MAP654I Practical introduction to Machine Learning
Practical Session 2 Regression

Rémi Flamary

The objective of this practical session is to manipulate and understand the machine learning problems and methods discussed in the course. The practical session will be done in Python 3 and it is strongly recommended to have a working Anaconda environment. The different sections of the session should be implemented in Jupyter notebooks and it is strongly recommended to take notes in the notebook during the session.

The individual reports (with figures and discussions) for the session will be uploaded on moodle in python notebook format. It is expected to have a working code (reproducible figures) and a short discussion in markdown format for all the results obtained in the practical session. Note that ALL plotting and visualization questions in the practical session require a **critical discussion** of the visualized result that will be graded.

The end of the report must contain a personal discussion about the session (what was hard to understand and implement, how you would do it next time, what was new, discussion of relation with the course, personal discussion about how to use these tools in a professional setting, ...).

Importing libraries

In this practical session we will use Numpy/Scipy Python libraries for handling numerical data and Matplotlib for plotting them.

```
import numpy as np
import pylab as pl
import scipy as sp
```

Note that a more standard import for matplotlib is `import matplotlib.pyplot as plt`. You can use the name you want for the plotting module but if you choose `plt` you need to replace it in the function names given in the following (`plt.plot` instead of `pl.plot`). We will also need to have access to some functions and classes in the Scikit-learn toolbox.

1 Dataset description

In this session you will use a subset of the dataset from the BCI (for Brain Computer Interface) competition IV. A detailed description and the raw data is available on the competition website [1].

Dataset You will use the dataset 4 from the competition. The objective of this data is to predict the finger flexion of a subject using only ElectroCorticoGram (ECoG) signals recorded simultaneously on the brain of the subject. In this session you will use the data from subject 3 and predict the flexion of its thumb along time.

Experimental protocol The three subjects from the dataset were epileptic patients with ECoG sensors in place for medical reasons. The subjects were equipped with gloves measuring the flexion of each of their fingers simultaneously with the ECoG signals. The subjects were asked to move their fingers with visual cue. The signal is a 10 minutes recording at 1000Hz of 64 electrodes.

[1] Competition website:http://www.bbci.de/competition/iv/

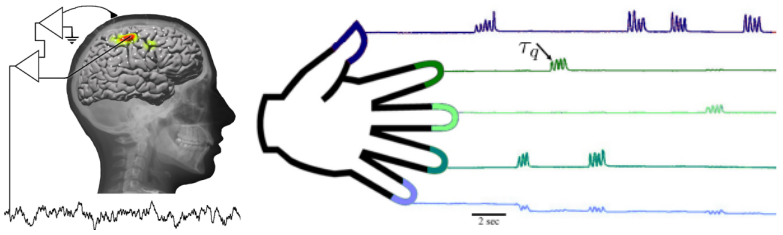

Fig. 1: Dataset 4 from BCI competition IV. It contains measured ECoG signals and the corresponding finger flexion on the subjects.

Data pre-processing The ECoG signal was filtered with a Savitsky-Golay band-passed filter that has been shown to work well on similar applications. The signal is then sub-sampled in order to obtain a sampling of $Fe = 50$Hz. The target signal (finger flexion) is also sub-sampled to achieve the same sampling. Finally we only keep temporal samples where the finger is detected to be in movement.

The file `ECoG.npz` contains the following variables:

Xall Matrix of $\mathbb{R}^{n \times d}$ containing n examples of d variables.

Yall Vector of $\mathbb{R}^n$ containing values to predict.

Fe Sampling frequency for the two signals.

Prediction performance evaluation The performance of a given model can be computed either using the average squared error :

$$MSE = \frac{1}{n} \sum_{i}^{n} (y_i - \hat{y}_i)^2 \tag{1}$$

that should obviously be the smallest possible. It can be computd with a single line of python or with the function `sklearn.metrics.mean_squared_error`.

Another measure of performance is the R^2 coefficient of determination that can be computed with `sklearn.metrics.r2_score` and is equal for linear model to the square of the correlation coefficient :

$$R = \frac{Cov(\mathbf{Y}, \hat{\mathbf{Y}})}{\sqrt{Cov(\mathbf{Y}, \mathbf{Y}) Cov(\hat{\mathbf{Y}}, \hat{\mathbf{Y}})}} \tag{2}$$

where $Cov(.,.)$ is the covariance between the two vectors. The correlation coefficient was used in the competition for the final ranking of the candidates methods. Both are normalized performance measures that will be equal to 1 for a perfect prediction of 0 for a random prediction.

2 Data visualization and pre-processing

- Load the dataset in memory and plot the EcoG signals and the finger movement on the same figure (`np.load`,`pl.plot`,`pl.subplot`).
- Visualize the data as a scatter plot where the color of the samples is the value to predict y and their position are variables 45 and 48 of $Xall$ (`pl.scatter` with `Yall` controlling the color).
- Split the data in a training and a testing set with $n = 1000$ training samples and the remaining samples as testing samples (`x=Xall[:n,:]` to select the n first lines in a matrix).

3 Least Squares regression (LS)

- Create the training matrix $\tilde{X}$ as defined in the course by concatenating a columns of 1s to the training samples (`np.concatenate,np.ones`).
- Estimate the LS parameters on the training data by solving the linear problem. Store those parameters as a vector $\mathbf{w}$ and a bias b (`np.dot,np.linalg.solve`).
- Predict the finger flexion of the subject on the training and test sets. Plot the predictions along with the true y (`pl.plot`). Is the prediction good on test data?
- Compute the performance as MSE and R^2 on training and test data. Conclusions?
- Estimate a linear regression with `sklearn.linear_model.LinearRegression` and check that the estimated coefficients are the same as those estimated above (`model.coef_,model.intercept_`).

4 Ridge regression

- Estimate the ridge predictor on the training data with $\lambda = 1$ and compute the prediction performances on training and test set (`sklearn.linear_model.Ridge`).
- Estimate the Ridge and, predict the labels and compute the performance for 100 values of the regularization parameter λ from 10^{-3} to 10^5. Plot the evolution of the MSE on training and testing data as a function of λ. Also plot the evolution of the linear parameters as a function of λ (in the same figure with subplot).
- Select the value of λ having the best performance on test data and estimate a Ridge model with this λ (`np.argmin`).
- Predict the finger flexion of the subject on the training and test sets. Visualize with a scatter plot the true values VS the predictions (`pl.scatter`).
- Plot the predictions along with the true y (`pl.plot`) as a function of time. Is the prediction good on test data?
- Compute the performance as MSE and R^2 on training and test data. Compare with LS estimator and discuss.
- Interpret the classifiers $\mathbf{w}$ for both LS and Ridge estimator by plotting their values and their absolute values (`pl.stem`). Do they show the same important variables (large magnitude)?

5 Variable selection with the Lasso

- Estimate the Lasso predictor on the training data with default value for the regularization parameter (`sklearn.linear_model.Lasso`).
- Predict and compute the prediction performances on training and test set, look at the estimated linear model. Notice anything?
- Estimate the Lasso and, predict the labels and compute the performance for 100 values of the regularization parameter λ from 10^{-3} to 10^5. (`np.logspace,for i,reg in enumerate(lst_reg):`). Plot the evolution of the MSE on training and testing data as a function of λ.
- Select the value of λ having the best performance on test data and estimate a Lasso model with this λ (`np.argmin`).

- Predict the finger flexion of the subject on the training and test sets. Plot the predictions along with the true y (`pl.plot`) as a function of time.
- Interpret the classifiers $\mathbf{w}$ for the Lasso estimator by plotting their values and their absolute values (`pl.stem`). What are the two most important variables?
- How many features were selected by the lasso. What would the reduction of the number of electrodes?

6 Nonlinear regression

For the non-linear regressors, `sklearn.ensemble.RandomForestRegressor`, `sklearn.svm.SVR`, `sklearn.neural_network.MLPClassifier` do the following:

- Fit the model with the default parameters and compute its prediction performance. Is it better than LS (that has no parameters), Ridge or Lasso estimators?
- Validate (by hand, or with a loop) some of the important parameters in order to maximize the performance on test. Is the performance very dependent on the parameters?
- Plot the predictions along with the true y (`pl.plot`) as a function of time. Is the regression method good and why?

7 Final comparison of the performances

- Collect the test performances for all methods investigated above in a table (in a dataframe and printing it for instance). Which methods work the best in practice?
- What are the most interpretable models?
- Which model is best from a medical/practical perspective?
- Do we need non-linearity in this application?
- Is validation on the test data a good practice? What would you do if you need to provide a model to a client for prediction in production?