University of
Stavanger

Stavanger, January 2, 2020

Laboratory exercise 4

ELE520 Machine learning

A PDF version of the report of the student solution to the exercise including figures and answers to questions shall be submitted on CANVAS.[1]

In this laboratory project the objective is to develop a classifier based on the use of the parametric and nonparametric approaches explored in the previous exercises.

In the first two problems, you will work with two datasets. One for training and one for testing. The two datasets are labelled, so that we know the true state of nature (class relationship) for all of the feature vectors.

When we design a classifier, it is very important to know how its expected performance will be when it will classify new unlabbeled data in the future. One such performance measure is the error rate.

To get an idea of how the classifier will perform on unlabelled data, we will classify data not included in training the classifier. This is the role of the test data set. We will let the classifier label all the test data, and we will compare these proposed labels with the true ones.

If we want to compare several classifier designs, we can evaluate the performance of each and then select the one with best performance.

It will also be interesting to classify the training data used for designing the classifier. This will give us an idea if the classifier is overtrained or not. We will term this as doing *reclassification*. If the reclassification error rate is much lower than the rerror rate found when classifying the test data, this indicates that the classifier is overtrained.

To be able to do this, you can follow the following procedure:

1. Classify the training data labelled as class 1 and count the number of misclassifications.
2. Classify the testing data labelled as class 1 and count the number of misclassifications.
3. Repeat the two previous steps until the number of misclassifications are registered for all classes.

[1] If it is not possible to export the Jupyter notebook to PDF, the ipynb and py files can be submitted.

4. Divide the total number of misclassifications in the training set by the total number of training vectors to determine the reclassification error rate.
5. Divide the total number of misclassifications in the test set by the total number of test vectors to determine the test classification error rate.

There are som more performance measures to be found, which can easily added once a confusion matrix has been computed.

The confusion matrix $A = [A(i,j)]$, $i = 1, \dots, c$, $i = 1, \dots, c$ is defined so that its element $A(i,j)$ is the number of feature vectors labelled ω_i that are classified as ω_j. The following performance metrics are common, using the information in the confusion matrix:

- The *Recall* (R_i) which is the percentace of feature vectors with true state of nature ω_i that are correctly classified.

$$R_i = \frac{A(i,i)}{\sum_{j=1}^{c} A(i,j)} \tag{1}$$

- The *Precision* (P_i) which is the percentace of feature vectors classified as ω_i that are correctly classified.

$$P_i = \frac{A(i,i)}{\sum_{j=1}^{c} A(j,i)} \tag{2}$$

- The *Overall Accuracy* (Ac) which is the percentace of feature vectors that are correctly classified.

$$Ac = \frac{\sum_{i=1}^{c} A(i,i)}{\sum_{i=1}^{c} \sum_{j=1}^{c} A(i,j)} \tag{3}$$

The first thing to do is to make the classifier that designs the classifier based on the training data information and classifies input data it does not have information of the true state of nature.

Problem 1

a) In the file *lab4.p* you will find simulated data for a two class problem. In the cell structures `X` and `Y` you will find data that can be used for training and testing of the classifier. The data can be loaded as shown in the following lines of code:

```
        pfile = 'lab4.p'
        with open(pfile, "rb") as fp:
            X, Y  = pickle.load(fp)
```

b) The work you have conducted on the laboratory so far has focused on computing and visualising *density and discriminant* functions. You can use the programs you have made in these exercises to make a *classifier*.[2]

Make a program to be called by giving the command *classify*. Let the the input training data be (Y) and data to be classified (X).

- It might be convenient to let Y be a list with M cells, where Y[i] corresponds to the training data from class $\omega_i, i = 1, \ldots, M$.
- It might also be convenient to let X be a data matrix of containing the feature vectors to be classified.

Upon calling the function, it shall return

- g containing the discriminant function values for each feature vector in X. It might be convenient to let g be a matrix so that row number i in g, g(i,:), coresponds to the discriminant function values $g_i(\mathbf{x})$, where $i = 1, \ldots, M$.
- C indicating the classification for each feature vector in X.

c) Use a ML-classifier:

- Classify the training set (reclassification).
- Classify the test set.
- Compute the error rate ($P(\text{error}) = 1 - A_c$) and the ratio of correct classifications for each of the classes, $P(\text{correct}|\omega_i) = R_i, i = 1, 2, \ldots, M$). It is recommended to determine the confusion matrix and use this as a basis to determine these performance metrics.

The following listings shows a suggested code for the reclassification and the testing.

```
    for k in range(0, M):
        gx[k], Cx[k], pxwx[k], Pwx = classify(Y, X[k],
met, discr, prm)
        gy[k], Cy[k], pxwy[k], Pwy = classify(Y, Y[k],
met, discr, prm)
```

The variables generated in these iterations are available in the file *lab4_data.p*. If you want to check that your pdfs, discriminant functions and classifications, the variables can be loaded according to the following code.

```
pfile = 'lab4_data.p'
with open(pfile, "rb") as fp:
    pxwx, Pwx, gx, Cx, CNx, pxwy, Pwy, gy, Cy, CNy  =
pickle.load(fp)
```

[2] My own solution was made by basing the classify function described below on the labsol4 function from the previous laboratory exercise. In the classify function all visualisation has been removed and discriminant function values are estimated for the feature vectors in the data set. New functions normdD and knndD have been made to accomodate this, based on the norm2D and knn2D functions.

d) Repeat the previous subtask using a Parzen-classifier with window width $h1 = 0.1$.

e) Repeat the previous subtask using a Parzen-classifier with window width $h1 = 5$. Comment the results.

f) Repeat the previous subtask using a k_n-nearest neighbor classifier with $k_n = 1$. Comment the results.

g) Repeat the previous subtask using a k_n-nearest neighbor classifier with $k_n = 5$. Comment the results.

The classification results your classifiers should yield in problems 1 c-e are given in table 1.

Classifier	$P(error)$	$P(correct\|\omega_1)$	$P(correct\|\omega_2)$
ML reclassification	0.10	0.93	0.85
ML testing	0.05	0.97	0.94
Parzen $h1$=0.1 reclassification	0.00	1.00	1.00
Parzen $h1$=0.1 testing	0.20	0.91	0.64
Parzen $h1$=5 reclassification	0.08	0.93	0.91
Parzen $h1$=5 testing	0.06	0.95	0.93
Nearest neighbor k_n=1 reclassification	0.00	1.00	1.00
Nearest neighbork_n=1 testing	0.08	0.95	0.86
Nearest neighbor k_n=5 reclassification	0.09	0.93	0.89
Nearest neighbork_n=5 testing	0.07	0.93	0.91

Table 1: Results from reclassification and testing

Problem 2

In this task, you must adapt the classifier so that it is capable of handling an arbitrary number of classes, M, and feature vectors with arbitrary dimension, l.

To be able to develop the classifier, several datasets with differing number of classes and dimension have been simulated and made available in the data file *lab4_2.p*. The data can be loaded as shown in the following lines of code:

```
  pfile = 'lab4_2.p'
  with open(pfile, "rb") as fp:
      X_2D3cl ,  X_2D4cl ,  X_2D4cl_ms ,  X_2D4cl_hs ,
 X_3D3cl_ms, Y_2D3cl ,  Y_2D4cl ,  Y_2D4cl_ms ,  Y_2D4cl_hs ,
 Y_3D3cl_ms  = pickle.load(fp)
```

The variables contain training and test data (`Y/X`) for various dimensions (`2D/3D`) and number of classes (`3cl/4cl`) and an optional indicator of medium or high separation (`ms/hs`).

In each of the following subtasks, you will repeat the experimental procedures used in problem 1. The tables showing the results should have the same layout as table 1 with the difference that extra columns will be added as the number of classes increase.

You will also make scatter plots of the data, using different colors and symbols for the different classes. Use two subplots, one for the training data and one for the test data..

a) Make scatter plot and classify the 2D 3 class data set.

b) Repeat the previous subtask, but this time for the 2D 4 class data set.

c) Repeat the previous subtask, but this time for the 2D 4 class data set with medium separability.

d) Repeat the previous subtask, but this time for the 2D 4 class problem with high separability.

   Explain the difference in results comparing subtasks b-d.

e) Repeat the previous subtask, but this time for the 3D 3 class problem with medium separability.