Introduction Supervised Learning
IMA205

Pietro Gori

Deadline: Upload the answers and the code to the site pédagogique before the 20th of February 2019 (23h59).

Python environment

This practical session is based on Python 3. If you want to use your own computer, I suggest that you use Anaconda with the following libraries:

- Python = 3.6.8
- numpy version = 1.15.4
- scikit-learn = 0.20.0
- matplotlib = 2.2.3
- nilearn = 0.5.0 (to install with pip, not with conda)

If you are using the computers at Télécom, it is already installed. Before doing the TP, please open a terminal and type:

- export PATH=/cal/softs/anaconda/anaconda3/bin:$PATH

In this way, you will use the Python version installed in anaconda 3. After that, if you want, you can create a new environment with:

- conda create –name TP2-IMA205 python=3.6.8 numpy=1.15.4 scikit-learn=0.20.0 matplotlib=2.2.3
- source activate TP2-IMA205
- pip install nilearn (for the second part)

In the same terminal, type *spyder* which is a scientific environment specifically conceived for Python (it is similar to Matlab...). Of course, if you prefer, you can use other environments like PyCharm.

NB: If you have a lot of free space, you can also use a local version of anaconda 3 where you can install other libraries or versions. Please copy the one of Télécom to your home directory by doing:

- $/cal/softs/anaconda/anaconda3/bin/conda\ create\ -n\ my\_root\ --clone="/cal/softs/anaconda/anaconda3"$

Then, you will be able to create a new environment with the libraries/version you prefer: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html.

Emotion Recognition based on facial landmarks

We will use 400 images of the FEI dataset (https://fei.edu.br/~cet/facedatabase.html) to recognize (i.e. classify) the emotion of a person by analyzing 68 facial landmarks (already estimated and placed). You can find an example of the 68 facial landmarks in Fig.1. We will focus on two emotions: neutral and happy. So this can be seen as a binary classification problem.

Please open the file 'main_FEI.py' , execute each part and answer the questions. The code does the following:

1. Import all packages and functions
2. Read data
3. Shuffle the data. Why in your opinion ?
4. Plot all landmarks aligned using a Generalized Procrustes analysis (GPA). This has been seen in IMA204, please see https://en.wikipedia.org/wiki/Generalized_Procrustes_analysis for more information.
5. For each configuration, we compute the distance from the average configuration and use it as classification feature. We scale the data such that each feature will have average equal to 0 and unit variance. Why is this important in your opinion ? After that, we split the data into training and test sets and we use a LDA as classification algorithm. Look at the resulting confusion matrix. Are the results good ? Why ?
6. (Optional) If you want, you can try to use directly the position of the landmarks as features
7. We use cross-validation with all methods seen today (you will discover Quadratic Discriminant Analysis (QDA) at the end of this TP...). Are the results satisfactory ? For all methods ?
8. We change features and compute the distances between all combinations of landmarks. Are the results better than before ? Do you get any warning ? Why in your opinion ?
9. Which technique could you use to overcome the warning ? Why ? Comment the results.
10. The last part is about a clever selection of the features. Maybe we could simply choose a smaller set of landmarks... write a set of landmarks where it is written $select\_land = XXXXXXXX$ like $select\_land = [1, 10, 45]$. Which ones would you choose ?
11. The last part is about testing this new combination of features and plotting the images where the prediction of the model was wrong. Do you think that images were correctly labeled ? Remember that one should always check the so-called "ground truth"...

Alzheimer prediction using PET signals

In the second part of this practical session, you will write a Python script (similarly to the previous one) to predict if a patient has Alzheimer's disease based on PET signals averaged over a set of 69 ROIs (Region of Interest, see Fig.2). We will use data of 60 subjects from the ADNI database (http://adni.loni.usc.edu/). Data has been pre-processed by the Clinica team using the procedure explained here: http://www.clinica.run/doc/Pipelines/PET_Volume/.

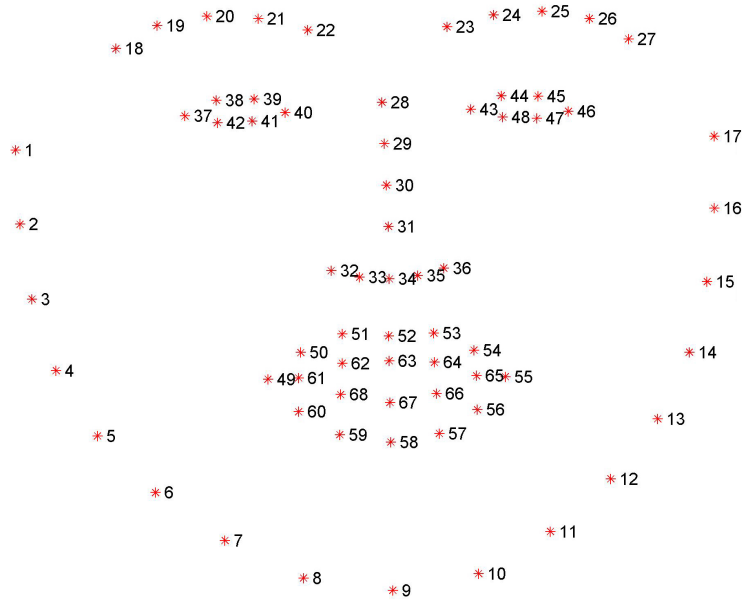


Figure 1: 68 facial landmarks

You will find the data in the file: 'Data_ADNI.npz' where X is a matrix containing the averaged PET signals (each row is a subject and each column a feature) and y is a vector containing the diagnosis (0 for controls and 1 for Alzheimer's patients).

Please load the data with:

```
dim=2 # dimension
Working_directory="./"
with np.load(Working_directory +'Data_ADNI.npz') as data:
    X = data['X'] # original landmarks
    y = data['y'] # landmarks after GPA
    labels = data['labels']

N,M = X.shape # number subjects and ROIs
class_names = ["control","alzheimer"] # y=0, y=1
```

You will also find the atlas used to define the ROIs in 'atlas.nii'. You can plot it using *nilearn version 0.5.0* and the following code:

```
from nilearn import plotting
plotting.plot_roi('PATH_TO_atlas.nii', title="Atlas")
plotting.show()
```

Create a function (i.e. 'main_ADNI.py') which discriminates between Alzhimer's patients and controls. Use the functions presented in 'main_FEI.py' and scikit-learn (https://scikit-learn.org/stable/index.html).

(Optional) Try the following cases and answer the questions:

- Use fewer subjects in the training set and see what happens to the training and test errors (try it several times, do the results change ?)
- Do you need to use all ROIs ? Do you have a problem of collinearity ? Knowing that hippocampus and medial temporal lobe atrophy are highly correlated with Alzheimer's disease, what could you do ? Hint: look at the labels using:

```
for i in range(N):
    print(labels[i,:])
```

- Among the methods seen during the lecture, which one is the best ?

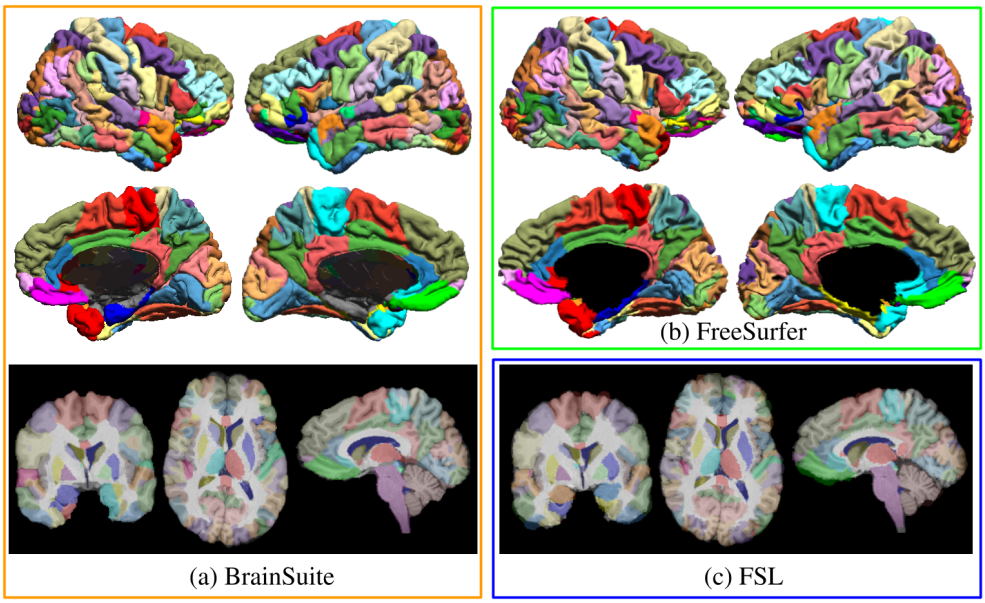


Figure 2: Examples of atlas with their corresponding ROIs subdivision. Image taken from BrainSuite website.

Theoretical questions

OLS

We have seen that the OLS estimator is equal to $\beta^* = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$ which can be rewritten as $\beta^* = H\mathbf{y}$. Let $\tilde{\beta} = C\mathbf{y}$ be another linear unbiased estimator of β where C is a $d \times n$ matrix, e.g. $C = H + D$ where D is a non-zero matrix .

- Demonstrate that OLS is the estimator will the smallest variance: compute $\mathbf{E}[\tilde{\beta}]$ and $\text{Var}(\tilde{\beta}) = \mathbf{E}[(\tilde{\beta} - \mathbf{E}[\tilde{\beta}])(\tilde{\beta} - \mathbf{E}[\tilde{\beta}])^T]$ and show when and why $\text{Var}(\beta^*) < \text{Var}(\tilde{\beta})$. Which assumption of OLS do we need to use ?

Ridge regression

Suppose that both $\mathbf{y}$ and the columns of $\mathbf{x}$ are centered ($\mathbf{y}_c$ and $\mathbf{x}_c$) so that we do not need the intercept β_0. In this case, the matrix $\mathbf{x}_c$ has d (rather than d+1) columns. We can thus write the criterion for ridge regression as:

$$\beta^*_{ridge} = \arg\min_{\beta}(\mathbf{y}_c - \mathbf{x}_c\beta)^T(\mathbf{y}_c - \mathbf{x}_c\beta) + \lambda||\beta||_2^2 \tag{1}$$

- Show that the estimator of ridge regression is biased (that is $\mathbf{E}[\beta^*_{ridge}] \neq \beta$).

- Recall that the SVD decomposition is $\mathbf{x}_c = UDV^T$. Write down by hand the solution β^*_{ridge} using the SVD decomposition. When is it useful using this decomposition ? Hint: do you need to invert a matrix ?

- Remember that $\text{Var}(\beta^*_{OLS}) = \sigma^2(\mathbf{x}^T\mathbf{x})^{-1}$. Show that $\text{Var}(\beta^*_{OLS}) \geq \text{Var}(\beta^*_{ridge})$.
- When λ increases what happens to the bias and to the variance ? Hint: Compute MSE $= \mathbf{E}[(y_0 - x_0^T\beta^*_{ridge})^2]$ at the test point (x_0, y_0) with $y_0 = x_0^T\beta + \epsilon_0$ being the true model and $x_0^T\beta^*_{ridge}$ the ridge estimate.
- Show that $\beta^*_{ridge} = \frac{\beta^*_{OLS}}{1+\lambda}$ when $\mathbf{x}_c^T\mathbf{x}_c = I_d$
- From the KKT equations we have that $\lambda(g(\beta) - t) = 0$ which means $\lambda(||\beta||_2^2 - t) = 0$ (See Eq. 25 in the slides of the lecture). Assuming $\mathbf{x}_c^T\mathbf{x}_c = I_d$ and using the previou result ($\beta^*_{ridge} = \frac{\beta^*_{OLS}}{1+\lambda}$) find the relationship between λ and t. What happens when $\lambda = 0$? Or when $\lambda = \infty$?
- Looking at Eq.36 and Fig.5 of the slides, comment the shrinkage effect of Ridge and LASSO. Do they shrink in the same way the coefficients β^*_{OLS} ?

Elastic Net (Optional)

Using the previous notation, we can also combine Ridge and Lasso in the so-called Elastic Net regularization :

$$\beta^*_{ElNet} = \arg\min_{\beta}(\mathbf{y}_c - \mathbf{x}_c\beta)^T(\mathbf{y}_c - \mathbf{x}_c\beta) + \lambda_2||\beta||_2^2 + \lambda_1||\beta||_1 \tag{2}$$

Calling $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$, solving the previous Eq. is equivalent to:

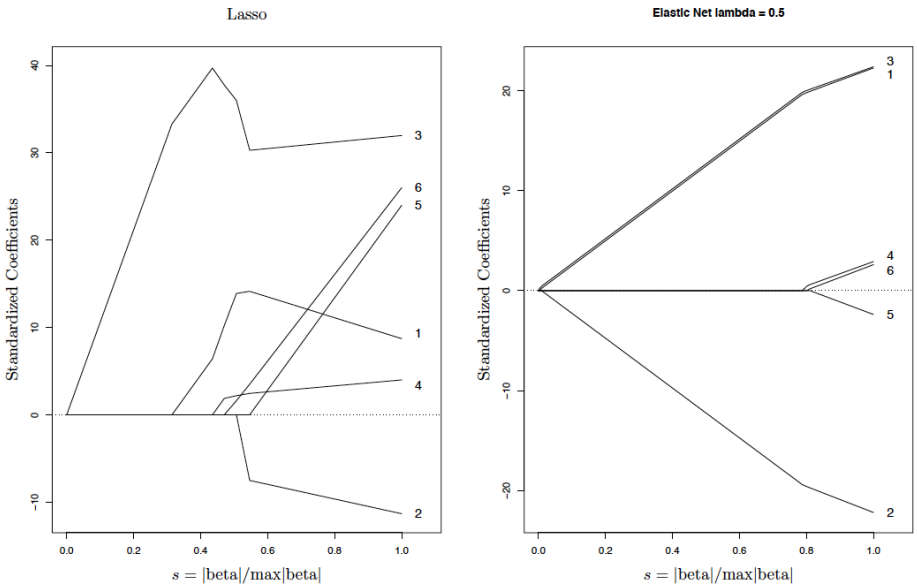
$$\beta^*_{ElNet} = \arg\min_{\beta}(\mathbf{y}_c - \mathbf{x}_c\beta)^T(\mathbf{y}_c - \mathbf{x}_c\beta) + \lambda\left[\alpha(\sum_{j=1}^{d}\beta_j^2) + (1-\alpha)(\sum_{j=1}^{d}|\beta_j|)\right] \tag{3}$$

- This regularization overcomes some of the limitations of the Lasso, notably:
  - If $d > N$ Lasso can select at most N variables $\rightarrow$ ElNet removes this limitation
  - If a group of variables are highly correlated, Lasso randomly selects only one variable $\rightarrow$ with ElNet correlated variables have a similar value (grouped)
  - Lasso solution paths tend to vary quite drastically $\rightarrow$ ElNet regularizes the paths
  - If $N > d$ and there is high correlation between the variables, Ridge tends to have a better performance in prediction $\rightarrow$ ElNet combines Ridge and Lasso to have better (or similar) prediction accuracy with less (or more grouped) variables
- Compute by hand the solution of Eq.2 supposing that $\mathbf{x}_c^T\mathbf{x}_c = I_d$ and show that the solution is: $\beta^*_{ElNet} = \frac{(\beta^*_{OLS})_j \pm \frac{\lambda_1}{2}}{1+\lambda_2}$

LDA

In LDA we assume that all classes have the same covariance matrix Σ.

- What happens if we assume that each class k has its own covariance matrix Σ_k? Compute $f^*(x_j)$ given a training set $\mathcal{T}$ where x_j is a test sample.
- Show that the solution is a quadratic discriminant function. This is called Quadratic Discriminant Analysis (QDA).


Lasso
Standardized Coefficients
$s = |\text{beta}|/\max|\text{beta}|$
3
6
5
1
4
2
Elastic Net lambda = 0.5
Standardized Coefficients
$s = |\text{beta}|/\max|\text{beta}|$
3
1
4
6
5
2