??? GENERAL NOTES ON SUBMISSION, GRADING, ETC. ???

The programming problems consist in filling the gaps in the provided scripts according to the problem description and the comments within the code. The missing parts are typically indicated by '`...`' (not to be confused with line breaks!), whereas each '`...`' might be replaced by multiple lines of code. Following the provided code is recommended but only leads you to one of many possible solutions. If some part of the code seems unclear or counterintuitive to you, feel free to depart from it. Be aware, however, that the suggested variable names and structures are typically used later in the script, e. g. for plotting, and occur again in other scripts and problems. In any case, your code should produce the same output as the Musterlösung. **Note**: To be able to run the code, you must have the folder `utility` and all subfolders as well as the location of the ECG/EEG datasets added to your MATLAB path.

Main Script

In this problem, you will generate bivariate random data from two normal distributions and classify the data accordingly using logistic regression. The script `runTest.m` provides the main parameters.

First, you have to produce the data using these parameters and an appropriate function, concatenate the two sample populations, and create a vector of labels (ones for one population, zeros for the other). `X` will be a 2*`nSamples`×2 matrix and `L` will be of length 2*`nSamples`.

Next, we want to see what the two underlying distributions look like along the two data dimensions x_1 and x_2 (not to be confused with the two sample populations `X1` and `X2`). Create a grid of 100×100 equidistant points within the bounds of the sample populations, using `xVector`, and compute the two PDFs for each point. (If you have trouble with the suggested one-liners, use for-loops instead.) The result `pDist` should be a 100×100 matrix containing the sum of both PDFs.

Now we train a General Linear Model on the sample data. Look into the MATLAB documentation for `glmfit` to find the right arguments for the distribution and link function.

To display the decision boundary of the fitted model, we solve the iso-error problem, i. e. we compute the line on which $P(C_1|x) = P(C_2|x) = 0.5$. This means solving the equation

$$0.5 = \sigma(w^T x + w_0)$$

with σ being the logistic function, $w = (w_1, w_2)$ together with w_0 given by the coefficients `coeff` we just computed, and $x = (x_1, x_2)$. We have to rewrite the equation for x_2 and set x_1 to `xVector`.

Now we validate the fitted model (find the appropriate function!) on the same 100×100 data points as before to get the posterior densities predicted by the model. (We only compute $P(C_1|x)$, since $P(C_2|x) = 1 - P(C_1|x)$.)

Finally, we want to cross-validate the model, using our own function.

Classification Function

The function `modelFitVal.m` should take a data matrix, a correpsonding label vector, and the number of partitions, then iteratively divide the data and labels into training sets and test sets, fit and validate the GLM, and return the mean percentage of correctly predicted labels.

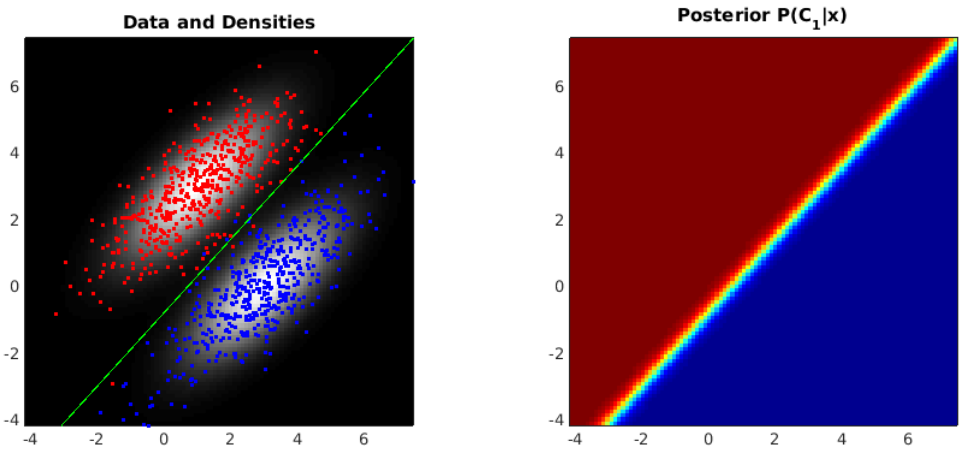
First, you should try to set up a vector of randomized partition tags with values 1 to `kPart` (e. g. `partTag = [2 1 4 3 1 3 ...]` for `kPart = 4`) and the same length as `X`.

In the subsequent loop we divide the data and labels into test and training sets using this vector. In each iteration, select the test set via the current partition tag and the training set via the remaining `kPart-1` tags.

Now you perform logistic regression as before and compare the rounded validation data, i. e. the predicted labels for the test data, to the actual labels. The percentage of correct predictions is then accumulated over iterations.

Plotting and Exploring

Your code should print the model's average classification performance and produce the two figures below, showing the sample data with their underlying distributions and the model's decision boundary, and the class posterior probabilities, respectively.



Note that the plots will vary a bit at between runs because of the randomness in sampling and cross-validation. Classification performance should be at or near 100 %.

You may want to try different distribution parameters and sample sizes and see how performance and plots change. Try to understand the warning messages you might get.