Due Date: Computerpraktikum am 18.07. und 19.07.2024

Verantwortlich für Vorlesung bzw. Computerpraktikum:
Dr. P. Goldenzweig, Dr. J. Kieseler, Dr. S. Stefkova
Tutoren: Dr. G. De Pietro, Dr. R. Quishpe, M. Mormile, J. Eppelt, A. Brusamolino, Dr. X. Zuo, Dr. S. Brommer

Computerpraktikum zur Vorlesung Moderne Methoden der Datenanalyse - Blatt 9

Exercise 9: Multivariate Classification

This exercise is dedicated to studies of a multivariate data-set, the famous 'Iris flower data-set' used in the year 1936 by Sir Ronald Fisher as an example of discriminant analysis. The data-set describes the morphological variation of Iris flowers and contains 50 samples from each of the three species Iris-setosa, Iris-virginica and Iris-versicolor.

The file `iris.data` (available in the exercise repository) contains the data. The columns contain four morphological features of the three species: the sepal length and width and the petal length and width (all in cm), and the species as classified by a botanist (Iris-setosa = 0, Iris-versicolor= 1, Iris-virginica = 2). Fortunately, the biological details are not of too much importance for the studies proposed here, and you may safely name the variables $v_1, \ldots, v_4$ and call the classes '1', '2' and '3'. In this exercise you will develop classifiers based on this data-set as a training sample to identify different iris species, for people who are not particularly gifted botanists.

To guide your own implementation of studies of the data-set, a code basis is provided in the corresponding jupyter notebook. Inspect it and use it as a starting point for your own work. Study the example code, which contains a general python class `Plotter`, which provides methods to display several aspects for the evaluation of the performance of a classifier. The classifier itself is to be implemented as another class containing a method `evaluate()`, which returns the value of the test-statistic under study and is used by the class `Plotter`. The method `plot_contour()` implemented in this class draws scatter plots of the data superimposed with contours representing the result of evaluation of the test-statistic. Also included is a method `plot_test_statistic()` to display the distributions of the test-statistic on background and signal samples, and a method `plot_roc()` to visualise the performance, the so-called 'ROC' curve. Also included in the template is an implementation example of a classifier class, `CutClassifier`, which implements a simple, cut-based analysis, where the method `evaluate(x)` simply returns the number of cuts satisfied by the point in variable space given in the array x.

- **Exercise 9.1:** **voluntary**

  In multivariate analyses, it is most important to gain an overview and some initial level of understanding before deciding on strategies for optimal classification. One-dimensional distributions (= histograms) and pair-wise scatter plots of the available features (= variables) are the best way to visualise the available raw information. Use the examples in the template class `Plotter` to make these plots, ideally arranged as two-dimensional array of histograms (on the diagonal) and scatter plots (off-diagonal). With appropriate colour coding, all three classes of the data-set can be shown in the same set of histograms.

Which variables provide the best separation of Iris-setosa from the other two species? How can Iris-versicolor or Iris-virginica be separated from the others?

In the following (obligatory) part, we only use the first two variables in each line of the data-set, the sepal length and width, to separate one of the species, the ‘signal’, from the other two, the ‘background’.

- **Exercise 9.2: Naive Likelihood-Ratio** **obligatory**

  As a very simple approach, we start by building a two-dimensional histogram of the likelihood ratio of the signal and background distributions. First, choose a ‘suitable’ binning, and then, in each bin, determine the signal-to-background ratio, which already is the desired Likelihood ratio in the bin.

  Implement a class in analogy to the example provided in `CutClassifier`. In particular, implement the method `evaluate(x)` for your new class that returns the likelihood-ratio for each bin corresponding to the test point x.

  Use Iris-versicolor as the signal, the other species as background. Visualise the results in the two-dimensional parameter space using the methods of the provided class `Plotter`.

  What are the problems of this simple approach? What happens if you use Iris-setosa as the signal?

  Hint: Use the `numpy` function `histogram2d` (or *histogramdd*) to create a two-dimensional histogram. You may profit from the function `find_bin` defined in the provided code to find the bin which corresponds to a given point in variable space.

- **Exercise 9.3: Logarithm of Likelihood-Ratio** **obligatory**

  Next, we use parametrised distributions, assuming that the signal and background distributions can be reasonably described by two-dimensional Gaussian distributions, defined by the mean values and covariance matrices of the signal and background data-sets.

  Implement a class with a method `evaluate` returning the logarithm of the likelihood-ratio for each point in the sample.

  Visualise your results, as in the previous exercise, with Iris-versicolor as the signal.

  What is the advantage of this approach compared to the previous case? What happens if you use Iris-setosa as signal?

  Hint: The `numpy` functions `mean`, `cov`, `.dot` and `linalg.inv` are very helpful for this problem.

- **Exercise 9.4: Fisher’s discriminant** **obligatory**

  In order to study a truly one-dimensional classifier, implement Fisher’s discriminant. Write an appropriate class which calculates the Fisher vector in the `fit()` method, and write a method `evaluate(x)` returning the projection of the feature vector x onto the fisher vector. Study the resulting plots. Use again Iris-versicolor as the signal.

  What is better in this approach, and where are possible limitations? What happens if you use Iris-setosa as signal?

- **Exercise 9.5:** **voluntary**

  With the code implemented so far, you are ready now to use the full four-dimensional data, i.,e. to employ all four features (sepal length, sepal width, petal length, petal width), hopefully with only minor modifications or additions to your code.

Is it useful to consider the approaches of exercises 9.2 and 9.3 (naive likelihood-ratio in histogram bins, or the parametrisations of signal an background by four-dimensional Gaussian distributions) in the four-dimensional feature space?

With your implementation of Fisher's Method you are ready to separate Iris-virginica and Iris-versicolor even better. Give it a try!

Remark: The basic code skeleton you have developed in this exercise can easily be used for other data-sets, and an extension by other methods, e.g. those mentioned in the lecture, is also thinkable.