NOMMA 1.0

Non-Parametric Optimization Methods
for Model Assessment

Volkmar Sauerland
Kiel University

December 2017

1 Introduction

NOMMA 1.0 provides C++ implementations of a couple of non-parametric regression methods. The regression methods are of interest on their own but also useful for the assessment of parametric models [1] since they can provide lower error bounds on them.

There are two subfolders `regression` and `regressionCPX`. The first folder contains implementations of tailored algorithms for certain types of non-parametric regression [2, 3, 4]. These implementations do not require third party software to be installed. The folder `regressionCPX` contains an extension of the content in `regression`. The additional methods rely on direct formulations of regression problems in terms of *quadratic programs (QPs)*. The QPs are build and solved using the third party software *CPLEX*, which must be installed on your system in order to use `regressionCPX`.

2 Generating and Running Example Program

Both subfolders `regression` and `regressionCPX` contain the following files

- `Makefile`. On a LINUX platform with GNU g++ compiler, you can generate the executable `regEx` by typing `make`. In `regressionCPX` you need to change the variables `CPLEXDIR` and `CONCERTDIR` to point to the folders of your CPLEX installation.
- `regEx.cpp`. The example main program reads observational data (a uni-variate time series) and applies some regression methods implemented in `regression.cpp` and writes obtained regression time series to output files. Here, each line of the input file `sineLikeNoise_0.45_200.dat` consist of two real numbers: a time value and an associated measurement value. The first column (the time values) must be sorted in increasing order. The output files have the same format as the input file.
- `regression.cpp/hpp`. The class `regression` implements methods that are considered in our article [1] for the assessment of biogeochemical ocean models.
- `knot.cpp/hpp`. An auxiliary structure introduced in [4] and used to calculate "isotonic regression under Lipschitz constraint".
- `sineLikeNoise_0.45_200.dat`. Data file with 200 samples of the function $\sin(t) + 0.3 \cdot \sin(t) + \mathcal{N}(0, 0.45)$, serving as test observational data

After having generated the executable `regEx` (by typing `make`) you can type, e.g., `./regEx` to calculate some non-parametric regression time-series for the example data `sineLikeNoise_0.45_200.dat`.

3 Usage for Own Applications

The program `regEx.cpp` and the `Makefile` in both subfolders serve as examples how the implemented regression methods in `regression.cpp/hpp` might be used for your own applications. In the following we introduce the purpose and the syntax of each implemented method. A single call of any provided method will calculate a globally optimal fit to a given uni-variate time series of measurements $(\mathrm{xd}_i)_{i=1}^N$, taken at times $(\mathrm{td}_i)_{i=1}^N$, subject to a certain non-parametric property. The result is a uni-variate time series $(\mathtt{xr}_i)_{i=1}^N$ of the same length. All method implementations consider the summed squared errors

$$\sum_{i=1}^{N} (\mathrm{xd}_i - \mathtt{xr}_i)^2$$

as the objective misfit between data time series and regression time series. Subsection 3.1 summarizes the methods that are contained in the `regression.cpp` source of subfolder regression. Subsection 3.2 summarizes the additional QP-based methods implemented in subfolder regressionCPX. Subsection 3.3 gives guidelines, how to use the software in order to assess data-fits by parametric models, e.g., global biogeochemical ocean models.

3.1 Methods using Tailored Algorithms

There are three regression methods implemented in the source regression.cpp in folder regression

- `double pav( int N, int sign, double *td, double *xd, double *xr )`
  calculates an optimal regression time series subject to monotonicity using the *pool adjacent violators (PAV) algorithm* [2]. Parameters:
  - N: length of the time series that is to be fit
  - sign: indicator if regression time series will be monotonically non-decreasing (sign=1) or monotonically non-increasing (sign=-1)
  - td: data time series times
  - xd: data time series values
  - xr: regression time series
  - return value: summed squared errors of xr w.r.t. xd
- `double lpav( int N, int sign, double L, double *td, double *xd, double *xr )`
  calculates an optimal regression time series subject to monotonicity and bounded steepness using the *Lipschitz pool adjacent violators (LPAV) algorithm* [4]. Parameters:
  - N: length of the time series that is to be fit
  - sign: indicator if regression time series will be monotonically non-decreasing (sign=1) or monotonically non-increasing (sign=-1)
  - L: maximum allowed steepness $\left(\left|\frac{\texttt{xr[i + 1] - xr[i]}}{\texttt{td[i + 1] - td[i]}}\right| \leq L\right)$
  - td: data time series times
  - xd: data time series values
  - xr: regression time series
  - return value: summed squared errors of xr w.r.t. xd
- `double lpmr( int N, int k, int mode, double dMin, double dMax, bool considerSteepness, double *td, double *xd, double *xr )`
  calculates optimal regression time series subject to piecewise monotonicity and (optional) steepness bounds using a dynamic programming approach due to [3] and pav or lpav as subroutines. Parameters:
  - N: length of the time series that is to be fit
  - k: number of alternating monotonic segments
  - mode: chooses first monotonic regression segment to be increasing (mode $= 1$), decreasing (mode $= -1$), or to yield the lower misfit (value=0)
  - dMin: negative lower steepness bound
  - dMax: positive upper steepness bound
  - considerSteepness decides whether steepness bounds are considered (TRUE) or not (FALSE)

– `td`: data time series times
– `xd`: data time series values
– `xr`: regression time series
– `return value`: summed squared errors of `xr` w.r.t. `xd`

3.2 Additional Methods

The four additional methods in the `regression.cpp` source of subfolder `regressionCPX` formulate the considered properties in terms of quadratic programs which are solved using CPLEX.

- ```
  double isoReg( int N, int sign,
                 double *td, double *xd, double *xr )
  ```
  calculates an optimal regression time series subject to monotonicity by solving the corresponding QP

  $$\begin{aligned} \min \quad & \sum_{i=1}^{N} (\texttt{xr[i]} - \texttt{xd[i]})^2 \\ \text{s.t.} \quad & \texttt{sign} \cdot \texttt{xr[i]} \leq \texttt{sign} \cdot \texttt{xr[i + 1]} \end{aligned}$$

  Actually, `isoReg` is a (less efficient) alternative to `pav`. Parameters:

  – `N`: length of the time series that is to be fit
  – `sign`: indicator if regression time series will be monotonically non-decreasing (sign=1) or monotonically non-increasing (sign=-1)
  – `td`: data time series times
  – `xd`: data time series values
  – `xr`: regression time series
  – `return value`: summed squared errors of `xr` w.r.t. `xd`

- ```
  double slopeReg( int N, double dMin, double dMax,
                   double *td, double *xd, double *xr )
  ```
  calculates an optimal regression time series subject to steepness bounds by solving the corresponding QP

  $$\begin{aligned} \min \quad & \sum_{i=1}^{N} (\texttt{xr[i]} - \texttt{xd[i]})^2 \\ \text{s.t.} \quad & \texttt{xr[i + 1]} - \texttt{xr[i]} \geq \texttt{dMin} \cdot (\texttt{td[i + 1]} - \texttt{td[i]}), \\ & \texttt{xr[i + 1]} - \texttt{xr[i]} \leq \texttt{dMax} \cdot (\texttt{td[i + 1]} - \texttt{td[i]}). \end{aligned}$$

  Parameters:

  – `N`: length of the time series that is to be fit
  – `dMin`: negative lower steepness bound
  – `dMax`: positive upper steepness bound
  – `td`: data time series times
  – `xd`: data time series values
  – `xr`: regression time series
  – `return value`: summed squared errors of `xr` w.r.t. `xd`

- ```
  double minMaxReg( int N, int kMinA, int kMinB, int kMaxA, int kMaxB,
                    double dMin, double dMax, double T,
                    double *td, double *xd, double *xr )
  ```
  calculates an optimal regression time series subject to the properties of having exactly one local minimum, exactly one local maximum, bounded steepness, and, optionally, being periodic. For this problem, a QP similar to the "`slopeReg` QP" is solved for each allowed combination of minimum and maximum position. An alternative is the generalization `lpmrIPQ`. Parameters:
  - `N`: length of the time series that is to be fit
  - `kMinA, kMinB, kMaxA, kMaxB`: the minimum and maximum values of the regression time series are supposed to appear in the index intervals [`kMinA, kMinB`] and [`kMaxA, kMaxB`], respectively
  - `dMin`: negative lower steepness bound
  - `dMax`: positive upper steepness bound
  - `T`: supposed period of the regression time series
    $\mathtt{T} \leq 0$ means that no period is specified
  - `td`: data time series times
  - `xd`: data time series values
  - `xr`: regression time series
  - `return value`: summed squared errors of `xr` w.r.t. `xd`

- ```
  double lpmrIQP( int N, int nMin, int nMax, int sign,
                  double dMin, double dMax, double T,
                  double *td, double *xd, double *xr )
  ```
  calculates an optimal regression time series subject to the property of having a predefined number of minima, a predefined number of maxima, bounded steepness and, optionally, being periodic. This problem is formulated in terms of an *integer quadratic program (IQP)*. Parameters:
  - `N`: length of the time series that is to be fit
  - `nMin, nMax`: number of local minima and local maxima in the regression time series. Conditions: $\mathtt{nMin} + \mathtt{nMax} \leq N - 2$, $|\mathtt{nMin} - \mathtt{nMax}| \leq 1$
  - `sign`: if $\mathtt{nMin} = \mathtt{nMax}$, `sign` indicates if the regression time series is chosen to start with a non-decreasing segment ($\mathtt{sign} = 1$), with a non-increasing segment (`sign=-1`), or to yield the lower misfit of both choices ($\mathtt{sign} = 0$)
  - `dMin`: negative lower steepness bound
  - `dMax`: positive upper steepness bound
  - `T`: supposed period of the regression time series
    $\mathtt{T} \leq 0$ means that no period is specified
  - `td`: data time series times
  - `xd`: data time series values
  - `xr`: regression time series
  - `return value`: summed squared errors of `xr` w.r.t. `xd`

3.3 Usage for the Assessment of Parametric Models

Suppose you have a uni-variate time series of measured data to which you fit some parametric model. If your model is an elaborated non-linear one, it is difficult to find its globally best adaption to the data. However, if you can prove (or justify) that your model always satisfies a property that is addressed by some of the methods above you can apply this method to your data in order to obtain a lower bound on the best attainable misfit w.r.t. your parametric model.
It is also possible to obtain lower bounds on the model-data misfit of a multi-variate model, say

$$F : P \times T \to \mathbb{R}^m,$$

with k parameters, parameter space $P \subseteq \mathbb{R}^k$, time interval $T \subseteq \mathbb{R}$, and m dependent variables of interest. The objective misfit between the multi-variate model is often derived from a weighted sum of corresponding SSE misfits for the single dependent variables, e.g.,

$$\sum_{j=1}^{k} w_j \cdot \sum_{i=1}^{N} \left(F_j(p, \mathtt{td}_i) - \mathtt{xd}_{i,j}\right)^2 ,$$

where each member of the data time series $(\mathtt{xd}_i)_{i=1}^N$ is a vector of measurements value of the m dependent variables and F_j, $j \in \{1, \ldots, m\}$, denotes the j-th component F (which belongs to the j-th dependent variable). Note that the m components of F and of the m-variate data time series might be associated with different quantities of interest but also with different grid boxes of a spacial partition. Now, a lower bound on the above model-data misfit is derived by summing up single lower bounds correspondingly, e.g.,

$$\sum_{j=1}^{k} w_j \cdot \alpha_j ,$$

where each α_j is the lower bound obtained from a non-parametric regression method applied to the j-th dependent variable, only. An example for this approach is the assessment of a global biogeochemical ocean model in [1].

References

[1] V. Sauerland, U. Löptien, C. Leonhard, A. Oschlies, and A. Srivastav. Error assessment of biogeochemical models by lower bound methods. *Geoscientific Model Development Discussions*, pages 1–22, 2017. In review.

[2] R. E. Barlow, D. J. Bartholomew, J. M. Bremner, and H. D. Brunk. *Statistical Inference under Order Restrictions. Theory and Application of Isotonic Regression.* Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, London, 1972.

[3] I. C. Demetriou and M. J. D. Powell. Least squares smoothing of univariate data to achieve piecewise monotonicity. *IMA Journal of Numerical Analysis*, 11(3):411–432, 1991.

[4] L. Yeganova and W. J. Wilbur. Isotonic regression under Lipschitz constraint. *Journal of Optimization Theory and Applications*, 141(2):429–443, 2009.