Project Assignment
System Identification 2019-2020

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 30% in the final grade of the course (15% for part 1, and 15% for part 2). The assignment is carried out in groups of **three** 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 name and email address of each member of the group.

The assignment consists of two problems. In the first problem, a radial basis function approximator is used to model the behavior of an unknown, static function. The second problem concerns nonlinear ARX identification of an unknown dynamical system, using polynomial approximation. The evaluation is performed differently for the two parts, see each part for details. **Crucial rule:** it is strictly forbidden to copy code, text, or results from other students or from online resources. Automated tools are in place to check this, and there will be absolutely zero tolerance for copying: failing to obey this rule automatically and immediately leads to ineligibility for the exam. So, be extremely careful!

Part 1. 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, zero-mean, and normally distributed. The function has two input variables and one output variable. You will have to develop a model for this function. 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{1}`, `X{2}` containing P grid points for input dimension `dim`.
- A set of corresponding outputs `Y`, a matrix of size $P \times P$, where `Y(k1,k2)` is equal to the value of f at point (`X{1}(k1)`, `X{2}(k2)`).

We will create a Gaussian, radial basis function (RBF) approximator of f. For simplicity, we will place the RBFs on an equidistant $R \times R$ grid. (This is *not* the same grid as the grid of points above!) One single such RBF has the formula:

$$\phi_i(x) = \exp\left[-\frac{(x_1 - c_{i,1})^2}{b_1^2} - \frac{(x_2 - c_{i,2})^2}{b_2^2}\right]$$

where:

- $i = 1, \ldots, R^2$ is the index of the RBF on the grid. Note that the grid is two-dimensional so we need R^2 RBFs. For a good representation power, shape the grid so that there is an RBF in each corner.
- c_i is the center of RBF at index i. Note that $c_{i,1}$ gives the position on the first dimension and $c_{i,2}$ on the second.

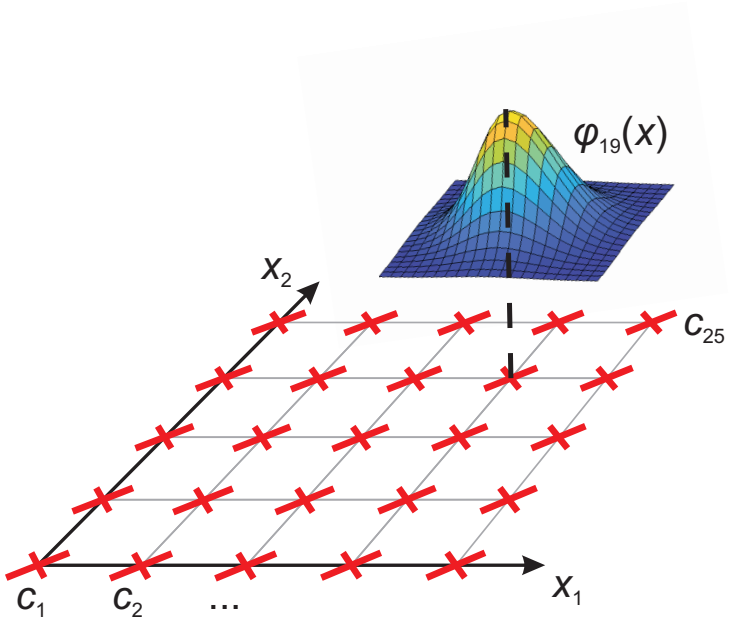

Figure 1: Illustration of a grid of RBFs with R=5. Each red X is a center, and one of the 25 RBFs is shown.

- b_1 is the radius of the RBF on dimension 1, and b_2 the radius on dimension 2. Note that b does not change with RBF index i, so all the RBFs have the same shape, only their centers differ (they move on the grid).

The overall approximator is then:

$$g(x) = \sum_{i=1}^{R^2} \phi_i(x)\theta_i$$

where $\theta = [\theta_1, \ldots, \theta_{R^2}]^T$ is the vector of parameters. See Figure 1 for an illustration of the RBF grid.

If we take the domain of the RBF grid to be the same as that of the data, then the only choices that need to be tuned are R and b. Once these choices have been made, 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, given the equation of the RBFs, it is nonlinear in the variables x). Details can be found in the lectures, Part 4: *Mathematical Background*, see the linear regression sections. The regressors are the RBFs.

The **requirements** are given next. Program such an RBF approximator with configurable grid size R and RBF radius b. Try to fit approximators with varying grid sizes R, so as to obtain the most accurate one. Start with b_1 and b_2 equal with the distances between two grid points on each of the two dimensions, to obtain a smooth approximator. Afterwards, optionally, you may also tune b. Validation (also for the purpose of comparing e.g. different values of R) 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 R and possibly b, as well as 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.

For some inspiration you can also look at your solution to practical assignment 4, *Linear regression for function approximation*.

Evaluation of part 1

Part 1 must be worked out in the form of a short written report (in English, one report per group), with the associated code. The deadline for the report and code is **November 10th 2019, 24:00**. In case of delays, each newly entered day of delay results in a 2 point decrease in the maximum grade (for instance, delivering the report on November 12th at 00:10 AM leads to a maximum grade of 6 since the second day of delay has been entered).

Please **pay attention and follow to the letter** the following rules for delivery. A uniformized, semi-automated processing of solutions is essential for efficient grading, and any deviation from the rules makes your submission require additional, manual processing time, which will likely be unavailable and may therefore mean that your solution cannot be graded!

- Your submission must consist of exactly two files, named exactly like this: LN1LN2LN3.pdf and LN1LN2LN3.zip, where LNi is the last (family) name of student "i" in your project group. For example: IonescuFarkasBonta.pdf and IonescuFarkasBonta.zip.
- The first file is the report, which must be delivered **in PDF format** (not DOCX or any other source format; PDF only). 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.
- The second file contains your code itself, sent separately as a **ZIP archive** (not RAR, and not 7Zip or other formats; classical ZIP only). Important: There should be no subdirectories in this archive, all the m-files must be top-level.
- These files will be submitted via a DropBox file request, the link to which will be supplied before the deadline. Do not submit multiple versions as these cannot be taken into account; only your first submission will be considered, so make sure it is complete and correct.
- The creation date of the file on DropBox is taken for the purpose of delay computation per the rules above.

Part 2. Nonlinear ARX identification

To work on this part of the project, you preferably need background on *linear* ARX models. Part 5 of the lectures, *ARX identification*, focuses on this.

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. Then, the nonlinear ARX model is:

$$\begin{aligned}\hat{y}(k) &= p(y(k-1),\ldots,y(k-na),u(k-nk),u(k-nk-1),\ldots,u(k-nk-nb+1)) \\ &= p(d(k))\end{aligned} \tag{1}$$

where the vector of delayed outputs and inputs is denoted by $d(k) = [y(k-1),\ldots,y(k-na),u(k-nk),u(k-nk-1),\ldots,u(k-nk-nb+1)]^T$, and p is a polynomial of degree m in these variables.

For instance, if $na = nb = nk = 1$, then $d = [y(k-1),u(k-1)]^T$, and if we take degree $m = 2$, we can write the polynomial explicitly as:

$$y(k) = ay(k-1) + bu(k-1) + cy(k-1)^2 + vu(k-1)^2 + wu(k-1)y(k-1) + z \tag{2}$$

where a, b, c, v, w, z are real coefficients, and the parameters of the model. Note that the model is nonlinear, since it contains squares and products of delayed variables (as opposed to the ARX model which would only contain the terms linear in $y(k-1)$ and $u(k-1)$). Crucially however, the model is still linear in the parameters so linear regression can still be used to identify these parameters.

Note that the linear ARX form is a special case of the general form (1), obtained by taking degree $m = 1$, which leads to:

$$\hat{y}(k) = ay(k-1) + bu(k-1) + c$$

and further imposing that the free term $c = 0$ (without this condition, the model would be called affine).

The **requirements** follow. Code a function that generates such an ARX model, for configurable model orders na, nb, delay nk, and polynomial degree m. 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 (2) 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 (2) 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 description 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.

Evaluation of part 2

Part 2 will be presented orally, and the code must also be submitted. The presentations will take place during week 13 of the semester, i.e. between 6 and 10 January. The exact schedule will be communicated a sufficient time in advance. Each group gets 20 minutes, out of which 10 are allocated for the presentation itself, and 10 for questions. All three members of the group must be there; excepting force majeure, any student who is absent will fail the project and thus be ineligibile for the exam. The grade will consider three aspects: the solution itself (code and results); presentation; and question answers. Questions will be targeted to differentiate the contribution of each group member, so each student will likely get a different grade.

The deadline for the presentation itself as well as the associated code is **December 20th 2019, 24:00**. In case of delays, each newly entered day of delay results in a 2 point decrease in the maximum grade, as for part 1 above.

Please follow the following rules for delivery, as accurately as for part 1.

- Your submission must consist of exactly two files, named exactly like this: LN1LN2LN3.pdf, .ppt, or .pptx, and LN1LN2LN3.zip, where LNi is the last (family) name of student "i" in your project group. For example: IonescuFarkasBonta.pdf and IonescuFarkasBonta.zip.
- The first file is the presentation; **PDF, PPT, or PPTX formats** are acceptable, but no other format.
- The second file contains your code itself, sent as a classical **ZIP archive** (no other formats accepted). Important: there should be no subdirectories in this archive, all the m-files must be top-level.
- These files will be submitted via a DropBox file request, the link to which will be supplied before the deadline. Do not submit multiple versions as these cannot be taken into account; only your first submission will be considered, so make sure it is complete and correct.
- The creation date of the file on DropBox is taken for the purpose of delay computation per the rules above.

Matlab programming and other remarks

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.

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.

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.