Project Assignment
System Identification 2017

Logistics

This MATLAB-based project assignment is a compulsory part of the System Identification course in the Control Engineering B.Sc. program of the Technical University of Cluj-Napoca. It will be graded and the mark counts for 20% in the final grade of the course. The assignment is carried out in groups of two students, and should take around 20 hours per person to solve, depending on your experience with MATLAB. Each group will receive their own data sets. To receive them, form groups and send as soon as possible an e-mail to the lecturer Lucian Buşoniu at `lucian@busoniu.net`. Please, mention the names of the two members of your group and their e-mail addresses. The assignment must be worked out in the form of a short written report (in English, one report per group).

Please email the report **in PDF format** to the lecturer at the same address. Do not forget to include your names on the title page of the report. It is required to include in the report complete listings of the MATLAB code (functions and scripts) that you developed for solving the assignment problems. In addition to that, your code must also be sent as a ZIP file, with the same email in which you send the report. Use MATLAB version R2006b or higher and mention the version you used.

The deadline for the full report and code is **December 17th 2017, 24:00 (midnight) at the latest**. In addition, an intermediary version of the report and code with the first problem solution must be sent by **November 19th 2017, 24:00**. In case of delays, each newly entered day of delay results in a 2 point decrease in the maximum grade for the assignment (for instance, delivering the full report on December 19th at 00:10 AM leads to a maximum grade of 6 since the second day of delay has been entered). Note that it is strictly forbidden to take over results from other students, as well as copy text from online resources. Failing to obey this rule constitutes plagiarism and will lead to your solution being disqualified, which results in ineligibility for the exam.

The assignment consists of two problems, both using polynomial approximators. In the first problem, the polynomial is used to model the behavior of an unknown function. The second problem concerns data-driven black-box modeling of an unknown dynamical system.

Fitting an unknown function

A data set of input-output pairs is given, where the outputs are generated by an unknown, nonlinear but static function f. The outputs are corrupted by noise, which is assumed to be additive and zero-mean Gaussian. The function has two input variables and one output variable. You will have to develop a model for this function, using a polynomial approximator. A second data set generated using the same function is provided for validating the developed model. The two data sets will be given as a MATLAB data file, containing one structure for each set. The training data structure is named `id` and the validation data structure `val`. Each of these structures contains the following fields:

- A set of grid coordinates `X` for the inputs, where `X` is a cell array of two vectors, each vector `X{dim}` containing n grid points for input dimension `dim`.
- A set of corresponding outputs `Y`, a matrix of size $n \times n$, where `Y(i, j)` is equal to the value of f at point `(X{1}(i),X{2}(j))`.
- The same data is provided in a different, 'flattened' format: the inputs `Xflat`, a wide matrix containing the n^2 input points, one on each column, and the corresponding row vector of outputs `Yflat`.

The 'structured' variants X and Y are easier to use for creating graphs, while the 'flat' variants are more suitable to fitting and using the approximator.

Your polynomial approximator g should have a configurable degree m. For example, for the first few values of m, the approximator has the form:

$$\begin{aligned}
m = 1, \quad & \hat{g}(x) = [1, x_1, x_2] \cdot \theta = \theta_1 + \theta_2 x_1 + \theta_3 x_2 \\
m = 2, \quad & \hat{g}(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1 x_2] \cdot \theta = \theta_1 + \theta_2 x_1 + \theta_3 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \theta_6 x_1 x_2 \\
m = 3, \quad & \hat{g}(x) = [1, x_1, x_2, x_1^2, x_2^2, x_1^3, x_2^3, x_1 x_2, x_1^2 x_2, x_1 x_2^2] \cdot \theta
\end{aligned}$$

where the first two polynomials have been made explicit for clarity. To increase numerical stability, you may want to rescale the input variables x_1, x_2 from their arbitrary initial intervals $[a_1, b_1]$, $[a_2, b_2]$ to the interval $[-1, 1]$, in which case your code should handle this internally. You can find out the intervals from the range of the data.

Once the degree m has been chosen, model fitting consists of finding the optimal parameter vector θ so that $g(x)$ best matches $f(x)$ on the identification dataset, in a least-squares sense. This can be done with linear regression, keeping in mind that g is linear in the parameters (even though it is nonlinear in the variables x). Details can be found in the lectures, Part 3: *Mathematical Background*, see the linear regression sections. The regressors for the polynomial case here are the powers of x_1 and x_2, e.g. for $m = 2$ they are 1, x_1, x_2, x_1^2, x_2^2, $x_1 x_2$.

The **requirements** are given next. Program such an approximator of configurable degree m. Try to fit approximators of varying degrees, so as to obtain the most accurate one. Validation should always be performed on the different, validation dataset. Report the mean squared errors for both sets and show a representative plot for the fit on the training and the validation data sets (true values compared to approximator outputs). Discuss the results, including the choice of degree and the quality of the model fit on the two data sets, relating them to the discussion during lectures on model choice and overfitting in regression.

While MATLAB can automate polynomial regression, it is strongly recommended that you code the approximator and regression procedure on your own, because you will need the insight in the second part of the project, which does *not* have an off-the-shelf implementation in MATLAB. For some inspiration you can look at your solution to practical assignment 4, *Linear regression for function approximation*, and inside the function `rbfapprox` which was provided in the beginning of that lab.

Black-box system identification

Before working on this part of the project, you will need background on *linear* ARX models. Part 5 of the lectures, *ARX identification*, focuses on this, and is taught in lecture 6.

A dataset is given, measured on an unknown **dynamic system** with one input and one output. The order of the dynamics is not larger than three, and the dynamics may be nonlinear while the output may be affected by noise. Your task is to develop a black-box model for this system, using a polynomial, nonlinear ARX model. A second data set measured on the same system is provided for validating the developed model. The two data sets will be given in a MATLAB data file, with variables `id` and `val` containing the two sets as objects of type `iddata` from the System Identification toolbox. Recall that the input, output, and sampling time are available on fields `u, y, Ts` respectively. As a backup in case the system identification toolbox is not installed on the computer, `id_array` and `val_array` contain the same two datasets but now in an array format, with the structure: time values on the first column, input on the second column, and output on the last column.

Consider model orders *na*, *nb*, and delay *nk*, following the convention of the `arx` MATLAB function. Let the vector of delayed outputs and inputs be denoted by $d = [y(k-1), y(k-2), \ldots, y(k-na), u(k-$

$nk), u(k-nk-1), \ldots, u(k-nk-nb+1)]^T$. Then, the nonlinear ARX model has the structure:

$$\begin{aligned}\hat{y}(k) = &- p_1(d)y(k-1) - p_2(d)y(k-2) - \ldots - p_{na}(d)y(k-na) \\ & z_1(d)u(k-nk) + z_2(d)u(k-nk-1) + \ldots + z_{nb}(d)u(k-nk-nb+1)\end{aligned} \tag{1}$$

where each p_i and z_j is a polynomial of the variables in d. Note that these polynomials replace the constant coefficients of the linear ARX model, leading to a model that is nonlinear in the delayed variables d for any polynomial degrees greater than zero. If the polynomials all have degree zero, then the model reduces back to the linear one. The parameters of the model are the coefficients in each of these polynomials. For clarity, we exemplify the case when all the polynomials are of degree $m = 1$, and $na = nb = nk = 1$. Then, $d = [y(k-1), u(k-1)]^T$ and there are just two polynomials in the model, one for $y(k-1)$ and one for $u(k-1)$:

$$\hat{y}(k) = - p_1(d)y(k-1) + z_1(d)u(k-1)$$

Each of these polynomials is of degree 1, e.g. $p_1(d) = a_1 + a_2 y(k-1) + a_3 u(k-1)$, and similarly for $z_1(d)$. Therefore, written explicitly the model is:

$$\begin{aligned}\hat{y}(k) = &- [a_1 + a_2 y(k-1) + a_3 u(k-1)]y(k-1) + [b_1 + b_2 y(k-1) + b_3 u(k-1)]u(k-1) \\ = &- a_1 y(k-1) - a_2 y(k-1)^2 - a_3 u(k-1)y(k-1) \\ &+ b_1 u(k-1) + b_2 y(k-1)u(k-1) + b_3 u(k-1)^2\end{aligned} \tag{2}$$

Crucially, model (1) is still linear in the parameters (in the example (2), a_i and b_j) so linear regression can still be used to identify these parameters. However, if the model is left in the default form (1), there is a problem: some of the regressors are linearly dependent, which if unaddressed, would lead to singularities in the linear system of equations. To make the model well-conditioned, these multiple terms of the same form should be combined into one, with a single coefficient/parameter. For the example model (2), the product $y(k-1)u(k-1)$ appears twice, and these two terms should be combined:

$$\hat{y}(k) = -a_1 y(k-1) - a_2 y(k-1)^2 + b_1 u(k-1) + b_3 u(k-1)^2 + c y(k-1)u(k-1) \tag{3}$$

where $c = -a_3 + b_2$.

The **requirements** follow. Code a function that generates such an ARX model, for configurable model orders na, nb, delay nk, and polynomial degrees. For simplicity you can set all the polynomials to have degree m. Make sure to group the linearly dependent terms as explained above, to avoid numerical problems. Code also the linear regression procedure to identify the parameters, and the usage of the model on arbitrary input data. Note that the model should be usable in two modes:

- One-step-ahead prediction, which uses knowledge of the real delayed outputs of the system; in the example, we would apply (3) at step k with variables $y(k-1), u(k-1)$ on the right-hand side.
- Simulation, in which knowledge about the real outputs is not available, so we can only use previous outputs of the model itself; in the example we would replace $y(k-1)$ on the right-hand side of (3) by the previously simulated value $\hat{y}(k-1)$.

Identify such a nonlinear ARX model using the identification data, and validate it on the validation data. Choose carefully the model orders and the delay, as well as the polynomial degree. To reduce the search space you may take $na = nb$. Report the one-step-ahead prediction error, and the simulation error for both the identification and the validation sets (use the mean squared error). Show a representative plot for the fit on the training and the validation data sets, for both simulation and prediction. Discuss the results, including the quality of the model fit on the two data sets.

The presentation above is self-contained, but you may e.g. look at the following papers for additional technical insight:

1. H. Peng et al., *RBF-ARX model-based nonlinear system modeling and predictive control with application to a NOx decomposition process*, Control Engineering Practice 12, pages 191–203, 2007. Here the model is explained in Sections 2.1-2.2, and uses tunable radial basis functions instead of polynomials.

2. L. Ljung, *System Identification*, Wiley Encyclopedia of Electrical and Electronics Engineering, 2007. Available as technical report LiTH-ISY-R-2809. See Section 4 for nonlinear models, again mainly using basis functions.

Matlab programming and other remarks

Strive for a compact and elegant MATLAB code, avoid the use of loops (`for`, `while`, etc.) and `if-then-else` constructs where vector operations would be easier and more readable. Search for "vectorization" in the MATLAB help system for helpful tips on the proper MATLAB programming style. However, do not exaggerate with applying vectorization: if the code is clearer with loops or if statements, use them.

If you are less familiar with programming in MATLAB, the following pointers may help. Type `doc` at the command line to access the documentation. A good initial read is the *Getting Started with Matlab* node of the documentation. *Matrices and Arrays*, *Programming Basics*, and *Plotting Basics* are also useful.

It is preferable that each group brings their own laptop to the project classes, and always works on the same laptop for the project. This is to prevent incompatibilities between the MATLAB versions installed on your computers and the University computers. If this is not possible for you, please discuss with the lecturer.