has two folders each containing 100 images from the German Traffic Sign Recognition Benchmark (GTSRB); a competition organized in IJCNN-2011 conference. There is also a template for loading the data and extracting their *Local Binary Pattern* features. The result is a feature matrix `F` and label vector `y`.

Use the same classifiers as in Question 3 for training a model for the GTSRB data using the LBP features. However, instead of evaluating the performance for a single train/test split, study the use of `sklearn.model_selection.cross_val_score`, which creates N splits computes their average accuracy.

5. `python` *Train ensemble methods with the GTSRB data. These did not yet appear in the lectures, but Google is your friend.*

   a) Train a 100-tree Random Forest classifier with the GTSRB and compute the accuracy.

   b) Train a 100-tree Extremely Randomized Trees classifier with the GTSRB and compute the accuracy.

   c) Train a 100-tree AdaBoost classifier with the GTSRB and compute the accuracy.

   d) Train a 100-tree Gradient Boosted Tree classifier with the GTSRB and compute the accuracy.