LAB 1: Binary classification and model selection

- This lab addresses binary classification and model selection on synthetic data.
- The aim of the lab is to play with the libraries and to get a practical grasp of what we have discussed in class.
- Follow the instructions below.

Goal:
This lab is divided in three parts depending of their level of complexity (**Beginner**, **Intermediate**, **Advanced**). Your goal is to complete entirely, at least, one of the three parts.

Download the file `regml2016_lab1.zip`, extract it and add all the sub-folders to the MATLAB path. This file includes all the code you need!

PART I: Beginner

Overture: Warm up

Run the file `gui_filter.m` 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 (the buttons plot training and plot test will allow you to toggle between training and test sets).
- Choose the *regularized 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 regularization 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 on the left, where a separation function has appeared (again the buttons plot training and plot test allow you to switch between the two).

Interlude: The Geek Part

Back on the MATLAB shell, check the content of directory `./spectral_reg_toolbox`. There you will find, among the others, the code for command `learn` (used for training), `patt_rec` (used for testing), `kcv` (used for model selection on the training set).

For more information about the parameters and the usage of those scripts, type:

```
>> help learn
>> help patt_rec
>> help kcv
```

Finally, you may want to have a look at the content of directory `./dataset_scripts` and in particular to file `create_dataset.m`, that will allow you to generate synthetic data of different kinds.

> **NOTE:**
> In the code we use a different notation from the one you have seen in the classes. In the *Regularized Least Squares* method (`rls.m`), the regularization parameter is `t` instead of λ.

Allegro con brio: Analysis

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

i) 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 various choices of `n` as long as MATLAB supports you!).
- The amount of noise in the generated data grows.

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

ii) Leaving all the other parameters fixed, choose an appropriate range for the regularization parameter `t`, `[t_min:t_step:t_max]` and plot the training and the test errors for each `t`. Use the KCV option to select the optimal regularization parameter and see how it relates to the previous plot.

iii) Leaving all the other parameters fixed, choose an appropriate range `[n_min:n_step:n_max]` of the number of points in the training set and plot the training and test errors (what do you observe as `n` $\to \infty$?)

PART II: Intermediate

Crescendo: Advanced Analysis

iv) 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 `sigma`, the kernel parameter.

- 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?

v) Consider *polynomial-RLS* and perform parameter tuning as in (iv). 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 regularization ("fixed value" in the GUI, eg. `[0.5, 0.1, 0.01, 0.001, 0.0001]`).

PART III: Advanced

Finale: The Challenge

The challenge consists in a learning task using a real dataset, namely *USPS (United States Postal Service)*: This dataset contains a number of handwritten digits images. The problem is to train *the best classifier* that is able to discriminate between the digits $\boxed{1}$ and $\boxed{7}$.

Once the classifiers are trained, they must be exported by means of the **`save_challenge_1.m`** script (to see how to use it, try the command `help save_challenge_1`). The file **`demo_lab1.m`** contains a code snippet to perform a simple binary classification task by means of the previously presented scripts.

Submission: You should drop your results in a MATLAB matrix file named `name-surname.mat` to the link: `http://www.dropitto.me/regml2016` with password `regml2016` by the end of the challenge session.
The results will be presented during the next class and the first five will be awarded an **awesome gift**. The score is based on the accuracy of the classifier on a completely independently sampled test set.

Deadline: 6:00 PM.