Regularization Methods for Machine Learning

LAB 3: Sparsity-based learning

- This lab is about feature selection within the framework of sparsity based regularization, using elastic net regularization.
- 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 two parts depending on their level of complexity (**Beginner**, **Intermediate**). Your goal is to entirely complete at least one of the two parts.

Setup instructions:
Running MATLAB Clone or download the course repository at `https://github.com/LCSL/RegML`. Add all the subfolders to the MATLAB path. The repo includes all the files you need!

Toy problem

We focus on a regression problem where the target function is linear. We will consider synthetic data that is generated (randomly sampled) according to a given probability distribution and affected by noise. You will be able to control the sizes of training and test sets, dimensionality of the data and the number of relevant features.

NOTE:
In the code we use different notation from what you have seen in class. Namely, we consider the minimization problem

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{2n}\|X\beta - Y\|^2 + \texttt{L2\_par}\frac{|||X|||^2}{2n}\|\beta\|_2^2 + \texttt{L1\_par}\|\beta\|_1,$$

where $X \in \mathbb{R}^{n \times p}$ and $Y \in \mathbb{R}^n$.

Furthermore, the sparsity parameter `L1_par` is, in some of the code, denoted by `tau`, while the smoothing parameter `L2_par` can be referred to as `smooth_par`.

PART I: Beginner

Overture: Warm up

Run the file `gui_l1l2.m` and the GUI will start. Have a look at the various components.

- **I.A** Generate a training set with the default parameters. Press the run button to start a training phase with the selected `L1_par` and `L2_par` parameters and perform testing. The first figure displays the cross validation error together with the test error. The second figure displays the sparsity of the reconstructed signal for the given `L1_par`. Compare the sparsity of the reconstructed signal with that of the original signal.
  Note: In this lab we perform regression, not classification. Therefore, the error is no longer measured as a percentage of wrongly assigned labels, but is rather measured by quantities of the form $(1/n)\|X\beta_{pred} - Y\|^2$.
- **I.B** Train your dataset by changing the values of `L1_par` and `L2_par` in `learnSparse_octave.m`.
  - **I.B1** Fixing `L2_par` $= 0$ and playing with `L1_par`, try to obtain a sparser or denser solution. How does the sparsity behave with respect to `L1_par`?
  - **I.B2** Repeat the experiment with `L2_par` > 0. How do the test error and the number of selected features vary?
- **I.C** Repeat the experiments of I.B1, but this time considering a noisy dataset (you can set the noise parameter to 0.3). How does this affect the sparsity? What can you say about the training error?

Allegro con brio: Analysis

In the following experiments either use `loadSparseDataset_octave.m` and `learnSparse_octave.m`, or, when needed, write the required code.

- **I.D** Have a look at the contents of the directory `./PROXIMAL_ TOOLBOXES/L1L2_TOOLBOX`. There you will find, among other things, the code for `l1l2_algorithm` (used for variable selection), `l1l2_kcv` (used for model selection with kcv or loo), and `l1l2_pred` (used for prediction on a test set).

  For more information regarding the parameters and the usage of those functions, use the help.

  Finally, you may want to have a look at the file `l1l2_demo_simple.m` (in the directory `demo_scripts`) for an example of the analysis. You can use it as a basis for writing your own code.

- **I.E** *(Prediction and selection)* Considering the elastic net regularization, we want to study the sensitivity of training and test errors with respect to the choice of regularization parameters and with respect to the number of relevant features of the solution, by changing one parameter at a time. To that end, you should try to pick some parameters, or run them in a loop, by exploiting the code in `l1l2_demo_simple.m`. Namely, study what happens as
  - **I.E1** ... you change the regularization parameter `L1_par` associated with the ℓ_1-norm.
  - **I.E2** ... you change (increase or decrease) the regularization parameter `L2_par` associated with the ℓ_2-norm.
    Hint: Try the following fixed parameters: 20 points of dimension 100, with 15 relevant features, a noise level equal to 1 and `L1_par`=0.1.
  - **IE.3** ... the size of the training set grows (this is not the same as generating different training sets of increasing size!).
    Hint: Try the same parameters as above, with `L2_par`=0.
  - **I.E4** ... the amount of noise on the generated data grows (the test set is generated with the same parameters as the training set).
- **I.F** *(Large p, small n)* Perform experiments similar to those above but now changing p (dimensionality of the points), n (number of training points) and s (number of relevant variables). In particular, look at how do the results behave when $p \gg n$, depending whether $s < n$ holds or not (e.g. try $n = 80$ and $p = 300$). Try to identify different regimes.

PART II: Intermediate

Crescendo: Data standardization

From now on do not use the GUI.

Import the classification dataset `part3-data.mat` given in the `data` folder at the root of the repo, which contains two matrices: `X` and `Y`. For this you can just double click on it in the file explorer, or otherwise write

```
temp=load('../../data/part3-data.mat'); X=temp.X; Y=temp.Y;
```

Here $X \in \mathbb{R}^{n \times p}$ is a matrix containing n points in $\mathbb{R}^p$, and $Y \in \{-1; +1\}^n$ is generated from $Y = X\beta$, where $\beta \in \mathbb{R}^p$ is an s-sparse vector, meaning that it has s nonzero components, with s being small.

- **II.A** You do not know how many relevant features generated this data. Try an estimate this number s, by using `l1l2_learn`, according to your observations in part I.

- **II.B** An other way to try to estimate s is to measure the correlation between the columns of X and Y. Indeed, the zero coefficients in β will ignore the corresponding columns in X while generating Y. To do so, you might use
  `c=abs(corr(X,Y)); stairs(c);`
  If you do not see which columns of X are more correlated with Y, you can use `[~,I]=sort(c);` to sort and identify the more relevant features. You should at this point have a precise idea of what is s. Can you also identify which are these s features?
- **II.C** Use again `l1l2_learn` and tune the sparsity parameter `L1_par` so that it selects only s features (s being your sparsity estimate from the previous question). Look at which are the selected features in your solution. Do they correspond to the ones you identified in II.B? If they do not, can you figure out why does that happen?