LAB4: Spectral filters and binary classification

Goal: to provide an overview of properties and roles of spectral filters for 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.

Please note that the lab deals with some material, which has not been presented in the lecture. Therefore, some additional reading is required. We refer to the Wikipedia page Regularization by spectral filtering.

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 dataset of type *spiral* (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 *Truncated SVD* filter and the *Gaussian kernel* (be sure to check the `autosigma` checkbox as *on*).
- 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.

NOTE: for the *NU-Method* (nu.py) and the *Landweber Method* (land.py), the regularization parameter t is the number of iterations, roughly $t \sim 1/\lambda$ speaking

Analysis

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

1) Generate data of *spiral type*. Considering three algorithms, namely *RLS, Truncated SVD and NU-Method*, observe how the training and test errors change as:
   - We change (increase or decrease) the cardinality of the training set. Set for instance, [10, 100, 1000] as long as computational time is reasonable.
   - We change (increase or decrease) the amount of regularisation for each algorithm. For instance, use *fixed value*.
   - The amount of wrong labels 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, use the KCV option to select the optimal model and see how it relates to previous results. Choose an appropriate range for the regularisation parameters, and plot the training error and the test error for each regularisation parameter.
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? How the different regularisation parameters affect the learning process? Which are the main differences in terms of regularisation between the methods?

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 effects of the regularisation and on the correct choice of `sigma`.

4) Consider *Gaussian kernel* and perform parameter tuning. 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?
5) Compare *RLS* and *Landweber Method* on a kernel of your choice
   - Tune the parameters with KCV.
   - Compare the time needed to obtain a solution.
   - Compare the training and test errors.