Statistical Data Analysis
Problem Sheet # 5
Due Monday 16 November 2020

For following exercises, please turn in the requested calculations, numbers and plots. Print any relevant code (only modules you wrote or modified in a nontrivial way) into pdf files and merge this with the rest of your problem sheet.

Exercise 1: For this exercise you will do a simple multivariate analysis either with the python `scikit-learn` package or using C++ with TMVA from ROOT. The input data consists of events of two types: signal and background. Three quantities $\mathbf{x} = (x_1, x_2, x_3)$ are are measured for each event. The marginal distributions of each of the three components are shown in Fig. 1.

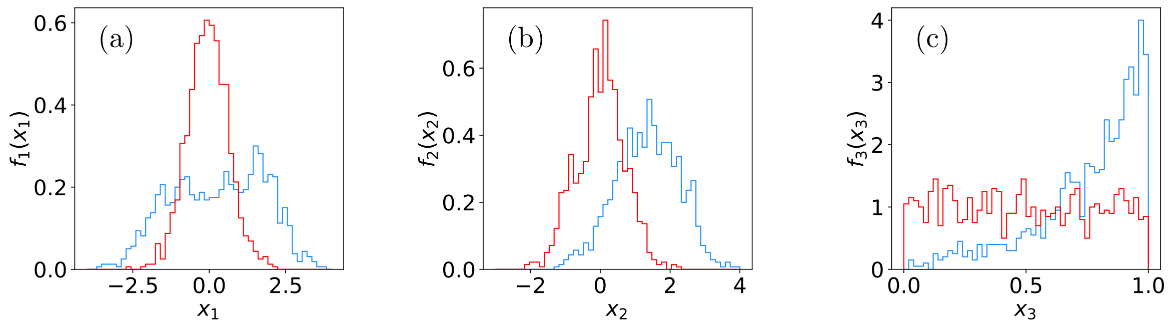

Figure 1: Marginal distributions of the three components of the feature vector $\mathbf{x} = (x_1, x_2, x_3)$ for events of the two classes: signal ($y = 1$, blue) and background ($y = 0$, red).

For the python option: This option uses python's `scikit-learn` package. As a starting point you can use the code here:

`http://www.pp.rhul.ac.uk/~cowan/stat/python/sklearn/`

The can use directly the input files `signal.txt` and `background.txt`, which contain each 10 000 events. For information on the various classifiers in `scikit-learn` see the documents on `scikit-learn.org`, specifically the very useful sample program at

`scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html`

For the C++ option: As a starting point use the code from

`http://www.pp.rhul.ac.uk/~cowan/stat/root/tmva/`

Everything needed is within that web page, but to get all of the code in the correct directories it is best to first copy the file `tmvaExamples.tar` to your working directory on either the RHUL linux cluster or a similar environment with root 6 installed.

Then from your directory type `tar -xvf tmvaExamples.tar`. This will unpack all of the files into the correct subdirectories within your working directory. Look in the `readme.txt` file for further details on how to build and run the various programs. Briefly, the relevant subdirectories and their programs are:

`generate` Contains a Monte Carlo program that generates data for training and testing of classifiers and stores them in the files `trainingData.root` and `testData.root`. These files are already there and you do not need to recreate them unless you want to generate more data or change the parameters of the models.

`tmvaTrain` Trains the classifier. In the default version, it determines the coefficients of a Fisher discriminant. When you run the program, the coefficients of the discriminating functions are written into a subdirectory `dataset/weights` as xml files.

`analyzeData` Analyzes the data using the test sample `testData.root` and the trained classifier.

To modify the programs `tmvaTrain.cc` to include a multilayer perceptron with one hidden layer containing 3 nodes, or to have a Boosted Decision Tree with 200 boosting iterations use you need to add:

```
factory->BookMethod(TMVA::Types::kMLP, "MLP", "H:!V:HiddenLayers=3");
factory->BookMethod(TMVA::Types::kBDT, "BDT200", "NTrees=200:BoostType=AdaBoost");
```

See the TMVA manual for more details. This will store the classifier's parameters in a file in the `dataset/weights` subdirectory.

Exercise 1: The data files supplied contain events of the two classes, signal and background. Each event is characterized by three quantities, $\mathbf{x} = (x_1, x_2, x_3)$.

The program provided already contains a Fisher discriminant and allows one to evaluate for each event the test statistic $t(\mathbf{x})$. Using this we want to select a sample of events enriched in signal by requiring $t > t_c$ with $t_c = 0$. That is, we test for each event the hypothesis that it is of the background type, and we reject that hypothesis (and thus select as signal) if $t > t_c$.

1(a) The code you are given calculates already the signal efficiency (the power of the test), i.e., $\varepsilon_s = P(t > t_c|\mathrm{s})$. Add the necessary code to find the background efficiency (the size of the test) $\varepsilon_b = P(t > 0|\mathrm{b})$.

1(b) What is the signal purity of the selected sample, i.e., $P(\mathrm{s}|t > t_c)$? Assume that the two event classes have equal prior probabilities.

1(c) Modify your program to include a multilayer perceptron with one hidden layer containing 3 nodes. Using the test data samples, make a histogram of the neural network's decision function for both event types. (With scikit-learn, you will need to use instead the output of the function `predict_proba`, which is monotonically related to the MLP output.) Compare to the corresponding histogram from the Fisher discriminant.

1(d) Select signal events by requiring $t_{\mathrm{MLP}} > t_c$ with $t_c = 0.5$. What are the signal and background efficiencies? What is the signal purity assuming equal prior probabilities for the two event types?

Exercise 2 (not required, but recommended for PhD students) This exercise is a continuation of Ex. 1.

2(a) Modify your program to include a boosted decision tree with 200 boosting iterations.

2(b) Now repeat this for different numbers of boosting iterations, say, 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 10,000, 50,000.

For each classifier, compute the total error rate using a boundary value of $t_c = 0$. That is, compute the fraction of events (signal and background) that are found on the wrong side of the boundary. Plot this as a function of the number of boosting iterations for both the training sample and the statistically independent test sample. Determine roughly the optimal number of boosting iterations.