LAB1: Binary classification and model selection

> *Goal: to play with the libraries and to get a practical grasp on how to deal with binary classification and model selection problems on synthetic data.*

This lab is divided into two parts depending of their level of complexity (**Beginner, Advanced**). Your goal is to complete entirely, at least, one of the two parts. Please note that a different notation can be used in the code as we used in the lectures.

PART I: Beginner

Warm up

Type in the Terminal 'python gui_filter.py' and a GUI will start. Have a look at the various components.

- With the data simulation option generate a classification dataset of type *linear* (press the button **Load data** to generate).
- Observe the generated data (two figures with test and training sets respectively). Change the value of *wrong label ratio* (between 0 and 1), load the data again and observe changes.
- Choose the *regularised least squares* filter and the *linear kernel*.
- Have a look at the parameter selection part and the various options of KCV (K-Fold Cross Validation). To choose the regularisation parameter you can either choose KCV or set a fixed value.
- Press the button **Run** to perform training and classification. Observe the plot of the KCV error and the balance between training and test errors. Also have a look at the plot area at the bottom, where a separation function has appeared.

The Geek part

Check the content of directory ./spectral_reg_toolbox. There you will find, among the others, the code for functions learn.py (used for training), patt_rec.py (used for testing), kcv.py (used for model selection on the training set). For more information about the parameters and the usage of the scripts, look inside the files.

Finally, you may want to have a look at the content of directory ./dataset_scripts and in

particular to file `create_dataset.py`, that will allow you to generate synthetic data of different kinds.

Analysis

Carry on the following experiments either using the GUI, when it is possible, or writing appropriate scripts.

1) Generate data of *linear type*. Considering *linear-RLS*, observe how the training and test errors change as:
   - We change (increase or decrease) the regularization parameter `t`.
   - The training set size grows, try different `n`.
   - The amount of wrong labels (noise) on the generated data grows. For instance, `[0.01, 0.02, 0.05, 0.1]`.

Run training and testing for various choices of the suggested parameters.

2) Leaving all other parameters fixed, choose an appropriate range for the regularisation parameter `t`, `[t_min, t_max]` and `# of t values` in the interval. Plot the training and the test errors for each `t`. Use the KCV option to select the optimal regularisation parameter and see how it relates to the previous plot.
3) Leaving all other parameters fixed, choose an appropriate range `[n_min, n_max]` of the number of points in the training set and plot the training and test errors. What happens as n goes to infinity?.
4) Leaving all other parameters fixed, investigate how the training and test errors changes with the increase of the noise. Try some `(n, noise)` (number of samples; wrong label ratio) pairs and compare the obtained training and test error.

PART II: Advanced

Advanced Analysis

Carry on the following experiments either using the GUI, when it is possible, or writing appropriate scripts. In this part you have to focus on the on the correct choice and interplay between various parameters `t, sigma`.

5) Consider *gaussian-RLS* and perform parameter tuning in this case. This time, together with the regularization parameter `t`, you'll have to choose an appropriate kernel parameter `sigma`.
   - Try some `(sigma, t)` pairs and compare the obtained training error and test error.
   - Fix `t` and observe the effect of changing `sigma`.

- Fix `sigma` and observe the effect of changing `t`.
- Do you notice (and if so, when) any overfitting/oversmoothing effects?

6) Consider *polynomial-RLS* and perform parameter tuning as in (6). How does the choice of the kernel affect the learning behavior of the algorithm? In particular, compare the performances of the polynomial and Gaussian kernels on the *spiral* and *moons* datasets with respect to the number of examples in the training set, e.g. [10, 20, 50, 100, 1000], and the amount of regularisation ("fixed value" in the GUI, e.g. [0.5, 0.1, 0.01, 0.001, 0.0001]).