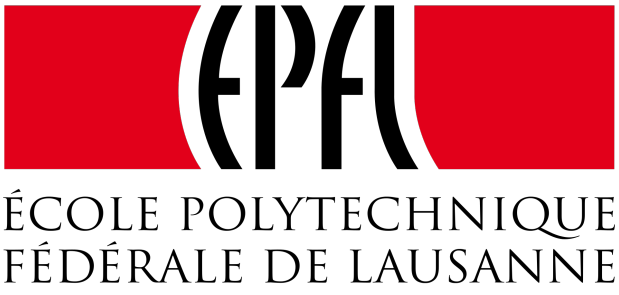
Data Analysis and Model Classification

Mini-project 3

Authors:
Julie Laurent
Alice Leydier
Yara-Maria Proust

Group ID : 2

December 4, 2017



1 Introduction

Understanding how the brain works and sends electrical signals to control the body is a subject in the center of many attentions. Indeed, it could allow the elaboration of prosthetic devices, which could help disabled patients. To do so, one needs to record neuronal activity with electrodes, and decode it to predict a certain movement.In this experiment, a monkey was moving a pole with its arm, while recording of firing rates, corresponding to arm movements on the pole, was performed by means of an invasive brain machine interface (BMI), via implantation of multiple electrodes (multi-unit recording). The collected neuronal data were then used to control an external robotic arm via the developed decoder. From this setup, a dataset *Data.mat*, containing the computed firing rates of all 48 identified neurons, was obtained.This dataset included a matrix *Data* and two "label" vectors *PosX* and *PosY*, corresponding to the Cartesian coordinates of the monkey's wrist in XY plane (actual hand trajectories). In *Data*, the features (spike rate of neurons, meaning the number of action potentials) were represented in its columns, while its lines indicated the time (every 50 ms). For each time t, the variable *Data* denoted the activity of 48 neurons recorded in a 1-second long window with a sampling frequency of 20 Hz, leading to 960 features (48*20). In total, there were 12862 time points (meaning a total recording of ∼10-11 minutes). *PosX* and *PosY* were used as basis to design a regressor model where the input was neuronal activity. The brain decoder would then be used to control the robotic arm, bypassing any muscular activity. In order to choose the best final model, several regression methods were explored: 1^{st} and 2^{nd} order regressions without any regularization, as well as LASSO and Elastic Nets regression methods.

2 Methods

Dataset Partitioning and Principal Component Analysis (PCA) Firstly, the dataset was partitioned into a training and testing set (training: 70%, test: 30%). A normalization was made on the training set only, as the testing (unseen) set should not be modified, and a PCA was computed on this normalized training set. Use of PCA (unsupervised technique) allowed the projection of the original data into a new features space defined by Principal Components (PCs). The obtained uncorrelated features (PCs) corresponded to linear combinations of the original ones and maximized the variance of the original data. The testing set was then transfered into the new PCs space as following:

$$testSet_{PCA} = \frac{testSet_{original} - \mu}{\sigma} * coeff$$

where μ and σ are respectively the mean and the standard deviation of the training set while $coeff$ represents the PC coefficients obtained with the PCA on the training set. The cumulative variance with respect to the total variance was computed and the number of PCs needed to explain 90% of the variance was finally found.

Linear regression on all PCs Using these PCs as new features, a linear regression (supervised technique) was performed on the training set only. This method attempts to predict the value of an output variable y (dependent variable), given the value of one or more input (independent) variables $x = \{x_1, x_2, ..., x_N\}$, such that the error between Y and f(X) is minimized, where X is a set of samples given, Y their known respective output values and $f : y \leftarrow f(X)$ the mapping function. In this case, Y corresponded to *PosX* and *PosY* and X to the samples with all PCs. *PosX* and *PosY* could be identified as the "labels" of the experiment. The difference with classification methods is that Y and $y = f(X)$ are continuous variables. To assess the performance of this univariate regression model, in predicting *PosX* and *PosY*, the linear regressor was trained on the training set composed of all the PCs. Then, training and testing errors were computed, by calculating the mean-squared error (MSE) between Y and the regressed vectors, both on the train and test partitions. A plot of the real vectors and the regressed ones was done.

Second order regression on all PCs and comparison with the linear regressor The same process was done, using a 2^{nd} order polynomial regressor on all PCs. Again, the training and testing errors were computed, and a comparison was done between the two types of regressors.

Gradual implementation of PCs After data partition, normalization and PCA, a loop gradually including PCs was implemented, when training the first and 2^{nd} order regressors. PCs were included by steps of 50, to accelerate the computation. At each step, performance of these regressors on the training and testing set was computed, and a plot was done with the obtained training and testing errors, when estimating *PosX* and *PosY*, with respect to the number of PCs used. The optimal number of PCs was assessed. A comparison between the number of PCs found with the cumulative variance or with this method was done.

Standard regression For the remaining parts of the project, only 5% of the samples were used for training and the rest for testing, in order to clearly see the effect of regularization when the data is scarce. Here, linear regression with an intercept was computed without PCA, such that $y = f(X) = X \cdot \beta + \beta_0$, with y the output labels, X the training data, β the vector of regression coefficients and β_0 the intercept (bias). The test error (MSE on test set) was then calculated.

LASSO regression method LASSO regression method was then computed on the training data, by adding a L_1 regularization constraint as following:

$$\min_{\beta} = ||y - X\beta - \beta_0||_2 \quad s.t. \quad \lambda||\beta||_1 \leq 1$$

To select the *hyperparameter* λ, a 10-fold cross-validation (CV) was performed. The several λ tested were a vector of 15 logarithmically spaced points between decades $1e^{-10}$ and 1. The CV MSE was plotted for each λ, and for the λ value corresponding to the best MSE, the corresponding β and *intercept* were used to regress the test data. The regressed and original position vectors were plotted and the test MSE was computed.

Elastic Nets regression method As LASSO optimizes sparsity, it only selects one of several features when groups of correlated features are formed. To counteract this limitation, a second constrain L_2 norm can be added, as proposed in the Elastic Nets method. The two constraints are weighted by an additional factor $0 < \alpha \leq 1$:

$$c(\beta) = \alpha||\beta||_1 + \frac{1-\alpha}{2}||\beta||_2^2$$

Elastic Nets was applied on the training data, with $\alpha = 0.5$, and a 10-fold CV was performed to select the *hyperparameter* λ (same as before). Again, the test data was regressed with the β and *intercept* corresponding to the optimal λ (smallest MSE). Original and regressed position vectors were plotted and the test error was computed. A single training-test split was done, varying the α linearly from 0.1 to 1, with a step of 0.1. For each α, the best λ was chosen thanks to a CV and the associated MSE was stored. Then, the optimal couple of *hyperparameters* λ and α was selected, by choosing the smallest MSE obtained. Once this optimal model found, its *hyperparameters* were used to regress the test data.

Models comparison Finally, the best model between LASSO and Elastic Nets regularization was chosen for the train set consisting in 5% of the data, and the importance of regularization was assessed. In order to compare regularized regression with the linear and 2^{nd} order regressions after PCA done in the first part, the test error of the Elastic Nets with α optimization was computed using a train partition of 70%.

3 Results

Dataset Partitioning and PCA It could be observed that 741 PCs were enough to explain 90% of the total variance of the data.

Linear Regression on all PCs Using the linear regressor, a training and testing error of respectively $2.8724e^{-4}$ and $6.2801e^{-4}$ were obtained when estimating the $PosX$ vector, and respectively $1.6720e^{-4}$ and $3.1847e^{-4}$ when estimating $PosY$.

Fig.1 showed that the original data was not perfectly represented by the obtained $PosX$ and $PosY$ estimation by regression.

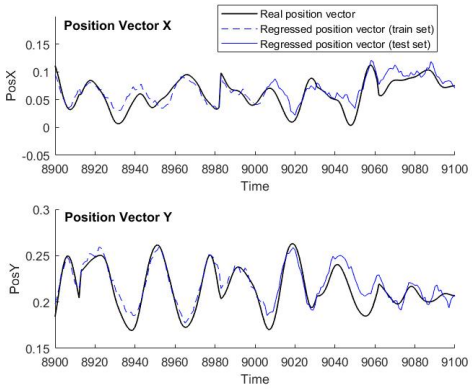


Figure 1: Linear regression on the train (dashed blue line) and test (plain blue line) sets, compared to the original data (black line). The top figure corresponds to the position vector X and the bottom figure to the position vector Y.

Second Order Regression on all PCs and Comparison with Linear Regression Using the 2^{nd} order polynomial regressor, a training and testing errors of respectively $2.2404e^{-4}$ and $6.7067e^{-4}$ were obtained when estimating the $PosX$ vector. For $PosY$ vector, the training and testing errors were $1.2921e^{-4}$ and $4.9193e^{-4}$. The training errors were thus smaller with a 2^{nd} order regression compared to linear regression, but the testing errors were larger.

Looking at Fig.2, more curvatures could be seen for the 2^{nd} order regression: the obtained curve was less smooth.

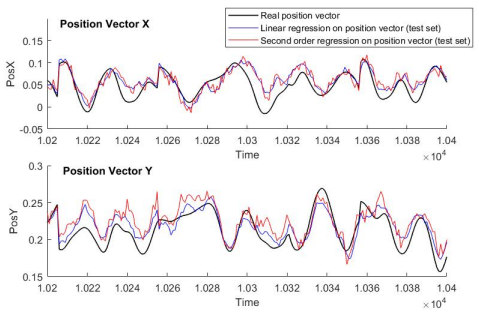


Figure 2: Comparison between linear (blue) and second order regression (red) on the test sets, compared to the original data (black). The top figure corresponds to the position vector X and the bottom figure to the position vector Y.

Gradual Implementation of PCs Looking at Fig.3, it could be seen that adding features decreased training errors for both regressions and both position vectors. The testing errors remained stable with the linear regression, whereas they increased with the number of PCs with a 2^{nd} order regressor, for both position vectors.

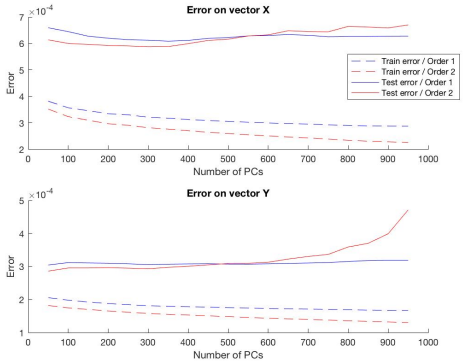


Figure 3: Comparison between the training (dashed line) and testing (plain line) errors, after a linear (blue) and second order (red) regression, with a different number of PCs (features), added 50 by 50. The top figure corresponds to the position vector X and the bottom figure to the position vector Y.

The smallest test errors for $PosX$ was obtained with 350 PCs for linear regression (test error of $6.0862e^{-4}$) and 300 PCs for the 2^{nd} order ($5.8737e^{-4}$). For $PosY$, both orders obtained minimal test errors with 50 PCs (order 1: $3.0448e^{-4}$, order 2: $2.8593e^{-4}$).

Standard Regression The train and test errors for $PosX$ were found to be respectively 2.4605e^{-32} and 8.8000e^{-3}. For $PosY$, they were

respectively 6.1663e^{-32} and 7.1000e^{-3}. As the training errors were very close to zero, it was not possible to distinguish the regressed training set from the real position vectors on Fig.4. The test errors were greater than the training errors. By comparing them to the test errors obtained when regressed with PCA (first order), they were larger by an order of magnitude 1. By looking at Fig.4, it could be seen regression on test set did not correspond at all to the initial position vectors.

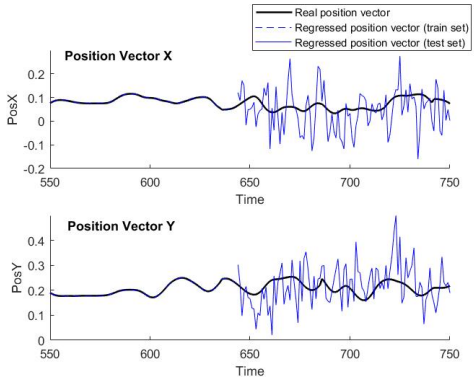


Figure 4: Standard linear regression (without PCA) on the train (dashed blue line) and test (plain blue line) sets, compared to the original data (black line). The top figure corresponds to the position vector X and the bottom figure to the position vector Y.

LASSO Regression Method The obtained number of non-zero β weights for each λ for $PosX$ and $PosY$ can be seen on Fig.5. The number of non-zero weights decreased with the increase of λ.

The CV MSE obtained was minimal for the 10^{th} λ value ($\lambda = 2.6827e^{-4}$) for both position vectors. Overall, the MSE was quite stable (between 1.80e^{-4} and 2.70e^{-4}) until the 11^{th} λ, but then increased rapidly until ∼7-7.5e^{-4}.

The test error obtained after regression with the optimal λ and corresponding β was 6.8829e^{-4} for $PosX$ and 3.7789e^{-4} for $PosY$. On Fig.6, the regression on the test set was close to the initial position vectors ($PosX$ and $PosY$) but still remained not optimal.

Elastic Nets Regression Method The obtained number of non-zero β weights for each λ for $PosX$ and $PosY$ can be seen on Fig.5. Again, the number of non-zero weights decreased with the increase of λ, but more slowly than with the LASSO method.

λ	1.00 e-10	5.18 e-10	2.68 e-8	1.39 e-8	7.20 e-8	3.73 e-7	1.93 e-6	1.00 e-5	5.18 e-5	2.68 e-4	1.40 e-3	7.2e -3	3.73 e-2	1.93 e-1	1.00
X LASSO	960	960	960	960	960	960	950	892	711	401	111	6	0	0	0
Y LASSO	960	960	960	960	960	959	943	876	659	382	139	9	0	0	0
X Elastic	960	960	960	960	960	960	948	919	793	525	238	32	0	0	0
Y Elastic	960	960	960	960	960	959	951	919	776	504	224	48	0	0	0

Figure 5: Non-zero β weights for all λ, using the LASSO and the Elastic Nets techniques, on both position vectors. In the elastic regularization done, $\alpha = 0.5$.

The CV MSE was also minimal for the 10^{th} λ value ($\lambda = 2.6827$e^{-4}), for both position vectors. The MSE had the same behavior across the λ values than with the LASSO regression.

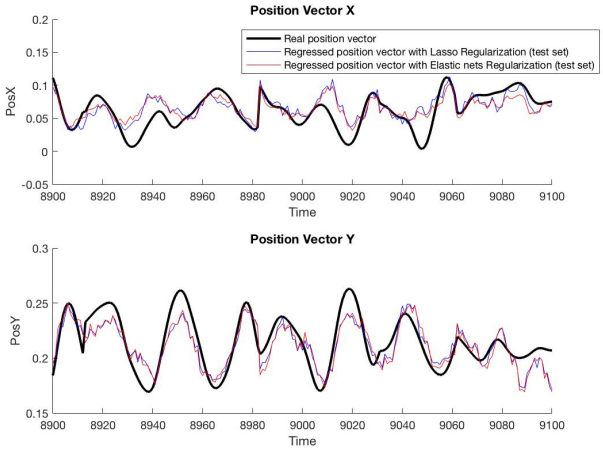


Figure 6: L_1 regularized regression (LASSO, in blue) and L_2 regularized regression (Elastic Nets, in red) on the test sets, compared to the original position vectors (in black) X (top graph) and Y (bottom graph).

The test error obtained with the optimal λ and corresponding β were 7.1852e^{-4} for $PosX$ and 3.9708e^{-4} for $PosY$. These errors were higher than with LASSO regularization. As before with LASSO, the regression seen on Fig.6 with the Elastic Nets regularization on the test set was not yet optimal.

Parameters Optimization When optimizing the couple of *hyperparameters* λ and α, the minimal MSE were found with the 10^{th} λ ($\lambda = 2.6827$e^{-4}) and $\alpha = 1$ for both $PosX$ and $PosY$.

The test errors obtained after regression with the optimal *hyperparameters* α and λ were

$6.8209e^{-4}$ for $PosX$ and $3.7955e^{-4}$ for $PosY$. These errors were smaller than the ones obtained with the Elastic Nets regression without α optimization, and were very close to the errors found with LASSO regression.

	PosX	PosY	Hyperparameters	Train partition
Linear / All PCs	6.2801e-4	3.1847e-4	-	70%
2nd Order / All PCs	6.7067e-4	4.9193e-4	-	70%
Linear / Gradual PCs	6.0862e-4	3.0448e-4	#PC X : 350 ; #PC Y : 50	70%
2nd Order / Gradual PCs	5.8737e-4	2.8593e-4	#PC X : 300 ; #PC Y : 50	70%
Standard	8.8000e-3	7.1000e-3	-	5%
LASSO	6.8829e-4	3.7789e-4	$\alpha_x = \alpha_y = 1$; $\lambda_x = \lambda_y = 2.6827e\text{-}4$	5%
Elastic Nets / α fixed	7.1852e-4	3.9708e-4	$\alpha_x = \alpha_y = 0.5$; $\lambda_x = \lambda_y = 2.6827e\text{-}4$	5%
Elastic Nets / α optimized	6.8209e-4	3.7955e-4	$\alpha_x = \alpha_y = 1$; $\lambda_x = \lambda_y = 2.6827e\text{-}4$	5%
Elastic Nets / α optimized	6.1162e-4	3.0646e-4	$\alpha_x = \alpha_y = 0.4$; $\lambda_x = \lambda_y = 2.6827e\text{-}4$	70%

Figure 7: Table of test errors obtained for each model.

Models Comparison The Elastic Nets regression with α optimization was done with a training set representing 70% of the samples, and test errors of $6.1162e^{-4}$ for $PosX$ and of $3.0646e^{-4}$ for $PosY$ were obtained.

4 Discussion

Linear and Second Order Regressions on all PCs As expected, test errors were larger than training ones for both linear and 2^{nd} order regressors. Indeed, when a regressor is built on one set, it has a high performance when tested on the same set, as it was tuned for this dataset. On the contrary, when it is tested on an *unseen* set, the performance obtained is lower. The final interest was in the test error, as it allows to predict how the model responds to an unseen dataset. Looking at the goodness of the fit for the linear regression graphs, the result obtained was good on the train set but the estimation was still not ideal on the test set. This is why a 2^{nd} order regressor was computed. The fact that training errors were smaller with a 2^{nd} order regression, was expected. Indeed, with the increase of the order of regression, the model was better tuned for the training set. Therefore, the fact that testing errors showed an increase with a 2^{nd} order regression might be due to over-fitting, as more parameters are used for higher order regressors. Indeed, the number of parameters was equal to 961 in the linear regression (intercept + 960 PCs), and 1921 for the 2^{nd} order (intercept + 960 PCs + 960 PCs). Certainly, the number of parameters obtained with the 2^{nd} order regression was too close to the number of samples. Thus, with an increased order of regression, the testing error is expected to increase because of over-fitting while the training error should decrease. To eliminate the undesired effect of over-fitting, a reduced number of PCs (and thus less parameters) had to be used.

Linear and Second Order Regressions with Gradual Implementation of PCs It has to be noted that no comparison based on the plot was done between the linear and 2^{nd} order regressions, as they were plotted in function of the PCs rather than in function of the parameters. The fact that test error increased with the number of PCs for the 2^{nd} order could be explained by the fact that adding PCs increased strongly the number of parameters used (as explained before) and thus the model was more prone to over-fitting. In the case of linear regression, the stability of the performance could be explained by a small complexity (few parameters) compared to the number of samples, so an increase of the number of PCs does not affect the error too much. The linear regressor should be chosen as it was less complex, more stable and returned smaller test errors when using enough PCs. As the least complexity was desired, a model leading to a minimal number of parameters should be chosen. The 2^{nd} order regression would lead to a slightly smaller error (0.96 times smaller) than with the linear for $PosX$, but increases the complexity (a.k.a. the number of parameters) by 1.71 times, even though more PCs were used for the linear regression. For $PosY$, the 2^{nd} order regression enables an error by 0.94 times smaller than with the linear, but raises the complexity by 1.98 times. Thus, the linear regressor was used for the rest of the project, as the decrease in test errors was negligible compared to the increase of complexity brought by

using the 2^{nd} order regressor. As 50 PCs are needed for $PosY$ and 350 for $PosX$, two different models should thus be used. The number of chosen features with this method was smaller than with the cumulative variance method. A number of 350 and 50 features would explain less than 90% of the variance, but the computation would be faster and the test error did not decrease a lot with more PCs involved.

Standard Regression As the training set was very small (only 5% of the data), the performance of the prediction on the test set was low (high error). Indeed, in this case, there was a number of 961 parameters (intercept + 960 features), which was higher than the number of training samples (643). Therefore, the model built was not robust. This high number of parameters compared to the number of samples of the training set could explain the over-fitting seen on the training set. A nested CV should thus be done, but was not performed, to better see the effect of regularization on over-fitting. As applying PCA did not show a significant reduction in the number of useful features (PCs) needed to obtain 90% understanding of the data, it was decided not to use this method.

LASSO and Elastic Nets Regularization By using the LASSO regularization, weights that contribute more were identified, and some features were removed, allowing to control the complexity of the regressor, by reducing the number of parameters. This allowed to prevent over-fitting. As seen, the λ was optimal for an intermediate value, as when too big, all the weights were equal to 0, leading to a big error, while when too small, no feature was removed, leading to over-fitting, as seen before with standard regression. This hypotheses seemed to be confirmed by the fact that the best λ obtained for both training samples was the first one allowing to obtain less parameters than training samples. Indeed, with the 9^{th} λ, 712 parameters (intercept + 711 non-zero β weights) were obtained for $PosX$ and 660 (intercept + 659 non-zero β weights) for $PosY$. Therefore, this LASSO regularization seemed to be a good way to avoid over-fitting. The fact that the error was higher with the Elastic Nets compared to the LASSO regularization could be due to the fact that less weights were equal to zero, which might have lead to more over-fitting. It is also possible that the error was not improved because the features were not correlated (as shown by the fact that PCA did not reduce the number of needed features explaining sufficient variance of the data), or because the couple of *hyperparameters* λ and α was not optimal. A single training-test split was thus done to optimize α.

Parameters Optimization The best model, with $\alpha = 1$, was equivalent to LASSO regularization. The same λ than the one obtained only with LASSO was thus found. This seemed logic as there were very few training samples (5%), meaning the most feature weights had to be set to 0 in order to get more samples than parameters (Elastic Nets adds a constraint limiting the number of possible weights to set to 0). The very small differences found on the test errors between the two LASSO models could be explained by the fact that a 10-fold CV was used when calculating the λ and thus when calculating the β weights. Also, with this method, the same model could be used for $PosX$ and $PosY$.

Models Comparison With a training set corresponding to 5% of the samples, it was observed that regularization was absolutely necessary as it reduced the test error by one order of magnitude. To compare this Elastic Nets regression with α optimization with the first and 2^{nd} order regressions with PCA done at the beginning, it was necessary to use the same train split for all models (train set corresponding to 70% of the data, which is also a more plausible data split). In this case, no nested CV was needed as there were enough samples compared to the number of parameters. The fact that the test error obtained with the univariate linear regression using PCA was smaller than the error obtained with the optimized Elastic Nets regression showed that the linear regression with PCA should be chosen when there is no over-fitting (enough samples). A regularization should thus be used only in case of over-fitting.