SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 4: January 29–February 2, 2018

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 text `python` and Pen&paper questions by text `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:

\[ \mu_0 = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \mu_1 = \begin{pmatrix} -5 \\ 2 \end{pmatrix} \]
\[ C_0 = \begin{pmatrix} 2 & 0.1 \\ 0.1 & 0.2 \end{pmatrix} \quad C_1 = \begin{pmatrix} 3 & -1 \\ -1 & 2 \end{pmatrix} \]

Calculate the projection vector \( w \). In order to be fully manual, invert the \( 2 \times 2 \) 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: \( T = \frac{1}{2} w^T (\mu_1 + \mu_0) \). 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: \( N(w^T \mu_1, w^T C_1 w) \) and \( N(w^T \mu_2, w^T C_2 w) \). Formulate the classification problem as a likelihood ratio test and choose the threshold based on that.

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

3. `python` Extract Local Binary Pattern features for classification.

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.
Load all images from the two folders and compute their Local Binary Pattern features using `skimage.feature.local_binary_pattern`. The function returns an image with same size as the original, so you will also have to compute the histogram with `numpy.histogram`. Note that this is a similar task to exercise 4 last week. The result should be a feature matrix $X$ and label vector $y$.

4. **python**  
Train classifiers for the GTSRB task.
Create a list of three classifiers with their default parameters:

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

Split your data into two parts—80% for training and 20% for testing—using `sklearn.cross_validation.train_test_split`.
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?

5. **python**  
Implement two utility functions we need later.

a) Implement the following function:

$$p = \text{gaussian}(x, \mu, \sigma)$$

which returns the Gaussian density with parameters $\mu$ and $\sigma$ at point $x$. In mathematical notation, the function is:

$$p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[\frac{-1}{2\sigma^2} (x - \mu)^2\right]$$

b) Implement the function:

$$p = \text{log_gaussian}(x, \mu, \sigma)$$

that returns $\ln p(x; \mu, \sigma)$. Do not use the the straightforward solution $p = \text{numpy.log(gaussian(x, \mu, \sigma))}$ would be numerically inaccurate. Instead, first manually simplify the expression

$$\ln p(x; \mu, \sigma) = \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[\frac{-1}{2\sigma^2} (x - \mu)^2\right]\right)$$

and write the corresponding code.

c) Plot both functions for $\mu = 0$ and $\sigma = 1$ in the interval $x \in [-5, 5]$ to check that the code is correct. *Hint:* create a vector of $x$-coordinates using `numpy.linspace` and pass that through your functions and plot.

\(^1\text{See }\)http://scikit-image.org/docs/dev/auto_examples/plot_local_binary_pattern.html