ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Mini-Project II

EE-516

Data Analysis and Model Classification

Cécile Crapart, Gabriele Gambardella, Eleonora Martinelli

Lausanne, 9th December 2018

Contents

1 Introduction

Regression is a supervised learning technique which defines a function mapping the behavior of continuous dependent variables starting from a set of independent variables with their known output values. The data here investigated were collected during an experiment in which a monkey had to move a pole while multiple electrodes (multi-unit recordings) were recording the neurons activity patterns in his brain. The position of the pole along the x and y axis was considered to be the actual arm trajectory resulting from the neuronal activity (*labels*). So, on the basis of the provided dataset, this work aims at building a regression model able to optimally predict the monkey's intended movements from the registered neurons' firing rates. The decoded brain signals will then be used to control the trajectory of the robotic arm in the x-y space, allowing to bypass any muscular activity.

2 Methods

The aim of the work is to optimally predict the robotic arm trajectory required to fit the monkey's intended movement. To do so, monkey's brain signal were recorded with a multi-unit set-up and sampling frequency of 20Hz (which means 1 sample was recorded each 50ms) and collected in a matrix called Data. The data for one *sample* corresponds to the firing rates of 48 neurons (or "neuronal assemblies"), computed for each 50ms time subwindow (20 subwindows *48 neuronal assemblies = 960 *features* per sample). Consecutive samples are shifted in time by 50ms. Moreover, the cartesian coordinate X and Y of monkey's wrist arm position at each time subwindow was recorded through the position of the pole. The resulting vectors, PosX and PosY, are considered to constitute the actual arm trajectory (*labels*) that we aim to predict. Note that the optimal regression method to predict the trajectory along the two axes may differ; that is why two regression methods were investigated.

2.1 Preliminary operations

Dataset partitioning The splitting of the dataset had firstly to be defined: it has been decided to put 60% of the initial samples in the train set, 20% in the validation set and the remaining 20% in the test set. It has been decided to put 60% of the initial samples in the train set to prevent overfitting, which can be caused by a too small train set. The original dataset is composed by 12862 samples and 960 features. As the number of samples is 13 times superior to the number of features, it was considered safe to perform a single train-validation-test splitting instead of cross validation. This also means that a small standard deviation is assumed on the performance values, without the need of further statistics to prove significance. Moreover, because the dataset is non-stationary, the samples used in this regression problem are correlated in time so the chronological order had to be preserved while splitting (the samples in each set are kept in chronological order and not randomly mixed).

Data Normalization In multi-unit recordings, some neuronal assemblies are identified based on neurons activation patterns, and these can be representative of the activity of a varying number of neurons. To overcome the problem of having different magnitudes for different features, the dataset was thus normalized before starting the optimization of the regression method. The training set was normalized with the inbuilt MATLAB function *zscore*. For validation and test set, the function zscore could not be used: in fact, in a real situation, the test and the validation set would be totally unseen, thus it would be impossible to normalize them using their own mean and standard deviation, which would be unknown. For this reason, we subtracted to each element of the validation and test set the mean of the train set, and divided this for the standard deviation of the train set.

2.2 Regression methods

Two regression methods were investigated: PCA followed by regression and Elastic Nets. In both cases the model was trained on the training set and the hyperparameters were optimized on the validation set. Finally, the performance of the chosen model was estimated on the test set. Both the analyzed methods include a dimensionality reduction step. As previously mentioned, 960 features are available to predict two variables, PosX and PosY. However, the higher is the dimensionality, the higher is the risk of overfitting and the higher is the complexity of the model, which might result in unnecessary high computational time and cost. Those drawbacks can be avoided by reducing the number of features considered. To do so, one has to decide which features and how many of them to remove. In the first case, PCA has been used to reduce dimensionality; in the second case, Elastic Nets. In both cases, regression was performed to predict the value of PosX and PosY (output dependant variables, y), given the neuron firing rates contained in Data (input independent variables, x). The mapping function $y = f(x)$ was determined through error minimization between y and f(x). The MATLAB function **immse** was used to evaluate the performance of the regression: the function computed the mean-squared error (MSE) between the **y** vector (PosX or PosY) and the regressed one (Data*regression coefficients). In both cases, the model selection rule used to optimize the hyper-parameters of the regression model was choosing the model which gave the lowest MSE on the validation set.

2.2.1 PCA and regression

Removing the redundancies in the input features of the dataset before performing regression is a way to address the dimensionality reduction problem and avoid overfitting. Indeed, in the given dataset, solving the total parameter matrix for the regression corresponds to an overdetermined system with either 7717 or 2573 equations (individual predictions for 60% or 20% of the dataset, respectively) and 960 unknowns. However, some equations provide a redundant information as they are linearly dependent. Firstly, the 20 firing rates considered per neurons are obviously correlated in time. Secondly, the activation pattern of two distinct neurons ("neuronal assembly") may be correlated and the firing rate of only one of these neurons would be sufficient to encode a relevant information about the intended movement. As a result, the 960 dimensional input vectors span a space of lower dimensions. Principal Component Analysis (PCA) is an efficient way to remove these redundancies. The 960-dimensional input vectors are projected into a new N-dimensional space (N < 960). Most of the information encoded in the original vectors is preserved because all features contribute to the computation of each principal component. This method was used to remove any small variations (mainly encoding noise) that are orthogonal to the manifold. For the training set, PCA coefficients and scores were obtained after normalization using the inbuilt **pca** function. The normalized validation and test sets were rotated in the new space by using the coefficients found by the PCA on the training set. Linear (first order) and second order regression models were used to predict PosX and PosY from Data: $y(x) = [[w_0 + w_1 * x]_{firstorder} + w_2 * x^2]_{secondorder}$. To do so, the MATLAB function **regress** was used with an intercept. The performance of each regressor was assessed as a function of the number of features used for training the regressor. The hyperparameters to be optimized in this case were the number of PCs used for training the regressor and the polynomial order of the regressor.

2.2.2 Elastic nets

Another way to address the issue of dimensionality is to use regularization techniques, which constraint the definition of the regression coefficients. The latter are driven close to zero (shrinkage) based on penalty coefficients in order to build more sparse and robust models. In the LASSO regularization, the L1 norm of the weight vector is constrained not to exceed $1/\lambda$. However, in case of correlated features, LASSO would only select one of them since it optimizes sparsity. So, using only LASSO is not suitable for this dataset which, as explained previously, contains correlated features. A second constrain can be added on the L2 norm to overcome this issue (Ridge Regularization): here the regularization terms are function of square of coefficients of parameters so that these can approach to zero but never become zero. The Elastic Nets are a combination of the L1 and L2 regularizations where the two constrains are weighted by the factor $0 < \alpha <= 1$ (when $\alpha = 1$, elastic nets reduce to LASSO). The hyperparameters to be optimized in this case were λ and α. For λ, a vector of 15 values equally logaritmically spaced was used (logspace(-10,0,15)), for α, the same was done with 10 values (logspace(-2,0,10)).

3 Results

3.1 Optimization on the validation set

The validation set was used to optimize the hyper-parameters presented in the Methods section. The most performant regressor was selected based on the lowest MSE. The performances of the three regressor types are presented in **Figure 1**, and the hyper-parameters used to reach these performance are presented in **Table 1**.

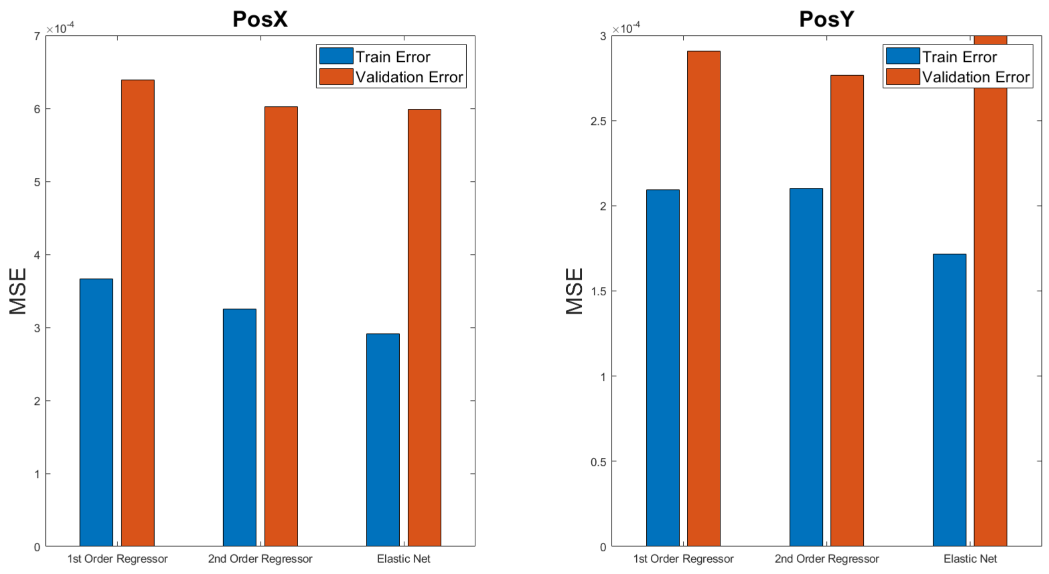

Figure 1: Performances (MSE) of three regression methods. The MSE was obtained with the hyper-paramters optimized on the validation set and presented in Table 1.

Table 1: Optimal hyperparameters on the validation set (iteration at which the minimal MSE was obtained for each type of regressor): number of features with respect to the order of the regressor, λ, α.

	PCA & Regression		**Elastic Nets**
Output variable	# of PCs (1st order)	# of PCs (2nd order)	Lambda, Alpha
PosX	100	100	λ=0.0014, α=0.0464
PosY	50	26	λ=0.0072, α=0.0167

Based on these results, the Elastic Net was selected to predict PosX (with the corresponding optimal λ and α, **Figure 2**). Before, the maximum number of PCs taken into account was higher, and the lowest MSE was reached by PCA followed by a 2nd order regressor. Nevertheless, overfitting was observed starting around 150 PCs taken into account. The number of features was then restricted to 100: in this case, the MSE obtained with PCA and second order regressor was higher than with Elastic Nets. Interestingly, for the optimal λ and α, 838 features have non zero weights while only 137 would be retained with only LASSO ($\alpha = 1$). This discrepancy is coherent with the previously mentioned high level of correlation in the dataset. PCA and second order regressor was selected to predict PosY based on the MSE (**Figure 1**), and only 26 features were required to achieve such performance (**Table 1**). This is largely inferior to the 733 features required to reach 90% of variance explained; moreover, the models starts overfitting from 27 features (**Figure 2**).

3.2 Final regression models

The final regression models for PosX and PosY were trained on the training set with the hyper-parameters optimized on the validation set. The test MSE for PosX was 4.6477*10^{-4} (Elastic Nets) and 2.4329*10^{-4} for PosY (26 PCs and second order regressor). Plotting the overlay of the real and the predicted PosX and PosY confirms the quality of the fit (**Figure 3**). The global trajectory of the arm is presented in **Figure 3** for a 1.25 s period of time. The fit is reasonably satisfying considering the scale used for the X and Y coordinate, and the low value of the MSE. The need for higher/lower precision of the prediction would depend on the task performed with the robotic arms.

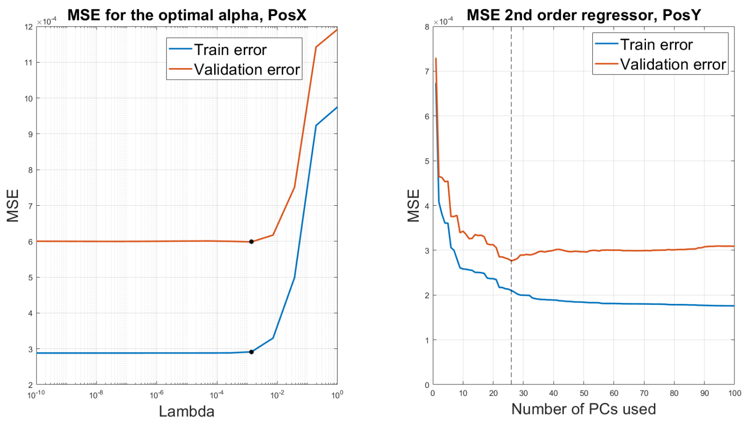

Figure 2: Performance (MSE) of the Elastic Nets regression method for PosX, with respect to the value of λ used, and of the second order regressor with respect to the number of PCs for PosY. For PosX, the optimal combination of α and λ is marked by a star, while for PosY the optimal number of PCs is shown by the line. Train and validation MSE are presented for both PosX and PosY.

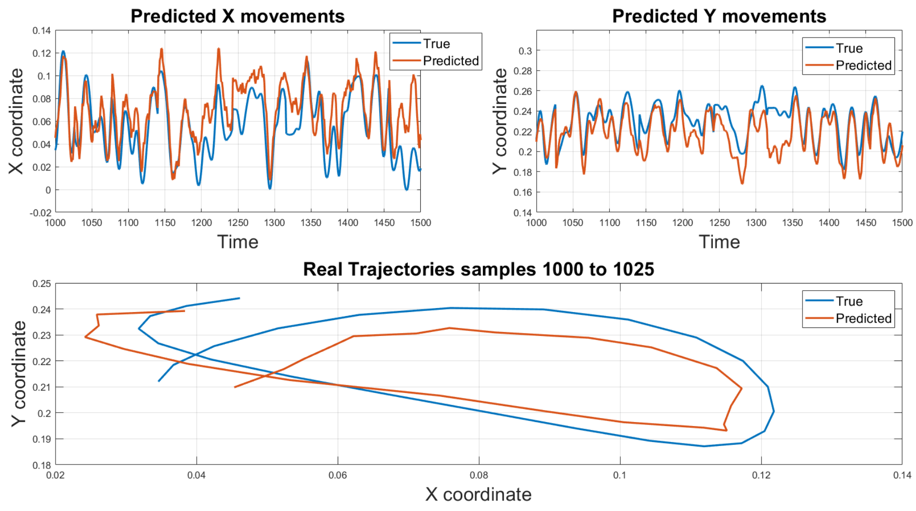

Figure 3: Predicted trajectory of PosX and PosY overlaid with the real trajectory, for samples from 1000 to 1500, corresponding to a 25 s time window. Global predicted and real arm trajectory for sample 1000 to sample 1025, corresponding to a 1.25 s time window.

4 Discussion

The percentage of samples included into train, validation and test set (60%, 20%, 20%) was fixed but could also be optimized. Regarding the features, 20 different firing rates during 50ms are considered for each neuron, that is 1 second before the actual time of the prediction of the arm position. Another way to reduce the dimensionality and simplify the model would be to optimize the number of 50ms time subwindow retained prior to a prediction, for example 12 (600 ms prior motion). Moreover, in this model we assume that the arm motion is function of neuron activity only. However, physiologically, the motor cortex and the muscles are related by feedback loops to provide information about the state following a movement. These proprioceptive inputs could be implemented to improve the model. Other coordinate frames than Cartesian could be used such as joint location, velocity and acceleration. The performances of the predictors in each of the coordinate frame could be compared to select the most relevant one. Finally, other regression methods could be used such as Bayesian linear regression or Moore-Penrose pseudoinverse method. This model could be of great interest in the field of neuroprosthesis as it helps to predict arm movement and bypass the damages in motor control system.