Data Analysis and Model Classification
Guidesheet VII: PCA and regression

Ruslan Aydarkhanov Bastien Orset Julien Rechenmann
Ricardo Chavarriaga José del R. Millán

November 20, 2017

Introduction

The purpose of this exercise is to get familiar with regression techniques.

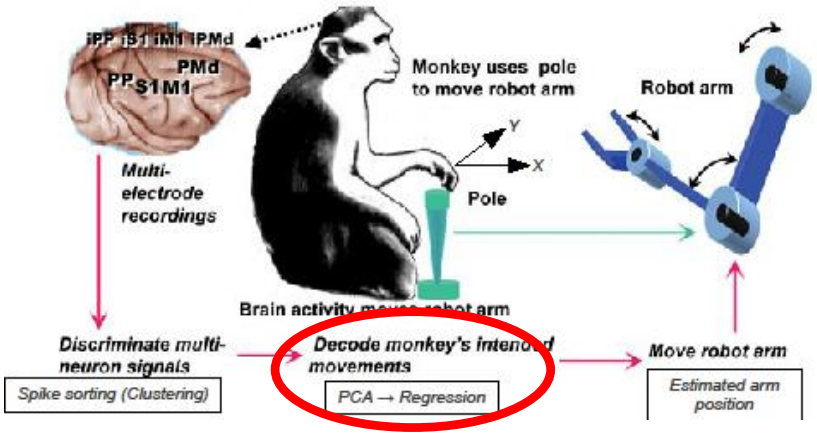

Figure 1: Objective of this guidesheet.

With regards to the Fig. 1, we will work on decoding monkey's intended movements.

Dataset partitioning and PCA

Partition your dataset (`Data`) in two different splits that will be used respectively for training and testing purposes. Take the first k% of the data for training and (100-k)% for testing (with k belonging to the interval [0,100]). Try, for example, with a 70% - 30% data partitioning. Then, apply PCA properly.

General questions

- On which data partition will you apply PCA for finding the coefficients of the PCs? The training partition
- Do you need to perform any transformation of your data before applying the PCA? YES: normalisation
- Will you reuse the coefficient of the PCs or estimate a new one on the other partition? Re-use of course! You never create a model (and its paramaters) out of the test data!

Regression

The purpose of this task is to provide an introduction to Linear Regression. This method attempts to predict the value of an output variable y (dependent variable), given the values of one or more input (independent) variables $\mathbf{x} = \{x_1, x_2, ..., x_N\}$. Therefore, it attempts to define a mapping $f : y \leftarrow f(\mathbf{x})$ such that the error between $\mathbf{Y}$ and $f(\mathbf{X})$ is minimized, where $\mathbf{X}$ is a set of samples given and $\mathbf{Y}$ is their known respective output values.

Regression is a supervised learning methodology, since the known values $\mathbf{Y}$ act as the "Labels" that will help define the mapping f. The difference w.r.t. classification methods is that y in this case is a continuous variable whose value we wish to predict given some input sample x we can measure, while in a classification problem, y is a discrete variable denoting the type of class we want to assign a sample $\mathbf{x}$.

For additional information on the concepts and equations used in regression, take a look at the supplementary material.

In this task, we will assess the performance of univariate and polynomial regression models for predicting two different variables `PosX` and `PosY` from `Data`. In particular, we will evaluate the results of the regression as a function of the number of features used for training the regressor and w.r.t. the order of the regressed polynomial.

General questions

- On which data partition will you train the regressor on?The training data
- On which feature matrix will you train the regressor on? On the original feature matrix (`Data`) or on the one projected on the PCs (`DataPCA`)? Why? DataPCA

Hands on Use the MATLAB function `b = regress(y(dataPartition),X(dataPartition,chosenFeatures))` to learn a regressor. `b` are the coefficients of the learned regressor, `y` is the target to be regressed and `X` is the data used for the regression.
Let's define `I = ones(size(y(dataPartition),1),1)` and `FM = FeatureMatrix(dataPartition,chosenFeatures)`.
For training a linear model we have to define `X` as:
`X = [ I FM ]` .
For learning a 2nd order regressor, `X` has to be defined as follows:
`X = [ I FM FM.^2 ]`
Similarly, for n-th order polynomials, the corresponding columns `FM.^n` have to be added to `X`.

- Using all the features of your chosen feature matrix, train a linear regressor for the `y = PosX(dataPartition)` data vector. Then, use the MATLAB function `immse(y,X*b)` to evaluate the performance of the regression (the function calculates the mean-squared error between the `y` vector and the regressed one `X*b`), both on the train and test partitions. Is there any difference between the errors on the train sets and those on the test sets? Why? DONE
- Repeat the previous step for the `PosY` data vector. DONE
- For both `PosX` and `PosY`, plot the real vectors and the regressed ones, both for the train and test partitions. You can compare the vectors using the MATLAB "Zoom in" plot tool and interpret the goodness of the fit. DONE
- Using all the features of your chosen feature matrix, train now a 2nd order polynomial regressor on your data and compare the results on the train and test sets to those obtained with the linear regression. Are there any differences? If yes, how do you interpret them? How do you think the regression will perform by increasing the order of the polynomial? DONE
- Implement now a loop to gradually include features when training your regressors.[1] Inside the loop, train a linear regressor and a 2nd order regressor for both `PosX` and `PosY`, and evaluate their performances on the train and test partitions. At the end of the training, plot the results of your

[1] Instead of including the features one by one, you can include features by steps of 50∼100 for accelerating the computation.

regressors (differentiating between train/test errors and polynomial order of the regressor) w.r.t. the number of features used. Which trend do you see when increasing the number of features used? How do you explain it?

The training error for second order regression is lower than first order independantly the number of features, but the testing error seems to increase more after 300 features for both x and y position, and is worst than first order testing error after 500 features. It seems that 300 features (corresponding to the lowest testing error) is a good number, since over-fitting seems to take place right after that number! HOW DO WE EXPLAIN IT: => OVERFITTING BITCH. The ratio between train and test error is the best @ circa 100-120 features*

- By looking at these plots, which number of features would you eventually use for regressing the provided data? Is this number comparable with the one found when looking at the variance explained by each feature in the PCA section of the exercise? What are the cons of choosing this number based on these results? Which technique learned during the course could be used to properly tune this hyperparameter?

* because we don't want to take a ratio below 0,5, that would mean that the testing error increases too much compared to the training, so even if overfitting occurs only at above 300 features, a good error ratio implies taking around 100 features, which compares to the number of features expressing more variance in the PCA (ADD FIGURE??)

The cons are:

The technique is cross-validation