# BIVARIATE NON-ISOTONIC STATISTICAL REGRESSION BY A LOOK-UP TABLE NEURAL SYSTEM [§]

SIMONE FIORI [†], TIANXIA GONG [*], AND HWEE KUAN LEE [‡]

**Abstract.** Linear data regression is a fundamental mathematical tool in engineering and applied sciences. However, for complex non-linear phenomena, the standard linear least-squares regression may prove ineffective, hence calling for more involved data modeling techniques. The current research work investigates, in particular, non-linear statistical regression of bivariate data that do not exhibit a monotonic dependency. The current contribution proposes a neural-network-based data processing method, termed *data monotonization*, followed by neural isotonic statistical regression. Such data monotonization processing is performed by means of an adaptive neural network that learns its non-linear transfer function from the training set. The artificial neural system that performs data monotonization is implemented through a look-up table (LUT), which entails few computationally-inexpensive algebraic operations to adapt and to compute the output from the input data-set. A number of learning rules to adapt such LUT-based neural system are introduced and compared, in order to elucidate their relative merits and drawbacks.

**Keywords:** Non-linear neural modeling, Look-up table neural network, Statistical regression, Isotonic regression, Bivariate data monotonization.

**1. Introduction.** Regression is a fundamental data processing technique that aims at inferring a target value on the basis of a set of predictors. The structure and the parameters' values of empirical mathematical regression models are determined using experimental data [22]. Regression may be employed, for instance, to model the concentration of bacteria corresponding to a given dose of drug in the experimental analysis of drug effectiveness. In bivariate data regression, it is assumed that $m + 2$ variables are related by the functional relationship:

$$y = \Phi(x, \nu_1, \ldots, \nu_m), \tag{1.1}$$

where $y \in \mathcal{Y}$ represents the *dependent variable* or *target*, $x \in \mathcal{X}$ represents the *independent variable* or *predictor*, $(\nu_1, \ldots, \nu_m) \in \mathcal{N}$ represent *nuisance variables*, and the function $\Phi : \mathcal{X} \times \mathcal{N} \to \mathcal{Y}$ represents a physical phenomenon. The symbol $\mathcal{X}$ denotes the input data-set, the symbol $\mathcal{Y}$ denotes the output data-set, while the pair $(\mathcal{X}, \mathcal{Y})$ denotes the training set. The $m$ nuisance variables may account for measurement errors and are typically hidden. The same functional relationship accounts for a multi-predictor problem when only one of the predictors is dominant [15]. The aim of statistical regression is to infer a model of the form $y = f(x)$, where the function $f : \mathcal{X} \to \mathcal{Y}$ is estimated on the basis of statistical information extracted from the training set $(\mathcal{X}, \mathcal{Y})$. Namely, rather than considering the $x$ and $y$ variables as paired and to look for a non-linear model that fits their values in the best way (according to a pre-defined measure-of-fit), the statistical information carried on by the data-sets $\mathcal{X}$ and $\mathcal{Y}$ in terms of their probability density functions only are made use of.

A special instance of non-linear modeling is *isotonic regression* [10], which arises in various fields, such as production planning, inventory control, and psychometry [8]. The notion of isotonic regression may be traced back to the seminal contribution [6] and was motivated by the observation that monotonic dependencies among variables are common in physical systems. Statistical isotonic regression is based on the fact that the majority of the challenging applied isotonic modeling problems are characterized by

[†]Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Via Brecce Bianche, I-60131 Ancona (Italy). (**Corresponding author**.)

[*]School of Computing, National University of Singapore, Computing 1, 13 Computing Drive, 117417 Singapore (Republic of Singapore)

[‡]Bioinformatics Institute, A*STAR Biomedical Sciences Institutes, 30 Biopolis Street, #07-01 Matrix building, 138671 Singapore (Republic of Singapore)

[§]This paper was published as **S. Fiori, T. Gong and H.K. Lee, *Bivariate nonisotonic statistical regression by a lookup table neural system*, Cognitive Computation, Vol. 7, No. 6, pp. 715 – 730, December 2015**

very large-sized training sets [3, 29], which afford an accurate estimation of the statistical features from the input/output data-sets (such as their marginal probability density functions).

In previous contributions by the first author on bivariate isotonic statistical regression [14, 15] and on trivariate isotonic statistical regression [16], the developed algorithms were implemented by an artificial neural system that adapts to match its own input-output statistics with the empirical statistics of the training set. The quantities of interest as well as the learnt regression model were represented in terms of *look-up tables* (LUTs) [13]. As a result, the devised neural learning algorithm does not involve cumbersome computations except for sorting/searching on LUTs and few simple algebraic operations. Moreover, the mathematical operations to be performed on the data-sets $\mathcal{X}$ and $\mathcal{Y}$ might be largely parallelized. In addition to a light computational cost, such regression procedure is capable of capturing complex non-linear phenomena thanks to its versatile LUT-based neural system architecture. However, a fundamental limitation of such an approach is that it entails a monotonic dependency between the input and the output data.

The current paper suggests and discusses a method to overcome the limitations of the previous contributions from the first author and make isotonic statistical regression applicable to non-monotonic training sets. The key idea is that *non-monotonic data can be transformed to monotonic data* by an invertible transform. After such transformation, neural isotonic regression may be applied to the monotonized training set and the regression result may be transformed back to its non-monotonic form. The non-linear transform, that makes non-monotonic data approximately monotonic, will be implemented by a learnable LUT neural system.

The neural regression algorithm proposed in the present contribution inherits several advantages from the isotonic statistical regression algorithm discussed in [15] while being able to circumvent the requirement of monotonicity of the inferred regression model. The main advantage of the proposed method is that, while in non-statistical regression methods the functional form of the model is given *a priori* and may not fit the data appropriately, in the proposed approach the shape of the regression function is unrestricted, namely, there is no assumption on the functional shape of the model underlying the training set nor the model is restricted to a combination of predefined basis functions. Such feature allows the devised neural regression procedure to cope with relationships among data exhibiting, for instance, jump discontinuities.

The present contribution discusses and compares three different learning rules to adapt the LUT-based neural system. Such learning rules, and the resulting statistical regression procedures, will be tested and compared on three data-sets, two of which were specifically designed to challenge the devised data-monotonization algorithms. The present paper is organized as follows. The Section 2 presents a review of the literature in neural-network based regression and illustrates, with several examples, the wide range of applications of nonlinear regression in system biology and cognitive computation. The Section 3 presents an overview of the neural statistical regression problem and of the fundamental assumptions underpinning it, along with the proposed solution. Section 3 also overviews the data-sets that will be used for testing and illustrates results of numerical experiments. The Section 4 concludes the paper and describes a few foreseen research lines, related to neural statistical regression, to be pursued in the near future.

**2. Literature review.** Data regression and modeling find applications in trend analysis, business planning, marketing, financial forecasting, environmental modeling as well as drug response prediction. An important example of application to the prediction of drug response is given by the analysis of kinase inhibitors responses recently presented in [34]. Kinases are critical in metabolism, cell signaling, protein regulation, cellular transport and secretory processes. A large number of kinase inhibitors have been approved as cancer therapies. Libraries of kinase inhibitors have been extensively profiled, providing a map of the strength of action of each compound on a large number of targets, and are used to define drug-kinase reactions that can *predict the effectiveness of untested drugs*. However, predicting the effectiveness of a drug on the basis of a model of cellular signaling is difficult, due to the partial knowledge of the

complex biological processes involving the targeted kinases. The paper [34] describes a method that integrates information contained in drug-kinase reactions with *in vitro* screening through regression. Such method uses the *in vitro* cell response of single drugs and drug pair combinations as a training set to build nonlinear regression models. In particular, such method was applied to a lung cancer cell line and was able to identify specific kinases known to play an important role in such kind of cancer.

Mathematical models are an essential tool, i.e., in systems biology, which is a rapidly growing research field that aims at understanding the interactions between the components of biological systems. As an example, the paper [36] illustrates the application of a non-linear regression technique in the simulation of tumor vessel network growth. The popular bivariate *linear* least-squares regression method, which is often encountered in sciences and engineering, fits a straight line (or a flat hyper-plane, in the case of multivariate regression) to a set of data points. Oftentimes, however, the true relationship that needs to be modeled is curved, rather than flat. A workaround method to cope with nonlinearly related data is to adapt the linear least-squares method to *transformed* data by creating new variables as nonlinear functions of the data. Upon constructing the new variables properly, the curved function of the original variables can be expressed as a linear function of the new variables. A popular example is the non-linear model $y = \Phi(x, \nu_1) = Ax^b\nu_1$, where $x > 0$ denotes the independent variable, $y > 0$ denotes the dependent variable, $A > 0$ and $b \in \mathcal{R}$ are constants and $\nu_1 > 0$ denotes a nuisance (i.e., error, noise) non-deterministic variable. Such relationship may be exactly linearized by taking the logarithm of the variable $y$, which will exhibit a linear dependency of the logarithm of the variable $x$, in fact, $\log y = \log A + b \log x + \log \nu_1$. Defining the new data-variables $Y = \log y$ and $X = \log x$, the deterministic model underlying the training set may be taken as $Y = f(X) = c + bX$, with $c = \log A$. In general, such an approach requires a large deal of work to individuate a suitable set of non-linear transformations that *linearize* the regression problem. Most available software applications to perform non-linear regression try to fit the data by drawing from a library of non-linear functional dependencies and select the most representative model as the one that warrants the best fit. *Non-linear regression*, instead, tries to cope with the underpinning non-linearity by working out the original training set directly. Statistical non-linear regression, in particular, is currently being applied to a variety of research fields, such as integrated electronics [2, 24], environmental research [18, 25], atmospheric research [23] as well as to different branches of artificial intelligence [26].

According to the discussion presented in the paper [5], there are two cultures in the use of statistical regression. One assumes that the data are generated by a given stochastic model, while the other uses algorithmic models and treats the data mechanism as unknown. Algorithmic regression has developed rapidly because it can be used both on large complex data-sets and as a more accurate and informative alternative to data regression on smaller data-sets. In particular, the worth of artificial neural systems to model complex, non-linear relationships among data is desirable for many real world problems, including regression. The use of neural networks, such as large-sized multi-layer perceptrons, however, entails some drawbacks: Out of necessity, artificial neural networks initialize the weights to random values, which explains the inconsistency in the results obtained by running a classical artificial neural network repeatedly on the same training set. In addition, the optimization process that tunes the weights might get trapped into local minima. A popular neural system, especially designed for the purpose of data regression, is the General Regression Neural Network (GRNN) [32]. A GRNN is made of a radial basis function layer and a special linear layer, and it is a one-pass learning algorithm with a highly parallel structure that provides estimates of continuous variables and converges to the underlying (linear or nonlinear) regression surface. Running a GRNN, likewise most of the neural systems, *requires the user to select a functional model*, under the form of network topology and transfer function of the artificial neurons. Moreover, the literature offers a number of key research contributions about extracting monotone estimators from data, which is a closely linked topic. Friedman and Tibshirani citefriedman considered the combination of isotonic regression and monotone smoothing of data. Hall and Shuang provide a way of extracting a monotone regressor from any kernel-based nonparametric estimator [21]. Ramsay analyzes monotonicity constrains for spline functions [30].

Non-linear regression is an ubiquitous topic in sciences and engineering, including cognitive computation. Regression was utilized in the paper [33] to model emotional intensity. In particular, such contribution aims at understanding the computational function of emotion through the development of a model of emotion that is derived from computational principles in neuroengineering. Such study focuses on those computational roles of emotion that can be assessed objectively through experiments. The paper [1] deals with oyster data classification. Shucking and grading oysters is a labor intensive task and it was automatized by means of a 3D computer vision system. Oysters were defined to be small, medium or large according to their volume and data were collected in a calibration experiment in which regression models were used to predict observed oyster volume, one using digital image area (2D) in pixels as a predictor, and another using digital image volume (3D). Likewise, regression was used to interpolate between spectral coefficients in the contribution [9], which deals with prosodic speech signal modification in high-quality environments. Such speech modification techniques allow *text-to-speech systems* to synthesize more expressive speeches without requiring extensive corpora resources. Regression plays a fundamental role in the analysis of *Granger causality* of two experimentally-measured variables performed in [35]. Correlation and Granger causality analysis were utilized to reveal the functional connectivity patterns of the primary sensory system applied to the design of the neural architecture of autonomous humanoid robots. The paper [37] discusses an automatic artifact removal method which combines *independent component analysis* and *wavelet denoising* to analyze attention deficit hyperactivity disorder electroencephalograms. In such context, a simple, yet effective way to remove some artifacts is provided by regression on the electroencephalographic data. In a related way, regression was made use of for characterizing autocorrelation functions in the analysis of electrocardiographic signals in the contribution [27], which studies the problem of classifying music-induced emotions on the basis of forehead biosignals and electrocardiogram. The paper [20] studied a method for characterizing neurological diseases by a biomechanical analysis of voice quality. Such study showed that a tremor during phonation may be due to unexpected changes of vocal fold tension, namely, a neurological disease may produce correlates in the vocal fold body stiffness. In such context, regression is used to model stiffness during a phonation cycle. Likewise, an application of regression was proposed recently in [4] with the aim of improving automatic detection of severe obstructive sleep apnea on the basis of patients' voices. In particular, such contribution proposed a nonlinear analysis of speech, where nonlinear features were conveyed to speech recognition systems using statistical regression techniques for both continuous and sustained speech. The paper [12] uses regression to extract relevant information from a set of *Mel frequency cepstral coefficients* (MFCCs) in the context of automatic speech recognition. Such study suggests to combine the MFCCs with their first-order and second-order regression coefficients along with *fractal dimension* features in order to improve the performances of an automatic speech recognition system. A related regression-based method was employed in [31] to investigate speaker group-dependent modeling aimed at the recognition of affective states on the basis of human utterance.

**3. Neural learning rules for data-monotonization and regression.** Let us denote the data-pairs to be modeled by $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, where $i = 1, 2, \ldots, N$, and the underlying relationship between the variables $x$ and $y$ by $y = f(x)$. The function $f(x)$ may be non-monotonic and may not have an analytic form. The data-pairs $(x_i, y_i)$ will constitute the training set to adapt a neural regression system.

While the neural isotonic statistical regression method proposed in [15] allows the $x$-data and the $y$-data to come unpaired, in the present non-monotonic regression context, it is necessary to require that the exact pairing between each value $x_i$ and each value $y_i$ be known, otherwise the regression problem would not admit a unique solution. The key idea of the current paper is that, in order to benefit from the neural statistical regression method for monotonic data explained in the contribution [15], it is necessary to transform the original training set to become approximately monotonic. The general scheme employed in the present paper is: (a) Transform the original training set so that the relationship between the dependent variable and the independent variable becomes (approximately) monotonic; (b) Perform a neural isotonic statistical regression by the method explained in [15] (and summarized in the Appendix A

for the convenience of the readers) c) Reverse-transform the built model to its original non-monotonic shape. Therefore, while the neural regression method proposed [15] allows missing records in a data-set, in the context of non-monotonic modeling, no incomplete records are allowed in the training set. In fact, although the inner isotonic regression procedure would be able to handle missing data in the records, the procedure of learning a data transformation that makes it monotonic the original non-monotonic data cannot handle incomplete records of the type $(\bullet, y_i)$. However, data-pairs of the form $(x_i, \bullet)$ can still be used for statistical regression directly without any pre-processing as they contribute to the estimation of the probability density function of the $x$-values.

The transfer function of the neural data-monotonization system reads $h(x) = g(x)\,\bar{f}(x)$, where $\bar{f}(x)$ denotes the model-function $f(x)$ lifted upward in such a way that $\bar{f}(x) > 0$ for every value of the variable $x$. The neural transfer function $g : \mathcal{X} \to \mathcal{Y}$ is termed *multiplicative transform function* and is to be learnt from the training set $(\mathcal{X}, \mathcal{Y})$. The monotonicity condition on the function $h(x)$ reads:

$$h'(x) = g'(x)\bar{f}(x) + g(x)\bar{f}'(x) > 0 \tag{3.1}$$

and will be ensured by designing an appropriate learning rule for the neural system. The current Section aims at introducing and discussing three learning rules to adapt the neural system used to perform such multiplicative transform, which are summarized below:

- **Learning rule 1**: It is based on data smoothing, regression model pre-estimation and learning of the multiplicative transform function;
- **Learning rule 2**: It allows data-wise multiplicative transform learning via a multiplicative update rule;
- **Learning rule 3**: It affords data-wise multiplicative transform learning via an additive update rule.

The relative merits and drawbacks of the introduced learning rules will be illustrated via numerical examples on three data-sets, two synthetic and one drawn from a real world application, namely:

**Training-set 1**: In the first data-set, the samples $x_i$ of the variable $x$ are generated as 301 numbers equally spaced in the interval $[0,\ 6]$ with a random number $0.1r_{x_i}$ added to each $x_i$. The data-points $y_i$ are generated according to the following function:

$$y_i = \begin{cases} 0.5x_i + 0.25r_{y_i}, & \text{for } 0 \le x < 1 \text{ or } 2 \le x < 3, \\ 0.5x_i + 1 + 0.25r_{y_i}, & \text{for } 1 \le x < 2, \\ -0.5x_i + 2.5 + 0.25r_{y_i}, & \text{for } 3 \le x < 4, \\ 0.5x_i - 1.5 + 0.25r_{y_i}, & \text{for } 4 \le x \le 6, \end{cases}$$

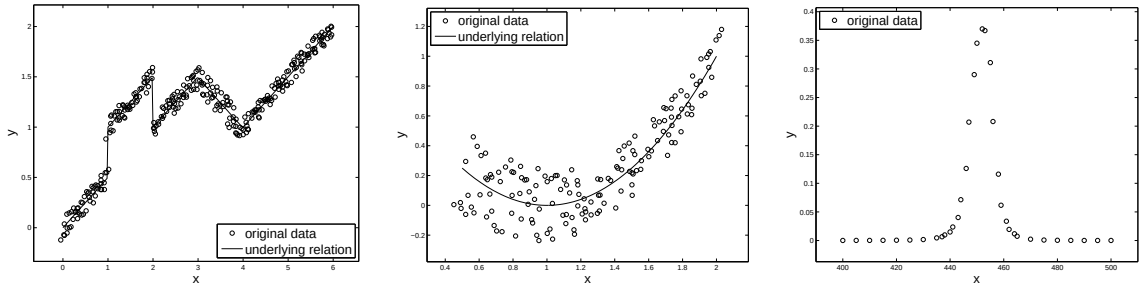
where $r_{x_i}$ and $r_{y_i}$ are random numbers drawn from a uniform distribution in the interval $[-0.5,\ 0.5]$. The data plot is shown in Figure 3.1(a). Such data-set is clearly non-monotonic, non-smooth and presents a jump discontinuity in $x = 1$.

**Training-set 2**: In the second data-set, the samples of the variable $x$ are generated as 151 numbers equally spaced in the interval $[0.5,\ 2]$ with a random number $0.1r_{x_i}$ added to each $x_i$. The data-points $y_i$ are generated according to the following function:

$$y_i = (x_i - 1)^2 + 0.5r_{y_i},$$

where $r_{x_i}$ and $r_{y_i}$ are random numbers drawn from a uniform distribution in the interval $[-0.5,\ 0.5]$. The data plot is shown in Figure 3.1(b). This data-set is non-monotonic and presents a large noise component in the $y$-variable.

**Training-set 3**: The third data-set was downloaded from the repository described in the reference [11] and is the result of a NIST study involving circular interference transmittance. The present data-set contains 35 records, where the response variable $y$ is a transmittance, while the predictor variable $x$ is a wavelength. The data plot is shown in Figure 3.1(c).




(a) Training data-set 1: The circles represents the data-point $(x_i, y_i)$ in the training set and the solid line represents the underlying relation $f(x)$ between the variables $x$ and $y$.

(b) Training data-set 2: The circles represents the data points $(x_i, y_i)$, where the solid line represents the underlying relation $f(x)$ between the variables $x$ and $y$.

(c) Training data-set 3: Real-world data from a circular interference transmittance analysis [11].

FIG. 3.1. *Three training sets used in the numerical experiments.*

In particular, each learning rule will be tested over each training data-set. A detailed analysis of the obtained results will be offered, along with a comparison with classical regression techniques (polynomial fit or Gaussian fit, depending on the data-set).

**3.1. Learning rule 1: Data smoothing, pre-estimation and learning of a multiplicative transform.** The first learning rule discussed in the present paper serves to explain the core idea of the current research endeavor. Let us assume that the training set $\{(x_i, y_i)\}$ has been shifted upwards to $\{(x_i, \bar{y}_i)\}$ in such a way that, for every value of the index $i$, it holds that $\bar{y}_i > 0$. The sought neural regression model may be written as $\bar{f}(x_i) = \bar{y}_i$. In addition, assume that the regression model $\bar{f}(x)$ be estimated by using a smooth function $k(x) \approx \bar{f}(x)$.

In order to obtain an estimate $k(x)$, a Gaussian kernel is made use of to smooth out the data. For each training data-point $(x_i, \bar{y}_i)$, a window of given width is slid along the $x$-axis to compute the values $k(x_i)$. In order to control the approximate number of data points falling within the sliding window, the width $r$ of the window is defined as:

$$r = \frac{x_{\max} - x_{\min}}{N} \cdot \frac{p}{2},$$

where the quantities $x_{\min}$ and $x_{\max}$ represent the minimum and the maximum value of the $x$ variable among all the available records in the data-set, respectively, the term $(x_{\max} - x_{\min})/N$ denotes the average distance between two adjacent data points along the $x$-axis and $p$ denotes the approximate number of data points falling within the Gaussian window. The integer constant $p$ needs to be chosen in such a way that the data will not become over-smoothed nor under-smoothed: For sparse data, a lower value of the constant $p$ will be chosen, so that the sliding window will result narrow enough to avoid over-smoothing, while for denser data, a larger value of the constant $p$ will be chosen so that the sliding window will be wide enough for the data not to result under-smoothed. Let $x_i$ locate at the center of the sliding window and let there be $j$ contiguous points, from $x_{i-j}$ to $x_{i-1}$, on the left-hand side of the point $x_i$ and $k$ neighbors, from $x_{i+1}$ to $x_{i+k}$, on the right-hand side of the point $x_i$, within the sliding window. The weights assigned to the data-point $x_i$ and its neighbors are computed from the Gaussian kernel and are denoted as $w_{i-j}, \ldots, w_i, \ldots, w_{i+k}$. The value $k(x_i)$ for the data-point $(x_i, \bar{y}_i)$ is computed as:

$$k(x_i) = \frac{\sum_{a=i-j}^{a=i+k} \bar{y}_a \, w_a}{\sum_{a=i-j}^{a=i+k} w_a}. \tag{3.2}$$

In order to learn a multiplicative transform function $g(x)$ to turn the data-set $\{\bar{y}_i\}$ into the monotonized data-set $\{z_i\}$, let us set $h(x) = g(x)\,k(x)$ and let us require that the function $h(x)$ be monotonic, i.e., that $h'(x) > 0$. Note that the multiplicative transform function is not unique: Any neural transfer function $g$ that is able to generate a monotonic result $h$ will suffice. The learning rule to adapt the LUT neural system implementing a multiplicative data transform is designed on the basis of the inequality:

$$h'(x) = g'(x)k(x) + k'(x)g(x) > 0. \tag{3.3}$$

Since the function $k(x)$ is always positive, by construction, it must hold that $g'(x) > -\frac{k'(x)}{k(x)}g(x)$. The above inequality may be rewritten as:

$$g'(x) = -\frac{k'(x)}{k(x)}g(x) + \epsilon_1, \tag{3.4}$$

with $\epsilon_1 > 0$. In order to write a rule that allows learning the multiplicative transform function, the range of the variable $x$ is subdivided into $n$ equally-spaced sections of width $\Delta = (x_{\max} - x_{\min})/n$, and a new set of $n$ grid points $\{x_0,\ x_1,\ \ldots,\ x_\lambda,\ \ldots,\ x_n\}$ is generated, where $x_\lambda = x_0 + \lambda\Delta$, with $\lambda = 1,\ \ldots,\ n$. Each necessary value $k(x_\lambda)$ is calculated by cubic spline interpolation so that the derivative function $k'(x)$ can be numerically approximated over the grid-points. Each value $g(x_\lambda)$ is calculated by estimating $g'(x_\lambda)$ iteratively over the grid points through the following **Learning rule 1**:

$$g(x_\lambda) = \Delta g'(x_{\lambda-1}) + g(x_{\lambda-1}), \tag{3.5}$$

$$g'(x_\lambda) = -\frac{k'(x_\lambda)}{k(x_\lambda)}g(x_\lambda) + \epsilon_1. \tag{3.6}$$

The boundary condition is set to $g(x_0) = 1$. After the values $g(x_i)$ of the neural transfer function have been learnt, the original data-set $\{(x_i, y_i)\}$ can be transformed to be a monotonically increasing data-set $\{(x_i, z_i)\}$, where $z_i = g(x_i)\,\bar{y}_i$. Isotonic regression can be applied to the transformed data-set made of the pairs $(x_i, z_i)$. After the regression function has been learnt, it needs to be transformed back to obtain the sought relation between the variables $x$ and $y$.

The operations of smoothing, learning and applying the multiplicative transform, and of performing regression, were tested on the three training data-sets overviewed at the beginning of the present Section. The number of grid points was set to $n = 1000N$, where $N$ is the number of records in the training set. The value of the constant $\epsilon_1$ in equation (3.4) was set to be proportional to the relative noise level in the data: If the value of the constant $\epsilon_1$ is too low relatively to the noise level in the data, some degree of non-monotonicity may be retained undesirably in the transformed data. Conversely, a large-valued constant $\epsilon_1$ may be unnecessary for data affected by lower noise levels. The Figures 3.2, 3.3 and 3.4 show the multiplicative transform function learnt by the *Learning rule 1* and the result of isotonic regression applied to the training *Data-set 1*, to the training *Data-set 2* and to the training *Data-set 3*, respectively. According to the Learning rule 1, which uses smoothing, the point-wise multiplicative transform function $g$ is learnt on the basis of the smoothed data, therefore, the function $g$ itself is smooth. However, as the numerical results illustrated in the Figures 3.2, 3.3 and 3.4 testify, when applying the neural multiplicative transform to the noisy training set and to the real-world training set, the transformed data contains some amount of noise and thus the result is *almost monotonically increasing* as a trend (but not exactly monotonic for all the data points). Nevertheless, the neural regression algorithm is able to learn the general trend from the training set and return a smooth line as regression result. When transforming the regression result back using the inverse of the function $g$, as such function is smooth, the transformed regression curve results smooth.

The main difficulty with this kind of learning rule is that the desirable degree of smoothness is hard to determine, hence under-smoothing and over-smoothing can occur if the window-size used in the smoothing step is not set properly. Moreover, the Learning rule 1 suffers from a serious practical

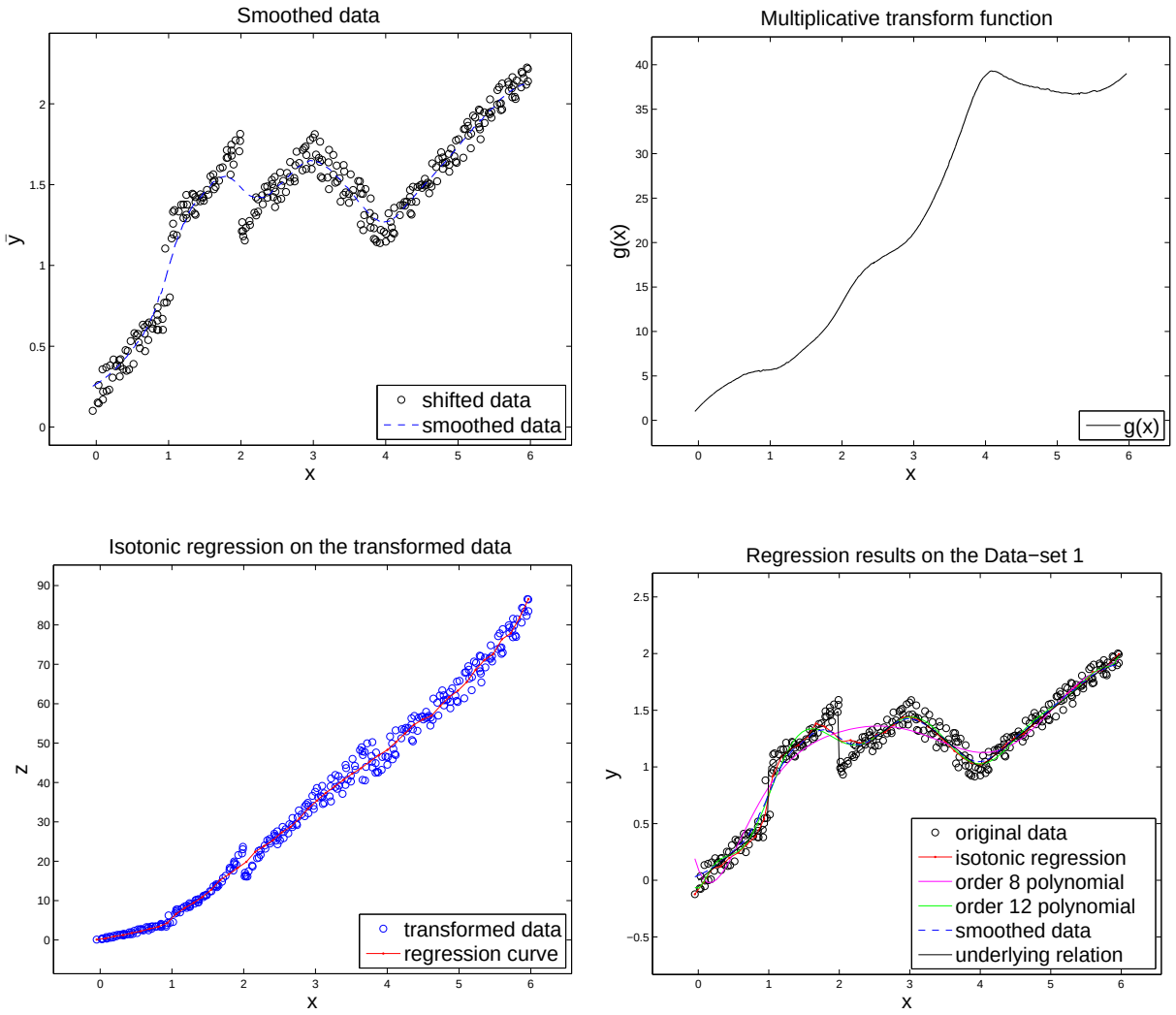



FIG. 3.2. *Multiplicative transform learnt by the* Learning rule 1 *and isotonic regression applied to the* Data-set 1*: Result of smoothing, multiplicative transform function, regression on monotonized data and comparison to polynomial regression.*

inconvenient, namely, it is based on a pre-estimation of the model behind the training data-set in order to learn the multiplicative transform function. The smoothing pre-processing operation may change the shape of the data radically, especially in presence of abrupt oscillations. In order to mitigate such drawbacks, the two following learning rules have been developed.

**3.2. Learning rule 2: Data-wise multiplicative transform, multiplicative update.** As opposed to the previous learning method, the multiplicative transform function $g$ is learnt directly from the training set without any smoothing nor pre-estimation. The first step consists again in shifting upwards the training set in order to ensure that the $y$-samples be positive-valued: Let $\bar{y}_i = y_i + c$, such that $\bar{y}_i > 0$ holds for every value of the index $i$ and let us set $\bar{f}(x_i) = \bar{y}_i$. The derivatives on the right-hand side of the differential inequality (3.1) may be approximated by finite differences. By means of the Euler

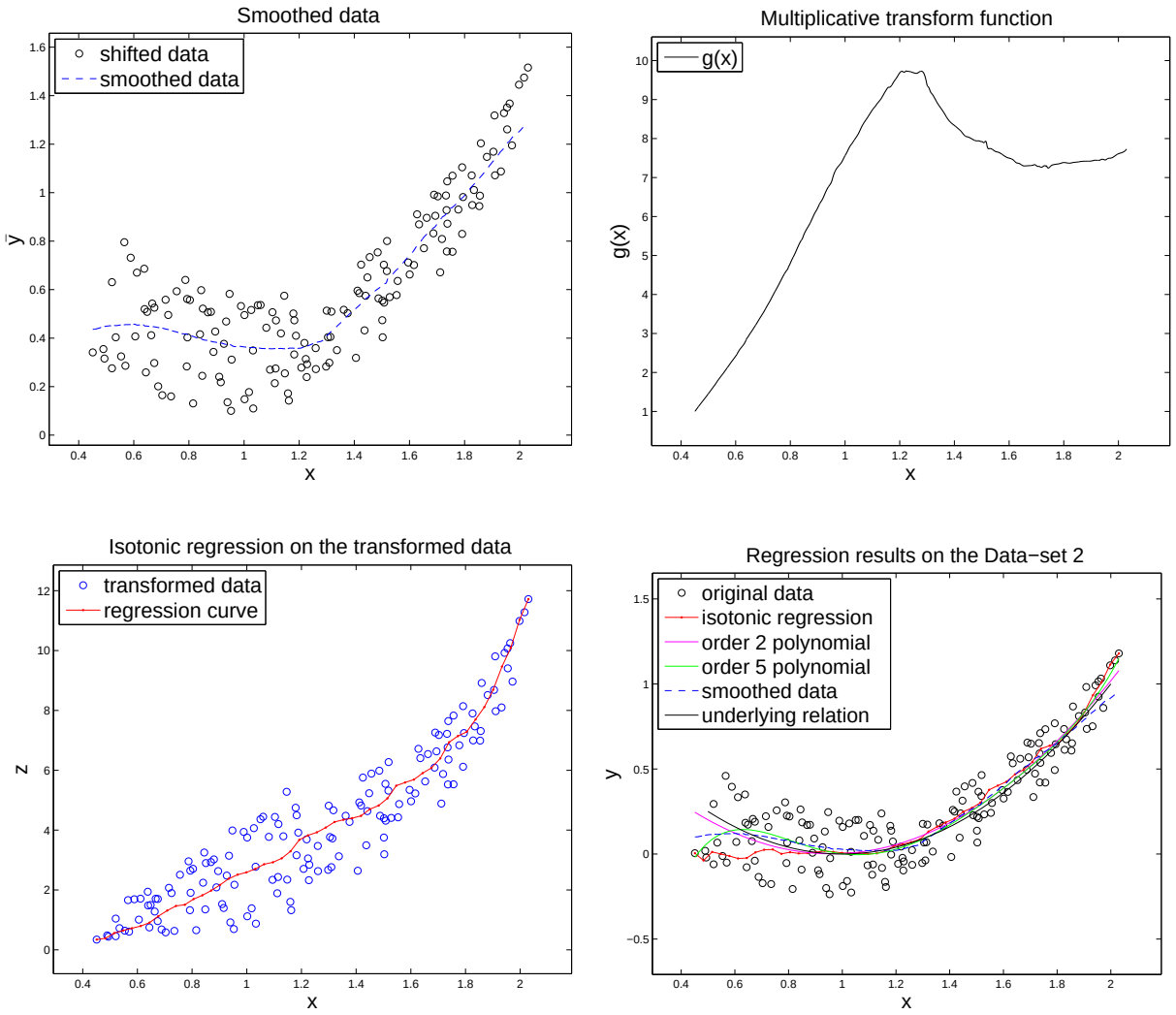



FIG. 3.3. *Multiplicative transform learnt by the* Learning rule 1 *and isotonic regression applied to the* Data-set 2*: Result of smoothing, multiplicative transform function, regression on monotonized data and comparison to polynomial regression.*

approximation, such inequality becomes:

$$\frac{g(x_i) - g(x_{i-1})}{\Delta_i} \bar{f}(x_i) + g(x_i) \frac{\bar{f}(x_i) - \bar{f}(x_{i-1})}{\Delta_i} > 0. \tag{3.7}$$

The above expression may be simplified by leaving the term $\Delta_i$ out, provided that it holds $\Delta_i = x_i - x_{i-1} > 0$ for every value of the index $i$. In order to achieve such result, the training-set pairs $(x_i, \bar{y}_i)$ are sorted in ascending order with respect to the $x$-variable, (namely, the data-pairs are sorted in such a way that $x_i > x_{i-1}$ strictly)[1]. Let us set, for the sake of notational conciseness, $g_i = g(x_i)$. The condition (3.7)

[1] If there exist multiple records with an identical $x_i$-value, low-valued random displacements may be added to the $x$-value of such records, in such a way that those records may be sorted.

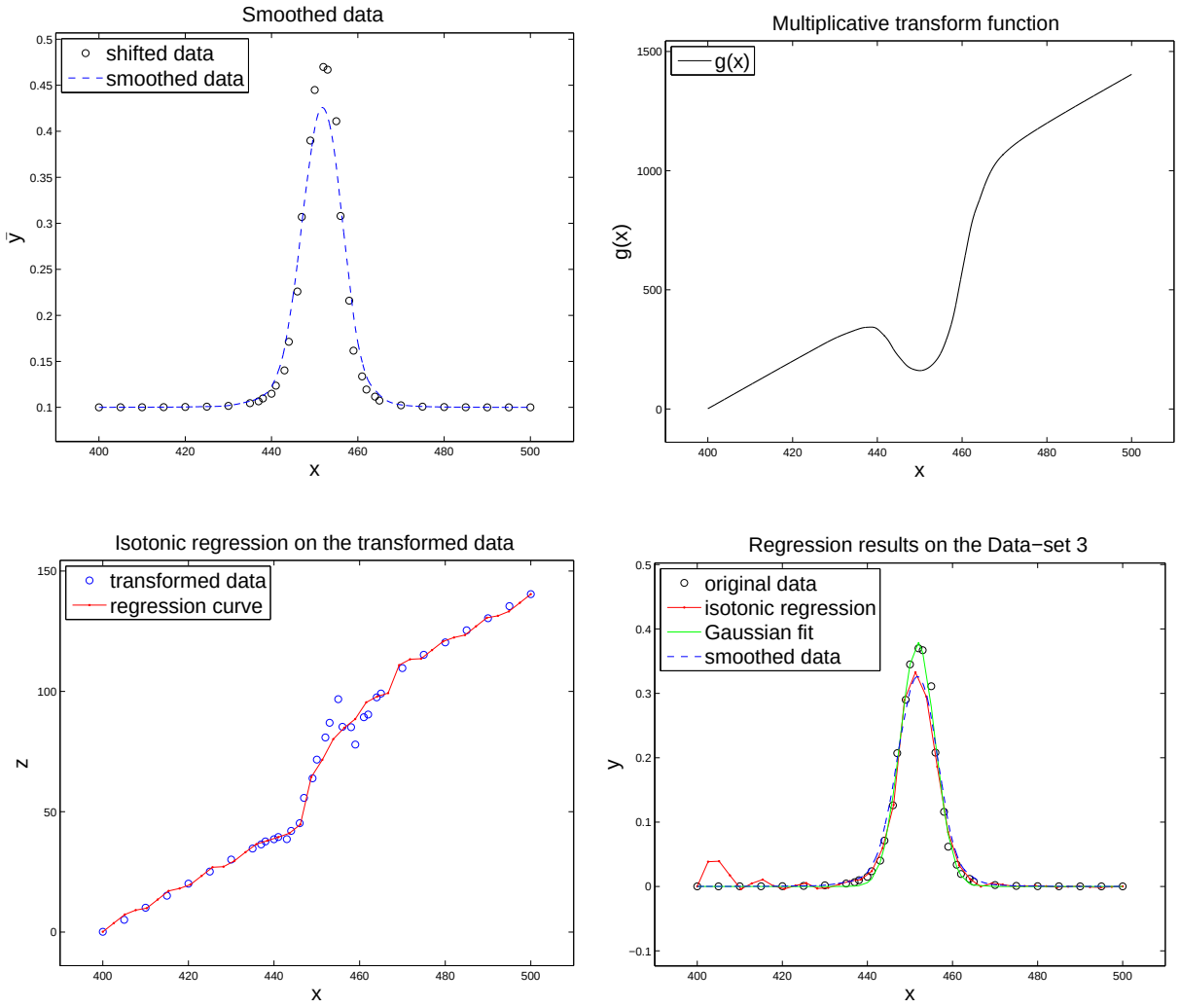



FIG. 3.4. *Multiplicative transform learnt by the* Learning rule 1 *and isotonic regression applied to the* Data-set 3*: Result of smoothing, multiplicative transform function, regression on monotonized data and comparison to Gaussian fit.*

becomes:

$$g_i(2\bar{y}_i - \bar{y}_{i-1}) > g_{i-1}\bar{y}_i. \tag{3.8}$$

The last inequality may be used to learn the value $g_i$ of the look-up table that represents the non-linear transfer function of the multiplicative transform neural system recursively from the value $g_{i-1}$, upon setting $g_1 = 1$, on the basis of the training data-set only. The corresponding **Learning rule 2** reads:

$$g_i = M\frac{g_{i-1}\bar{y}_i}{2\bar{y}_i - \bar{y}_{i-1}}, \tag{3.9}$$

with $M > 1$. The above neural learning rule is well-defined as long as the $\bar{y}$-data satisfy the condition $2\bar{y}_i - \bar{y}_{i-1} \neq 0$ for every value of the index $i$, which may be ensured by shifting the data upward some further.

Care should be taken that the learning rule (3.9) may learn an unsuitable transfer function when it happens that $2\bar{y}_i - \bar{y}_{i-1} < 0$ for certain values of the index $i$, which results in the sign of the neural LUT value $g_i$ differ from the sign of the LUT value $g_{i-1}$. When a value $g_i$ turns negative, the transformed data trend becomes locally monotonically decreasing instead of increasing, until the function $g$ changes sign again. In order to correct such sign-flipping errors, the $y$-data is lifted upward of a sufficient amount to ensure that $2\bar{y}_i - \bar{y}_{i-1} > 0$ for every value of the index $i$. A serious numerical drawback of the learning rule (3.9), which was verified through numerical tests and that might be a source of numerical instability, is that the denominator $2\bar{y}_i - \bar{y}_{i-1}$ might result very close to 0. In order to correct such unwanted effect, the constraint $2\bar{y}_i - \bar{y}_{i-1} > 1$ for every index $i$ may be put into effect: Since the denominator $2\bar{y}_i - \bar{y}_{i-1}$ in the learning rule (3.9) will always take values larger than 1, the values $g_i$ of the neural transfer function will not increase abruptly. In order to meet such requirement, the data-points $\bar{y}_i$ are replaced by the lifted-upwards values $\bar{y}_i + c_2$, where the constant $c_2$ is selected by requiring that $2(\bar{y}_i + c_2) - (\bar{y}_{i-1} + c_2) = 2\bar{y}_i - \bar{y}_{i-1} + c_2 > 1$. For instance, one may choose $c_2 = 1 - \min_i\{2\bar{y}_i - \bar{y}_{i-1}\} + \epsilon_2$, with $\epsilon_2 > 0$ Recall that the lifted-upward $y$-data must satisfy two requirements, namely:

$$\begin{array}{ll} \bar{y}_i = y_i + c, & \text{and } \ c = -\min_i\{y_i\} + \epsilon_1, \\ \bar{\bar{y}}_i = \bar{y}_i + c_2, & \text{and } c_2 = 1 - \min_i\{2\bar{y}_i - \bar{y}_{i-1}\} + \epsilon_2. \end{array}$$

The constant $\epsilon_1 > 0$ is set to a low-valued positive number. The constant $\epsilon_2$ is set to a number greater than, or equal to, 1 so that the denominator in equation (3.9) be larger than, or equal to, 1.

After completing the learning phase, the multiplicative transform LUT neural system outputs the monotonized training set $\{(x_i, z_i)\}$, with $z_i = \bar{\bar{y}}_i g_i$. Thanks to the structure of the Learning rule 2, the transformed set $\{(x_i, z_i)\}$ appears as strictly monotonically increasing. Isotonic regression is performed on the transformed training set and a look-up table $\{(\hat{x}_i, \hat{z}_i)\}$ is computed. In order to bring back the regressed curve to its original domain, it is necessary to inverse-transform such look-up table to the original domain. To this aim, a new function $\hat{g}$ is learned, by linear interpolation of the values of the function $g$ over the grid points $\hat{x}_i$ and the values $\hat{t}_i = \hat{z}_i / \hat{g}_i$ are put out, after removing the constant lift in the data, to obtain the final regression result.

In the numerical experiments with on the three training sets, the number of grid points to be included in the regression look-up table was set to 100. The Figure 3.5 summarizes the results of applying the Learning rule 2: In particular, it shows the learnt multiplicative transform function (left-hand column) and the result of regression (right-hand column). The test on the Data-set 1 shows that, as the learnt multiplicative transformation is based on the original data without smoothing, the Learning rule 2 can capture the sudden change of trend in the training data. Likewise, for the Data-set 3, the Learning rule 2 can approximate the general trend in the data-set, except for some errors caused by the interpolation in the isotonic regression process. However, in the case of the Data-set 2, the shape of the obtained regression line shows some undesirable abrupt changes, which are mainly due to the noise component in the training set. In fact, a closer examination of the numerical behavior of the Learning rule 2 reveals that, as the multiplicative transform function learnt by the Learning rule 2 is based on each individual data-point, the multiplicative transform function $g$ must counter the noise in the data in order for the transformed data to become strictly monotonic. As a result, the learnt transform function $g$ results noisy. When reverse-transforming the regression result, the noise introduced in the function $g$ by the learning rule is transferred to the final regression result.

**3.3. Learning rule 3: Data-wise multiplicative transform, additive update.** The learning rule discussed in the present Subsection differs from the Learning rule 2 in how the multiplicative transform function $g$ is learnt recursively from the training set. Instead of the multiplicative update rule (3.9), the following additive **Learning rule 3** is discussed:

$$g_i = \frac{g_{i-1}\bar{\bar{y}}_i}{2\bar{y}_i - \bar{y}_{i-1}} + \epsilon_3. \tag{3.10}$$

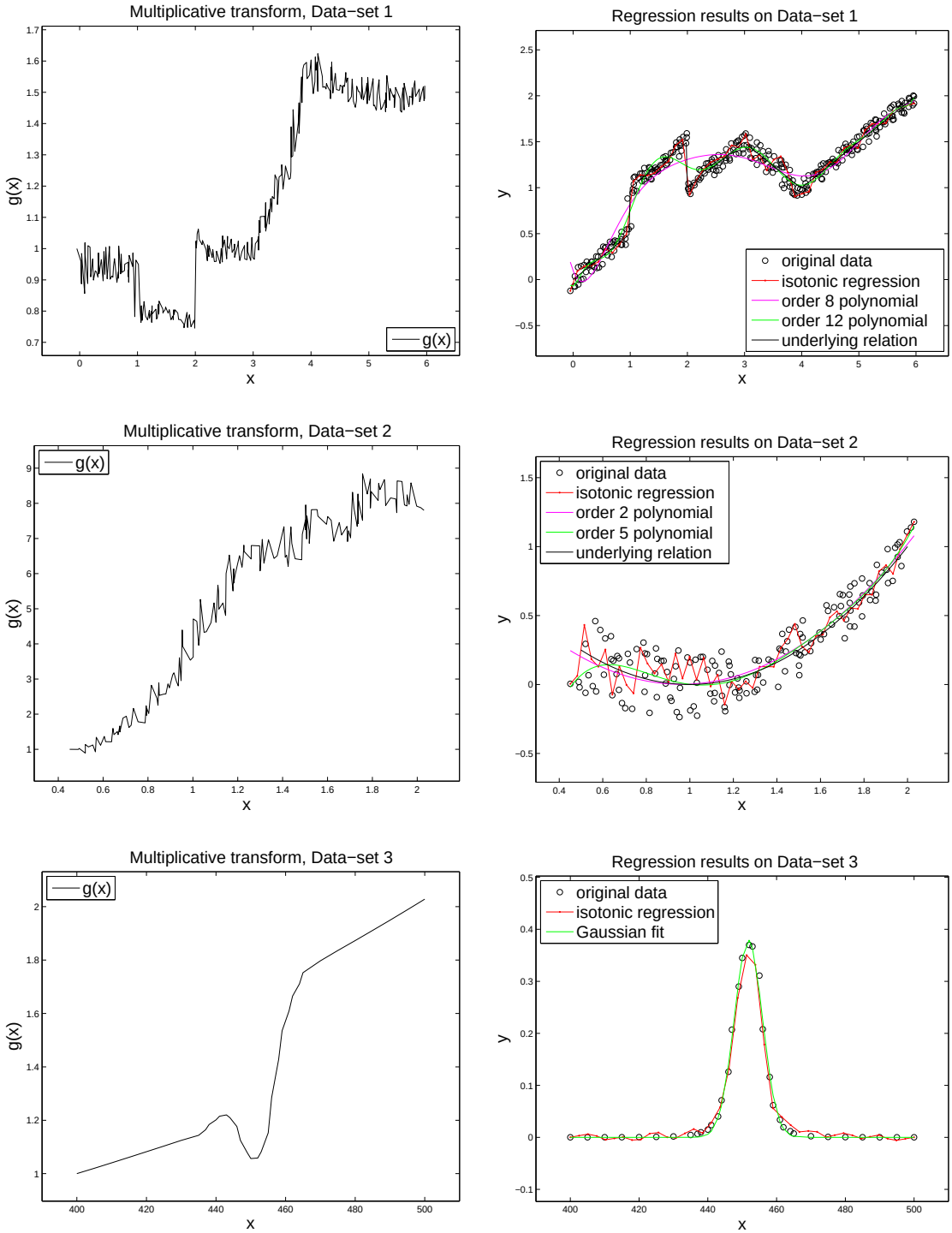



FIG. 3.5. *Multiplicative transform and isotonic regression using* Learning rule 2*: Multiplicative transform function, result of regression on monotonized data and comparison to polynomial fit (for the Data-sets 1 and 2) and Gaussian fit (for the Data-set 3).*

The constant $\epsilon_3 > 0$ is set to a low value. Since it was verified numerically that the learning scheme (3.10) exhibits the same sign-flipping problem as well as the problem of sudden increase of the values of the neural transfer function $g$ mentioned apropos the Learning rule 2, the same shifting-upward pre-processing of the $y$-data in the training set needs to be performed as for the Learning rule 2.

The Figure 3.6 summarizes the results obtained by applying the Learning rule 3 to the adaptation of the LUT neural system that implements the multiplicative data transform that makes the data-set exhibit a monotonic dependence. In particular, such Figure shows the learnt multiplicative transform function (left-hand column) and the result of regression (right-hand column). Similarly to what already observed apropos the Learning rule 2, the multiplicative transform function learnt by the Learning rule 3 is based on each individual data point in the training set without pre-smoothing, therefore, the advantages and disadvantages of the Learning rule 2 are likewise reflected into the Learning rule 3.

**3.4. Comparative discussion of the numerical results.** The Learning rule 1 uses smoothing to pre-process the training set before learning the multiplicative transform function. In particular, the Learning rule 1 uses spline interpolation to learn a fine curve to represent the smoothed data. On the other hand, the Learning rule 2 and the Learning rule 3 do not invoke smoothing: In order to transform the original training set into a monotonically increasing data-set, the multiplicative transform function $g$ is learnt in such a way that it counters any decrease in the $y$-value.

To what concerns the multiplicative transform function $g$, in the context of the Learning rule 1, it is learnt by estimating its derivative over grid points. The Learning rules 2 and 3 belong to the same category of adaptation schemes as they both use an Euler approximation to solve the differential inequality (3.1). The difference between the Learning rule 2 and the Learning rule 3 is that the multiplicative data-monotonization function $g$ is estimated by a multiplicative update scheme in the context of the Learning rule 2, whereas it is estimated by an additive update scheme in the context of the Learning rule 3.

The proposed learning rules to perform data monotonization and statistical regression have been compared with function-based regression methods such as polynomial-fit and Gaussian-fit in order to demonstrate the relative advantages and disadvantages of the proposed regression approach.

As shown in the Figures 3.2, 3.5 and 3.6, polynomial regression was tested on Data-set 1 to get a term of comparison with the proposed neural regression method. The best order of the polynomial determined by the application of the Akaike Information Criterion (AIC) [7] for the Data-set 1 was 12. The Figures show that neural statistical regression through the Learning rule 1 fits the training set better than polynomial regressions. In particular, such example shows that neural statistical regression is advantageous over function-based regression for those data-sets that are not bound to functions.

The Figures 3.3, 3.5 and 3.6 show the result of polynomial regression compared with the results obtained by the proposed neural regression methods on the Data-set 2. The best order of polynomial determined by the application of the AIC for the Data-set 2 was 5 as it fits better to the data compared to other orders. Since the underlying function of Data-set 2 is $y = (x-1)^2$, polynomial regression of order 2 was tested as well, for comparison purpose. From the results of regression, in can be readily appreciated that the Learning rule 1 performs similarly to the polynomial regression and, since the function underlying the data is smooth, the regression procedure based on the Learning rule 1 exhibits good performances.

The Figures 3.4, 3.5 and 3.6 show regression results for the Data-set 3 (which does not obey any polynomial function model). In this case, regression with a Gaussian function was tested against the developed neural regression method for comparison purpose. Both the proposed neural statistical regression and the Gaussian-fit regression are able to construct a curve close to the real-world data, with the regression method based on the Learning rule 2 exhibiting a closer fit.

In general, the neural isotonic statistical regression method, extended to non-monotonic data, proves advantageous over function-based regression when the underlying functional dependency is unknown or difficult to determine. In addition, the proposed learning rules can adapt to different data shapes while maintaining a relatively good fit. In those cases when the underlying functional dependency is known, the neural statistical regression still performs similarly to the function-based regression (in other terms,

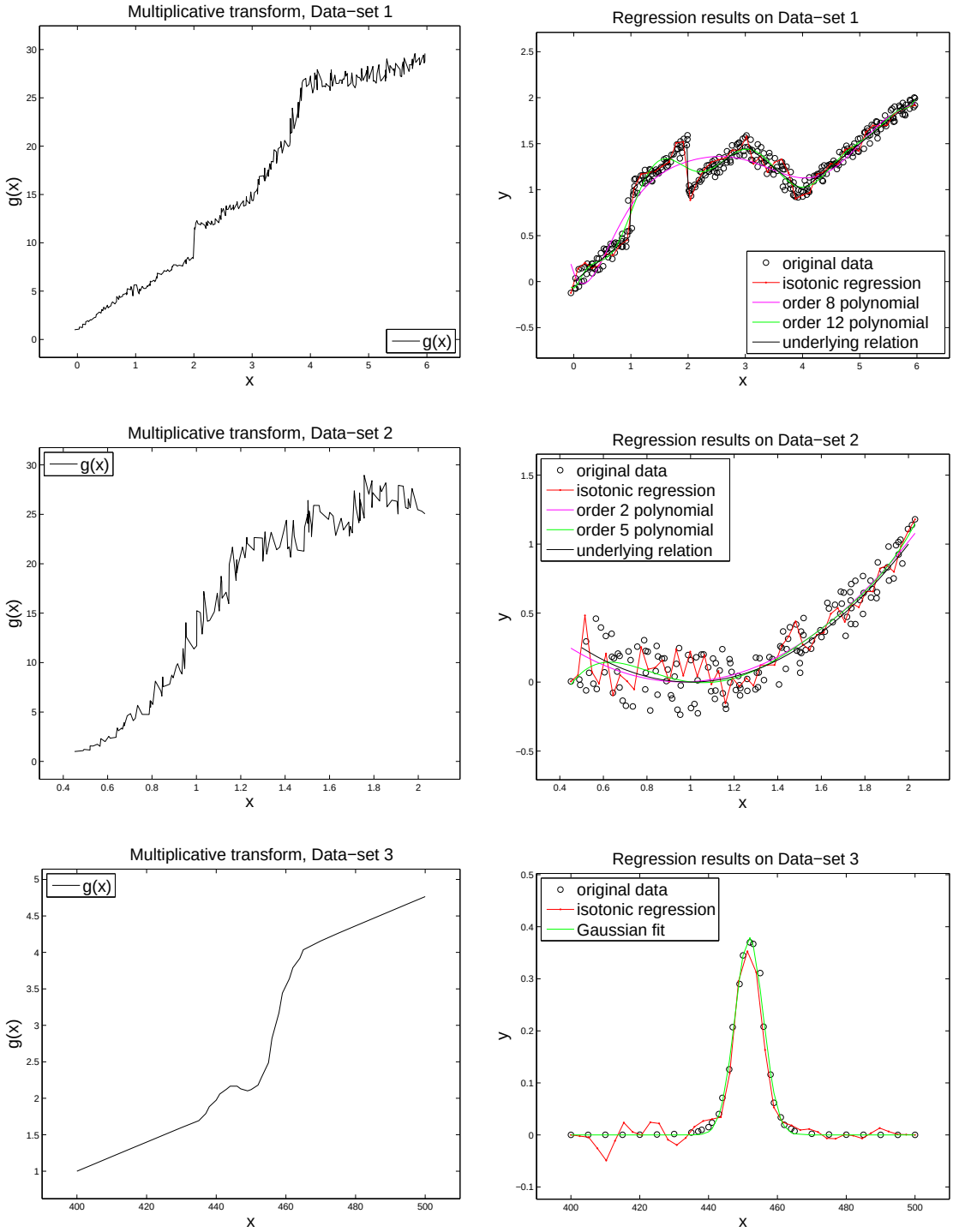



FIG. 3.6. *Multiplicative transform and isotonic regression using* Learning rule 3*: Multiplicative transform function, result of regression on monotonized data and comparison to polynomial fit (for the Data-sets 1 and 2) and Gaussian fit (for the Data-set 3).*

*unsupervised learning performs similarly to informed modeling*).

**4. Conclusions and future work.** We have proposed three neural learning rules to adapt a non-linear LUT-based neural system that is capable of transforming a non-monotonic training set into an approximately-monotonic data-set in order to enable a previously-developed isotonic regression algorithm to be applied to bivariate data that exhibit a non-monotonic dependency. While the Learning rule 1 requires a pre-estimation (although rough) of the model to be learned, the Learning rules 2 and 3 do not need such preliminary pre-processing. In fact, the Learning rule 1 is to be taken as a prototypical method to explain the fundamental concepts behind the present research endeavor, while the Learning rules 2 and 3 are to be considered our contribution to the field of neural statistical regression.

The proposed notion of multiplicative transform allowed us to extend the simplicity and the unrestricted model shape of isotonic statistical regression to a wide range of non-monotonic data. Although the isotonic statistical regression method presented in [14, 15] can be applied as is to monotonized data, two fundamental features of the neural isotonic statistical regression developed earlier could not be recovered, namely, the viability of regression in presence of unpaired measurements in a training set and the viability of regression on incomplete data-sets (namely, on data-sets with missing records). While, at the present stage of research, it appears that pairing is an essential feature to perform regression on non-monotonic data, it is possible to envisage that the restriction on the absence of missing data could be overcome by putting into effect a data-imputation technique.

Currently, we have developed artificial-neural learning rules for bivariate isotonic statistical regression. In applications, bivariate data are very frequently encountered, either because the underlying model is described in terms of two variable, or whenever a multi-variable problem may be reduced in complexity to a two-variable problem. In a wide range of applications, however, modeling a multi-variable phenomenon is likewise necessary. For instance, the case of one variable that depends of two independent variables encompasses already a wise range of real-world data-sets (see, for instance, the summary presented in the recent papers [16, 17], that discussed a *trivariate isotonic regression* problem). In the context of neural statistical regression, it seems unlikely that more than three or four variables could be simultaneously coped with, since the number of data-records necessary to estimate their joint probability density functions grows very quickly with the number of involved variables. In the future, we will work toward an extension of the presently-proposed learning methods to deal with non-monotonic statistical trivariate regression by LUT neural networks with multiple inputs.

REFERENCES


[1] A. Abdullah and A. Hussain, "A cognitively inspired approach to two-way cluster extraction from one-way clustered data", *Cognitive Computation*, June 2014

[2] S. Ahuja, A. Lakshminarayana and S.K. Shukla, "Statistical regression based power models", in *Low Power Design with High-Level Power Estimation and Power-Aware Synthesis*, pp. 59 – 70, Springer New York, 2012

[3] S. Angelov, B. Harb, S. Kannan and L.-S. Wang, "Weighted isotonic regression under the $L_1$ norm", in *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), pp. 783 – 791, 2006

[4] J.L. Blanco, L.A. Hernández, R. Fernández and D. Ramos, "Improving automatic detection of obstructive sleep apnea through nonlinear analysis of sustained speech", *Cognitive Computation*, Vol. 5, pp. 458 – 472, 2013

[5] L. Breiman, "Statistical modeling: the two cultures", *Statistical Sciences*, Vol. 16, No. 3, pp. 199 – 231, 2001

[6] H.D. Brunk, "Maxmimum likelihood estimates of monotone parameters", *Annals of Mathematical Statistics*, Vol. 26, pp. 607 – 616, 1955

[7] K.P. Burnham and D.R. Anderson, "Multimodel inference: understanding AIC and BIC in model selection", *Sociological Methods and Research*, Vol. 33, No. 2, pp. 261 – 304, 2004

[8] N. Chakravarti, "Sensitivity analysis in isotonic regression", *Discrete Applied Mathematics*, Vol. 45, No. 3, pp. 183 – 196, September 1993

[9] Á. Calzada Defez and J.C. Socoró Carrie, "Voice quality modification using a harmonics plus noise model", *Cognitive Computation*, Vol. 5, pp. 473 – 482, 2013

[10] J.S. Domínguez-Menchero and G. González-Rodríguez, "Analyzing an extension of the isotonic regression problem", *Metrika*, Vol. 66, No. 1, pp. 19 – 30, July 2007

[11] K. Eckerle, "Circular interference transmittance study," Report of the *National Institute of Standards and Tech-*

*nology (NIST), US Department of Commerce, USA*. Unpublished report, 1979. Data publicly available from the repository `http://www.itl.nist.gov/div898/strd/nls/data/eckerle4.shtml`

[12] A. Ezeiza, K. López de Ipiña, C. Hernández and N. Barroso, "Enhancing the feature extraction process for automatic speech recognition with fractal dimensions", *Cognitive Computation*, Vol. 5, pp. 545 – 550, 2013

[13] S. Fiori, "Hybrid independent component analysis by adaptive LUT activation function neurons", *Neural Networks*, Vol. 15, No. 1, pp. 85 – 94, January 2002

[14] S. Fiori, "Statistical nonparametric bivariate isotonic regression by look-up-table-based neural networks", in *Proceedings of the 2011 International Conference on Neural Information Processing* (ICONIP 2011, Shanghai (China), November 14-17, 2011), B.-L. Lu, L. Zhang, and J. Kwok (Eds.), Part III, LNCS 7064, pp. 365 – 372, Springer, Heidelberg, 2011

[15] S. Fiori, "Fast statistical regression in presence of a dominant independent variable", *Neural Computing and Applications*, (Special issue of the 2011 International Conference on Neural Information Processing - ICONIP'2011), Vol. 22, No. 7, pp. 1367 – 1378, 2013

[16] S. Fiori, "An isotonic trivariate statistical regression method", *Advances in Data Analysis and Classification*, Vol. 7, No. 2, pp. 209 – 235, June 2013

[17] S. Fiori, "A two-dimensional Poisson equation formulation of non-parametric statistical non-linear modeling", *Computers and Mathematics with Applications* (Elsevier), Vol. 67, No. 5, pp. 1171 – 1185, March 2014

[18] D.R. Forrest, R.D. Hetland and S.F. DiMarco, "Multivariable statistical regression models of the areal extent of hypoxia over the Texas-Louisiana continental shelf", *Environmental Research Letters*, Vol. 6, No. 4, 045002 (10 pp), October-December 2011

[19] J. Friedman and R. Tibshirani, "The monotone smoothing of scatterplots", *Technometrics*, Vol. 26, No. 3, pp. 243 – 250, 1984

[20] P. Gómez-Vilda, V. Rodellar-Biarge, V. Nieto-Lluis, C. Munöz-Mulas, L.M. Mazaira-Fernández, R. Martínez-Olalla, A. Álvarez-Marquina, C. Ramírez-Calvo and M. Fernández-Fernández, "Characterizing neurological disease from voice quality biomechanical analysis", *Cognitive Computation*, Vol. 5, pp. 399 – 42, 2013

[21] P. Hall and L.-S. Huang, "Nonparametric kernel regression subject to monotonicity constraints", *Annals of Statistics*, Vol. 29, No. 3, pp. 624 – 647, June 2001

[22] K. Jaqaman and G. Danuser, "Linking data to models: data regression", *Nature Reviews Molecular Cell Biology*, Vol. 7, pp. 813 – 819, November 2006

[23] M.A. Kulkarni, S. Patil, G.V. Rama and P.N. Sen, "Wind speed prediction using statistical regression and neural network", *Journal of Earth and System Sciences*, Vol. 117, No. 4, pp. 457 – 463, August 2008

[24] X. Li and H.-Z. Liu, "Statistical regression for efficient high-dimensional modeling of analog and mixed-signal performance variations", in *Proceedings of the* $45^{\text{th}}$ *ACM/IEEE Design Automation Conference* (DAC 2008, Anaheim Convention Center, California, USA, June 9-13, 2008), pp. 38 – 43, June 2008

[25] S. Liu, R.X. Gao, Q. He, J. Staudenmayer and P. Freedson, "Development of statistical regression models for ventilation estimation", in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (EMBC, September 3-6, 2009), pp. 1266 – 1269, 2009

[26] J. Liu and H. Li, "Application research of a statistical regression algorithm in the IVR system", in *Proceedings of the 2010 International Conference on Educational and Network Technology* (ICENT, Qinhuangdao (China), June 25-27, 2010), pp. 358 – 360, 2010

[27] M. Naji, M. Firoozabadi and P. Azadfallah, "Classification of music-induced emotions based on information fusion of forehead biosignals and electrocardiogram", *Cognitive Computation*, Vol. 6, pp. 241 – 252, 2014

[28] A. Papoulis, *Probability and Statistics*, Prentice Hall, 1996

[29] K. Punera and J. Gosh, "Enhanced hierarchical classification via isotonic smoothing", in *Proceedings of the* $17^{\text{th}}$ *International Conference on World Wide Web* (April 21-25, 2008, Beijing – China), pp. 151 – 160, 2008

[30] J. O. Ramsay, "Monotone regression splines in action", *Statistical Science*, Vol. 3, No. 4, pp. 425 – 441, November 1988

[31] I. Siegert, D. Philippou-Hübner, K. Hartmann, R. Böck and A. Wendemuth, "Investigation of speaker group-dependent modelling for recognition of affective states from speech", *Cognitive Computation*, August 2014

[32] D.E. Specht, "A general regression neural network", *IEEE Transactions on Neural Networks*, Vol. 2, No. 6, pp. 568 – 576, November 1991

[33] D.N. Tam, "Computation in emotional processing: Quantitative confirmation of proportionality hypothesis for angry unhappy emotional intensity to perceived loss", *Cognitive Computation*, Vol. 3, pp. 394 – 415, 2011

[34] T.P. Tran, E. Ong, A.P. Hodges, G. Paternostro and C. Piermarocchi, "Prediction of kinase inhibitor response using activity profiling, *in vitro* screening, and elastic net regression", *BMC Systems Biology*, Vol. 8, No. 74, pp. 1 – 10, 2014

[35] X. Yan, "Dissociated emergent response system and fine-processing system in human neural network and a heuristic neural architecture for autonomous humanoid robots", *Cognitive Computation*, Vol. 3, pp. 367 – 373, 2011

[36] X. Zhu, M. Welling, F. Jin and J. Lowengrub, "Predicting simulation parameters of biological systems using a Gaussian process model", *Statistical Analysis and Data Mining: The ASA Data Science Journal* (Special Issue: "Best Papers from the SLDM Competition"), Vol. 5, No. 6, pp. 509 – 522, December 2012

[37] L. Zou, S. Xu, Z. Ma, J. Lu and W. Su, "Automatic removal of artifacts from attention deficit hyperactivity disorder electroencephalograms based on independent component analysis", *Cognitive Computation*, Vol. 5, pp. 225 – 233, 2013

**Appendix A. Bivariate isotonic regression method.** The regression model to infer has a non-linear structure described by relationship $y = f(x)$, where $x \in \mathcal{X} \subseteq \mathcal{R}$ denotes the input random variable, having probability density function $p_x(\cdot)$, and $y \in \mathcal{Y} \subseteq \mathcal{R}$ denotes the output random variable, having probability density function $p_y(\cdot)$.

With the assumption that the regression model be *strictly monotonic*, namely $f'(x) > 0$ or $f'(x) < 0$, $\forall x \in \mathcal{X}$, the input-output distributions and the regression model may be shown to stay in the relationship:

- **Positive-slope regression model**: $f(x) = P_y^{-1}(P_x(x))$,
- **Negative-slope regression model**: $f(x) = P_y^{-1}(1 - P_x(x))$,

where the symbol $P_y^{-1}(\cdot)$ denotes the inverse of the cumulative distribution function $P_y(\cdot)$ pertaining to the $y$ random variable and the symbol $P_x(\cdot)$ denotes the cumulative distribution function pertaining to the $x$ random variable. In effect, the above transformations make sure that the distribution $p_x(\cdot)$ is transformed into the distribution $p_y(\cdot)$ according to the law of measure-invariance of probability density functions [28].

The above regression method does not depend explicitly on the data, but on the probability density/cumulative functions obtained from each data set separately.

To develop a fully-numerical bivariate isotonic statistical regression method, the following ingredients are of use: A suitable numerical estimation method for cumulative density functions and a suitable format for function representation/handling (with particular emphasis on numerical function inversion). In order to put the above regression equations into effect, a representation of the quantities of interest based on look-up-tables was chosen.

A real-valued look-up table with $N$ entries is represented by a pair $\mathrm{LUT} = (x, y)$, where $x \in \mathcal{R}^N$ and $y \in \mathcal{R}^N$. The entries $x_k$ of vector $x$ and the entries $y_k$ of vector $y$, with $k \in \{1, \ldots, N\}$, are paired and provide a point-wise description of an arbitrarily-shaped function. In order to handle the look-up tables for statistical regression purpose, the following operations are of use:

- **Histogram computation**: In order to numerically approximate the probability distribution of a data set $\mathcal{D}$, a histogram operator is made use of. The histogram-computation operation is denoted by $(x, y) = \mathrm{hist}(\mathcal{D})$. The constructed look-up table is built up as follows: $x_k$ equals the value of the $k^{\mathrm{th}}$ bin center, $y_k$ equals the number of data-points falling in the $k^{\mathrm{th}}$ bin.
- **Cumulative-sum computation**: On the basis of a look-up table $(x, y)$, a new look-up table $(x, v) = \mathrm{csum}(x, y)$ is constructed, where array $v$ contains the cumulative sum of the entries of array $y$, possibly normalized in order to approximate the numerical integration of the function represented by the look-up table $(x, y)$. In its unnormalized version, the cumulative sum is described by $v_1 = y_1$ and $v_k = v_{k-1} + y_k$ for $2 \leq k \leq N$.
- **Function/table interpolation**: Interpolation may be invoked to make computations with look-up tables on other points in the domain. In the present context, it is necessary to preserve the monotonicity of an approximated function, therefore linear interpolation only is made use of. Denote by $\mathcal{D}$ the $x$-coordinate point-set, where the function represented by a look-up table $(x, y)$ needs to be interpolated. The interpolation operation may be denoted by $\mathcal{I} = \mathrm{interp}(x, y, \mathcal{D})$, where the set $\mathcal{I}$ contains the interpolated $y$-values corresponding to the $x$-values in the set $\mathcal{D}$. Because of the hypotheses of monotonicity of the model underlying the data, a set-type representation is equivalent to an ordered-list representation.
- **Function/table inversion**: If a function is given a point-wise representation by the help of a look-up table $(x, y)$, then its inverse function may be easily given a point-wise representation by the look-up table $(y, x)$, namely, function inversion is equivalent to swapping look-up table's arguments. *Apparently, therefore, function inversion in the context of look-up tables representation is a computationally-costless operation.*

The bivariate isotonic regression method proposed in [15] consists of the following steps.

First, it is necessary to estimate the probability density functions of data within $\mathcal{X}$ and $\mathcal{Y}$ data sets. They may be numerically estimated – up to scale factors – by:

$$(x, p_x) = \text{hist}(\mathcal{X}),\ (y, p_y) = \text{hist}(\mathcal{Y}). \tag{A.1}$$

The numerical cumulative distribution functions of the $\mathcal{X}$ and $\mathcal{Y}$ data sets are estimated by numerical integration of the numerical probability density functions, which is achieved by the help of the cumulative-sum operator applied to look-up tables $(x, p_x)$ and $(y, p_y)$:

$$(x, P_x) = \text{csum}(x, p_x),\ (y, P_y) = \text{csum}(y, p_y). \tag{A.2}$$

If the statistical model has negative slope, the look-up table $(x, P_x)$ should be replaced by $(x, 1 - P_x)$ in what follows.

For regression purpose, the regression model needs to be evaluated on a ordered set of $x$-points denoted here as an array $\hat{x}$. The array $\hat{x}$ consists of $R$ points equally spaced over the range of interest for the $x$-variable.

The last step consists in numerically evaluating the quantity $P_x(\cdot)$ over the points in $\hat{x}$ and then in evaluating the function $P_y^{-1}(\cdot)$ over the values of $P_x(\hat{x})$, namely:

$$\mathcal{P}_x = \text{interp}(x, P_x, \hat{x}),\ \hat{y} = \text{interp}(P_y, y, \mathcal{P}_x). \tag{A.3}$$

In the second of equations (A.3), the inverse function $P_y^{-1}(\cdot)$ appears through the swapped look-up table of function $P_y(\cdot)$.

The pair $(\hat{x}, \hat{y})$ provides a look-up table representation for the non-linear bivariate isotonic regression model $f(\cdot)$.

Full details about the above-summarized statistical isotonic regression technique are available in [15].