Simplex Regression, Manual

François Wahl

June 2, 2018

This software is a toolbox for performing simplex regression. It is based on the article entitled 'Simplex Regression: Multivariable Parametric Regression under Shape Constraints' which can be found here: 'https://hal.archives-ouvertes.fr/hal-01262601/document'.

The soft is written in Matlab®2016 and is being delivered to you AS IS, with no warranty as to its use or performance.

Unfortunately, bugs or difficulties may subsist and the author would be very much interested in any report: please e-mail to Francois.Wahl@univ-lyon1.fr.

system of functions

Currently, the code works

- in one dimension, for 3 systems of functions $\{f_j(x)\}_{j=0}^{J}$, namely
  1. monomials $f_j(x) = x^{d_j}$, where $d_0 = 0$ and $d_1, \cdots, d_J$ is a sequence of integers.
  2. exponentials $f_j(x) = \exp(d_j x)$, where $d_0 = 0$ and $d_1, \cdots, d_J$ are reals (not necessarily positive).
  3. power functions $f_j(x) = (x + \delta)^{d_j}$, where $d_0 = 0$ and $d_1, \cdots, d_J$ is a sequence of positive real numbers, and δ a small offset added to the f_j, to avoid any singularity at $x = 0$. If δ is too small, numerical difficulties can still occur around 0 when calculating the vertices of the osculating simplex.
- in dimension 2 and above for traditional monomials: in each dimension v, $f_{v,j_v}(x_v) = x_v^{j_v}$, where $0 \leq j_v \leq J_v$.

examples

A few predefined examples are proposed in one and two dimensions in the Matlab files 'FW_main180429.m' and 'FW_mainHDS180529.m'. They should be used as a basis for writing your own codes.

1. In one dimension, we compare our results to Hawkin's simulations. The points used in the article are stored in the file named 'hawkins.txt'. The equation studied by Hawkin is

   $$y = 4x(x-2)^2(x+0.5)^2(x^2+2) + \epsilon, \text{ where } \epsilon \sim N(0,1).$$

   y must be increasing with x.

2. The second example, still in dimension 1, mimics a sigmoid function impaired by noise: see 'xyb50.txt'. Again, y is monotonically increasing.

3. In dimension 2, a set of n points are randomly chosen in $[0,1] \times [0,1]$ verifying:

$$y = \frac{1}{2}\left(x_2 + \frac{1}{1.4\pi}\sin(2\pi x_2)\right)\left(1 + (2x_1 - 1)^3\right) + \epsilon, \text{ where } \epsilon \sim N(0, 0.1).$$

We have $\dfrac{\partial y}{\partial x_1} \geq 0$ and $\dfrac{\partial y}{\partial x_2} \geq 0$.

4. In dimensions 4, the file 'fw.xls' referred in 'FW_mainHDS180529.m' contains 80 experimental points, on which we must fit a second multivariable polynomial of degree 2 in four variables. Precisely, we have

$$y = \alpha_0 + \sum_{0<j<3} \alpha_j x_1^{j_1} x_2^{j_2} x_3^{j_3} x_4^{j_4}, \text{ where } j = j_1 + j_2 + j_3 + j_4.$$

y increases when x_1 decreases, x_2 increases, x_3 increases, x_4 decreases.

coding

Here is a sketch of the Matlab code for running a simulation in one dimension:

```
1  % example 1
2  ModelTensor.nvar=1;
3  % choose one of the possible typekernel: 'poly', 'exp', power'.
4  ModelTensor.typekernel={'poly'};
5  ModelTensor.exponents=[0,1,2,3,4,5];
6  % number of the variables for which monotony is required:
7  % +1=increasing, -1=decreasing
8  numvar=1;
9  signevar=+1;
10
11 % Display
12 verbose='yes';
13
14 % Names
15 Nom_x={'x_1'};
16 Nom_y={'y'};
17
18 % Load the data set (Xexp,yexp)
19 ...
20
21 % Make a translation of Xexp between [0,1]: Xexp --> X
22 ...
23
24 % Launch the algorithm
25 [Equation0, Equationi,
       Contraintes]=FW_CC180529(X,yexp,numvar,signevar,ModelTensor,verbose);
26
27 % Post-treatment
28 ...
```

Listing 1: **one dimension**

In more dimensions (for example 2), the lines 1 to 10 are changed in:

```
1  % example 5
2  ModelTensor.nvar=2;
3  ModelTensor.typekernel={'poly','poly'};
```

```
ModelTensor.exponents=[
     0,0 ; 0,1; 0,2; 0,3; 0,4
     1,0 ; 1,1; 1,2; 1,3; 1,4
     2,0 ; 2,1; 2,2; %2,3; 2,4
    3,0 ; 3,1; 3,2; 3,3; 3,4
 ];
% variables for which monotony is required :
% +1=increasing, -1=decreasing
 numvar=[1,2];
signevar=[+1,+1];
 ...
```

Listing 2: **in more than one dimension**

A few comments are necessary.

1. In the structure ModelTensor, the field 'exponents' contains a vector of exponents when the dimension of the input is one or a matrix of exponents otherwise.

   - The exponent 0 corresponds to the constant term.
   - In dimension v, each row of ModelTensor.exponents matches one of the f_j:

     $$f_j(x) = f_{j_1}(x_1) \cdots f_{j_v}(x_v).$$

     If the function to regress contains J terms, this matrix should have $J+1$ rows. The first column corresponds to the exponents of the first variable in f_{j_1}, the second to the exponents of the second variable in f_{j_2}, and so on. For example, the matrix of exponents $\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 2 & 3 \end{bmatrix}$ is the representation of the polynomial $\alpha_0 + \alpha_1 x_1 + \alpha_0 x_1^2 x_2^3$. In Matlab format, it gives: `ModelTensor.exponents=[0,0; 1,0; 2,3]`. Currently, the code has been only tested for polynomials in more than one dimension.
   - For function of one variable, `numvar=1` is mandotory. For more than one variable, the field 'numvar' indicates the montony requirements. For example, for 3 variables, `numvar=[2,3]` means that the resulting function should be monotonically increasing or decreasing with the second and third variable.
   - The field 'signevar' is a vector the same length as 'numvar' giving the sign of the requirement. $+1$ means increasing, -1 decreasing. For example, with the previous `numvar=[2,3]`, `signevar=[+1,-1]` says that the function is expected to be monotonically increasing with the second and decreasing with the third variable. Nothing is required for the first variable.

2. 'FW_CC180529' requires that the X values belong to $[0, 1]$.

3. `verbose='yes'` has the effect of producing messages following the progress of the optimization process. Here is an example:
   `iter=3,nb of active cnt =2, total nb of constraints =16`
   `iter=3, RMSEold=0.0305462->RMSEnew=0.0271041, RMSEold-RMSEnew=0.0034421.`
   This means that during the third iteration, 2 constraints were active over a total of 16, and that the root mean square error has decreased from `0.0305462` to `0.0271041`. The optimization ends when the difference between two successive RMSE is less than 10^{-6} or if the maximum number of iterations is reached.

text output

The code produce text results. We explain how it works with an example depicted in the following code: 30 points are randomly selected in Hawkins' data set. The function to fit is expressed as a sum of exponentials within exponents given line 4 of the code.

```
% example 2
ModelTensor.nvar=1;
ModelTensor.typekernel={'exp'};
ModelTensor.exponents=[0,0.5,1.2,2,2.1,2.5];
% variables for which monotony is required :
% +1=increasing, -1=decreasing
numvar=1;
signevar=+1;

% Choose the data set.
nomvrai='hawkins';

% Number of points to be selected:
n=30;

% Display
verbose='yes';
% Figures
graph='yes';

% Noms
Nom_x={'x_1','x_2'};
Nom_y={'y'};
```

Listing 3: **parameters for launching an exponential fit**

The output looks like:

```
---------------
CONSTRAINTS VERIFIED AT THE CORNERS
---------------
variable :  x_1
Number of Derivatives <0= 0 (Min= ) Number of Derivatives >0= 2 (Max=
186.88)
---------------
CONSTRAINTS VERIFIED AT THE EXPERIMENTAL POINTS
---------------
Nbr variables = 1, size of the experimental data base= 30, nb of con-
straints=26
variable :  x_1
Number of first derivatives <0= 0 (Min= ) Number of first derivatives >0=
30 (Max= 165.998)
RMSE 1.201940

RMSE without constraints 0.953212
```

In dimension 1, x varies from its minimum to its maximum value. This gives 2 corners in the domain.

26 constraints were needed. Since we require the resulting function to be monotone increasing on its definition domain, all the first derivatives should be non negative. This is observed at the experimental points.

The calculated RMSE for the constrained regression is here `RMSE=1.201940`. For the unconstrained regression `RMSE=0.953212` is obtained.

Note that it may happen that the minimum value for the first derivative be slightly negative (something like `1.24e-14`). This numerically can be considered as 0.

graphical display

The graphical displays are clearly part of post-treatment procedure. They are shown here as an illustration. They should be modified to fit different needs.

- Two optional figures are produced in dimension 1: the first one is a diagram of residuals, that is the residuals versus the experimental y. The second one is a graphical comparison of the fit.

  The example shown below (figure 1) comes from the same piece of code as the previous section.

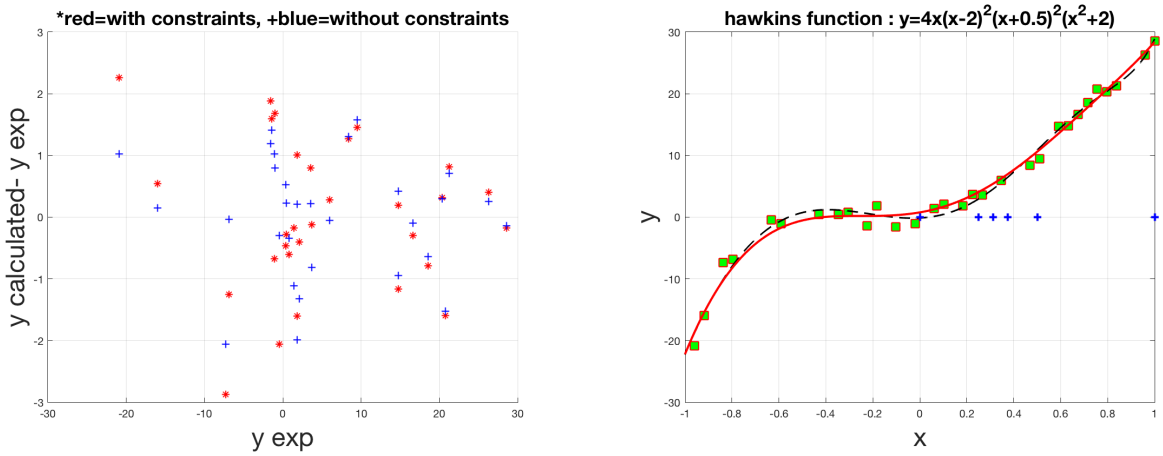

Figure 1: **exponential fit to Hawkins'data** Residue diagram on the left panel. On the right the plot compares the UNconstrained in dashed black and constrained regressions in red. The blue crosses indicate the limits in x of the simplexes generated during the optimization process.

- in dimension 2 and above, no more direct visualization is possible to check whether the obtained fit is correct or not. Instead we propose a kind of plot that we named 'octopus plot'. For a given x_0, we let vary successively only one coordinate. In the 'HDS' example, with four input variables, this gives rise to four curves. The figures of the article are reproduced below (figure 2).

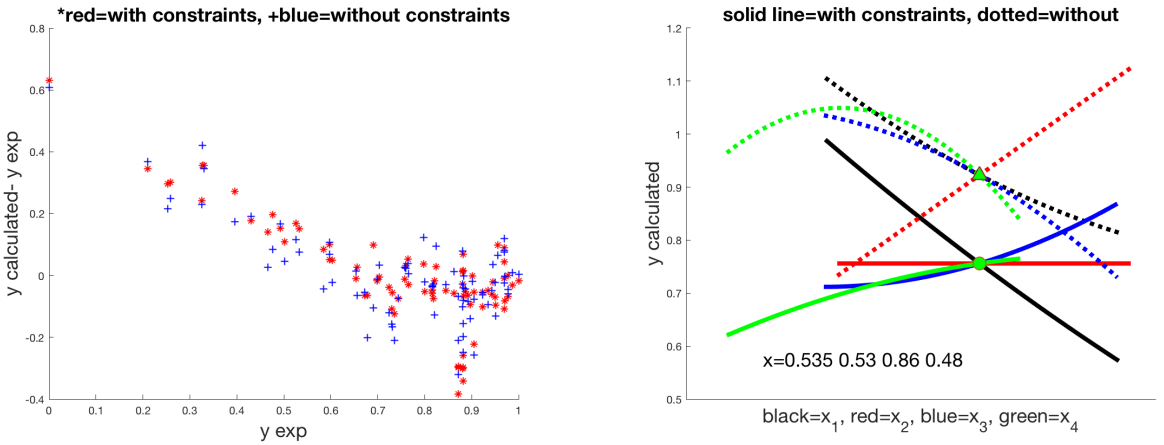

Figure 2: **polynomial fit to the data of HDS experiments** Residue diagram for the HDS data on the left panel. On the right the plot compares the UNconstrained and constrained regressions.

- additionally, in dimension 2, we propose the following diagrams to illustrate where the constraints have been calculated.

  Recall that on figure 1, on the right panel, the blue crosses shows the limits of the simplexes: each segment between 2 successive crosses corresponds to a simplex. The same feature can be seen in

dimension 2, except that now the simplexes are built along squares. In the following example, 2 monotony requirements are set with the code in Listing 2.

This gives the following textual result and diagrams.

```
---------------
CONSTRAINTS VERIFIED AT THE CORNERS
---------------
variable :  x_1
Number of Derivatives <0= 0 (Min= ) Number of Derivatives >0= 4 (Max=
1.00695)
variable :  x_2
Number of Derivatives <0= 0 (Min= ) Number of Derivatives >0= 3 (Max=
1.53403)
---------------
CONSTRAINTS VERIFIED AT THE EXPERIMENTAL POINTS
---------------
Nbr variables = 2, size of the experimental data base= 30, nb of con-
straints=2633
variable :  x_1
Number of first derivatives <0= 0 (Min= ) Number of first derivatives >0=
30 (Max= 0.827015)
variable :  x_2
Number of first derivatives <0= 0 (Min= ) Number of first derivatives >0=
30 (Max= 0.966707)
RMSE 0.094611

RMSE without constraints 0.067163
```

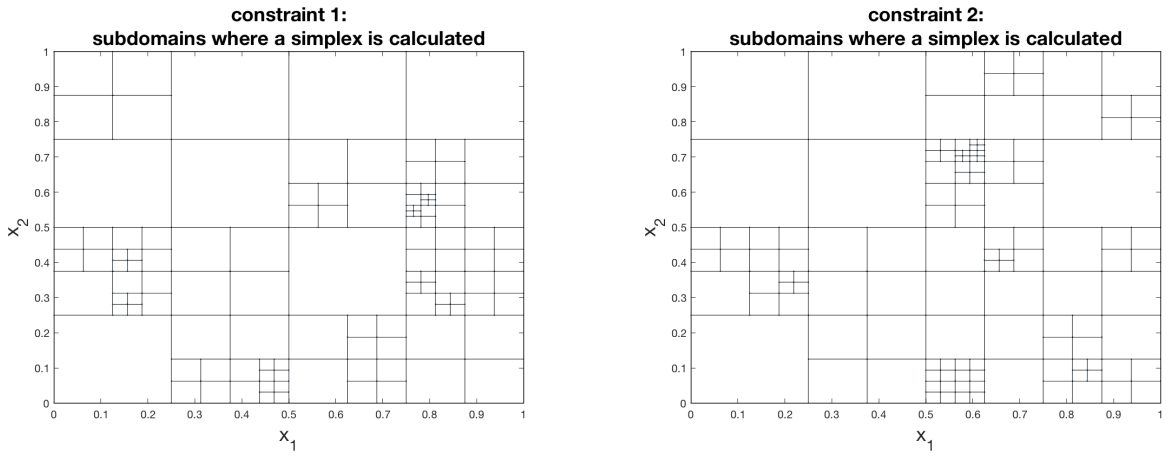

Figure 3: **constraints with 2 monotony requirements**