Non Linear Fitting Methods for Machine Learning

Edgar A. Martínez-García[1(✉)], Nancy Ávila Rodríguez[2], Ricardo Rodríguez-Jorge[1], Jolanta Mizera-Pietraszko[3], Jaichandar Kulandaidaasan Sheba[4], Rajesh Elara Mohan[5], and Evgeni Magid[6]

[1] Universidad Autónoma de Ciudad Juárez, Ciudad Juárez, Mexico,
edmartin@uacj.mx
[2] University of Texas at El Paso, El Paso, USA
nsavila@miners.utep.edu
[3] Opole University, Opole, Poland
jolanta.mizera-pietraszko@pwr.edu.pl
[4] Singapore Polytechnic, Singapore, Singapore
jai@sp.edu.sg
[5] Singapore University of Technology and Design, Singapore, Singapore
rajesh.elara@sutd.edu.sg
[6] Kazan Federal University, Kazan, Russian Federation
evgeni.magid@kfu.edu.ru

Abstract. This manuscript presents an analysis of numerical fitting methods used for solving classification problems as discriminant functions in machine learning. Non linear polynomial, exponential, and trigonometric models are mathematically deduced and discussed. Analysis about their pros and cons, and their mathematical modelling are made on what method to chose for what type of highly non linear multi-dimension problems are more suitable to be solved. In this study only deterministic models with analytic solutions are involved, or parameters calculation by numeric methods, which the complete model can subsequently be treated as a theoretical model. Models deduction are summarised and presented as a survey.

1 Introduction

So far today, modern intelligent machines are becoming part of numberless fields of application. Learning must be an implicit function inherent in intelligent control systems to enhance industrial robots, face and voice recognition systems, self-driving vehicles, artificial vision, domotics, air crafts control, mechatronic systems and so forth. Intelligent machines relay on their capabilities of perception and sensing, planning, acting, and gradually increasing the capability for learning. Machine learning inherently develops critical functions which may be

F. Xhafa et al. (eds.), *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*,
Lecture Notes on Data Engineering and Communications Technologies 13,
https://doi.org/10.1007/978-3-319-69835-9_76

summarised as patterns extraction, classification and recognition [3]. Recognition issues are also referred to as discrimination functions. Regardless a classification and training process is supervised or automatic, it needs to minimise the learning errors and enable itself to recover from faults. A classifier's purpose is to enhance multiple class separability, and reduce data sparseness in an information hyperspace. Discriminant functions with enough degree of dimension must warranty non linearity to recognise any data class [7]. Since a major common tasks is fitting a model for data class discrimination, the concepts of over-fitting and under-fitting are relevant [1,8,9]. A too complex model that has been over-fitted (usually non bijective mapping) has too many parameters w.r.t. the number of observations. It produces poor performance overreacting to minor fluctuations in the training data. Under-fitting occurs when a machine learning algorithm cannot track the underlying trend of the data, such as when fitting a linear model to non-linear data, having poor performance. This work concerns the mathematical analysis of non linear fitting functions $f(\mathbf{x}_i)$ that are used for class discrimination $\mathbf{d}_i$ during a machine learning process [2]. In this study only deterministic models with analytic solutions are involved, or parameters calculation by numeric methods, where the complete model can subsequently be treated as a theoretical model. Let $g(\mathbf{d}_i)$ be an arbitrary data model that is fitted by the model $f(\mathbf{x}_i)$ such that $f(\mathbf{x}_i) \approx g(\mathbf{d}_i)$. For empirical data $\mathbf{d} \in \mathbb{R}^d$ such that $\mathbf{d}_i = (x_1, x_2, \ldots, x_d)^\top$. To treat the data as a bijective function such that $\mathbf{f} : \mathbf{x}_i \rightarrow \mathbf{d}_i = f(\mathbf{x}_i)$, then the Cartesian origin of the function $f(\mathbf{x})$ is transformed with angles $\boldsymbol{\lambda}$ for any fitting convenience through

$$\mathbf{d}'_i = \mathbf{R}_d(\boldsymbol{\lambda}) \cdot \mathbf{d} \tag{1}$$

and Cartesian translation in a multidimensional space, by using the translation vector $\boldsymbol{\tau} = (\Delta x_1, \Delta x_2, \cdots \Delta x_d)^\top$

$$\mathbf{d}'_i = \mathbf{R}_d(\boldsymbol{\lambda}) \cdot \mathbf{d}_i + \boldsymbol{\tau} \tag{2}$$

For instance, for a 3D case, let us assume that $\mathbf{R}_3 = \mathbf{R}_1\mathbf{R}_2\mathbf{R}_3$, and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)^\top$

$$\mathbf{d}' = \mathbf{R}_1(\lambda_1) \cdot \mathbf{R}_2(\lambda_2) \cdot \mathbf{R}_3(\lambda_3) \cdot \mathbf{d} + \boldsymbol{\tau}_d \tag{3}$$

For any transformed data into a bijective system, the discrimination function is obtained $f(\mathbf{x}) = g(\mathbf{d})$, where now the new theoretical function is re-fitted in the discrimination space

$$f(\mathbf{x}) = \mathbf{R}_3^{-1}(\lambda_3) \cdot \mathbf{R}_2^{-1}(\lambda_2) \cdot \mathbf{R}_1^{-1}(\lambda_1) \cdot (f'(\mathbf{x}) - \boldsymbol{\tau}_d) \tag{4}$$

In Sects. 2, and 3 the exponential and polynomial methods to fit high order continuous non linear data are described. In Sect. 4 an interpolation method for high non linear order with discontinuities is described. In Sect. 5 a non polynomial fitting method for non linear functions is analysed. Subsequently, Sect. 6 treats the topic of transformation of trigonometric functions into purely polynomial functions. Section 7 discusses the topic of multivariate polynomials. Section 8 describes the multi-layer perceptron training method, which is naturally fitting model non linear highly multi-dimension data. Finally in Sect. 9 conclusion is summarised.

2 Exponential and Polynomial Regression

When data $(x_i, f(x_i))$ model an exponential discrimination function with amplitude a, and shape form b, such that

$$f(x) = ae^{bx} \tag{5}$$

We firstly may algebraically manipulate such a general model, by applying the Napierian logarithm in both sides of the equation,

$$\ln f(x) = \ln\left(\mathrm{e}^{bx}\right) \tag{6}$$

thus, developing the logarithm laws, we find that

$$\ln(f(x)) = \ln(a) + bx \tag{7}$$

We linearise the equation by redefining and substituting $\ln(f(x)) \doteq y$, as well as $\ln(a) \doteq c$, in such a way that,

$$y = c + bx \tag{8}$$

now substituting into the previous linear function to solve parameters b, and c by linear regression based on the mean square method,

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}; \qquad c = \frac{\sum y}{n} - b\frac{\sum x}{n} \tag{9}$$

substituting logarithm-based parameters and solving its algebraic inverse

$$\mathrm{e}\left(\ln(f(x)) = c + bx\right) \tag{10}$$

since $c \doteq \ln(a)$,

$$f(x) = \mathrm{e}^c\mathrm{e}^{bx} = \exp\left(\frac{\sum \ln(f(x))}{n} - b\frac{\sum x}{n}\right)\mathrm{e}^{bx} \tag{11}$$

Now for a logarithmic function, a linear regression is developed

$$f(x) = a + b\ln(x) \tag{12}$$

and

$$z \doteq \ln(x) \tag{13}$$

hence, we temporally substitute z,

$$f(z) = a + bz \tag{14}$$

the parameters are obtained through the mean squared linear regression,

$$b = \frac{n(\sum zf(z)) - (\sum z)(\sum f(z))}{n(\sum z^2) - (\sum z)^2} = \frac{n(\sum \ln(x)f(x)) - (\sum \ln(x))(\sum f(x))}{n(\sum \ln^2(x)) - (\sum \ln(x))^2} \tag{15}$$

and

$$a = \frac{\sum f(z)}{n} - b\frac{\sum z}{n} \tag{16}$$

Substituting in z, and dropping off x from Eq. (12)

$$f(x) = \left(\frac{\sum f(x)}{n} - b\frac{\sum \ln(x)}{n}\right) + b\ln(x) \tag{17}$$

When data exhibit a significant degree of error, unlike intersecting all points, but a single curve that represent the data trend as a group, is known as Regression. Reducing such trend error by the square least allows to adjust data fitting non linear functions [12]. The empirical model $y = y_m + \epsilon$ approximates a suitable theoretical model y_m. For instance, fitting the data assuming a 2^{nd} degree polynomial by the form,

$$y_m = a_0 + a_1 x + a_2 x^2 \tag{18}$$

Thus, the sum of the squared differences yields the residual formula s_r. The residual of the squared sum of the empirical and theoretical fitting model is $s_r = (y - y_m)^2$,

$$s_r = \sum_{i=1}^{n} \left(y_i - (a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_k x_i^k)\right)^2 \tag{19}$$

By partial derivatives the rate of change of the function w.r.t. each coefficient is determined by the next three equations.

$$\frac{\partial s_r}{\partial a_0} = -2\sum_{i=1}^{n}(y_i - a_0 - a_1 x_i - a_2 x^2) \tag{20a}$$

$$\frac{\partial s_r}{\partial a_0} = -2\sum_{i=1}^{n} x_i(y_i - a_0 - a_1 x_i - a_2 x^2) \tag{20b}$$

$$\frac{\partial s_r}{\partial a_0} = -2\sum_{i=1}^{n} x_i^2(y_i - a_0 - a_1 x_i - a_2 x^2) \tag{20c}$$

Equating to zero each function and algebraically factorising them, a set of linear equations in terms of their coefficients a_i are stated for subsequent solution,

$$(n)a_0 + \left(\sum_i x_i\right) a_1 + \left(\sum_i x_i^2\right) a_2 = \sum_i y_i \tag{21a}$$

$$\left(\sum_i x_i\right) a_0 + \left(\sum_i x_i^2\right) a_1 + \left(\sum_i x_i^3\right) a_2 = \sum_i x_i y_i \tag{21b}$$

$$\left(\sum_i x_i^2\right) a_0 + \left(\sum_i x_i^3\right) a_1 + \left(\sum_i x_i^4\right) a_2 = \sum_i x_i^2 y_i \tag{21c}$$

Algebraically ordering in the matrix form as a linear system $\mathbf{v} = \mathbf{A} \cdot \mathbf{a}$,

$$\begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \\ \sum_i x_i^2 y_i \end{pmatrix} = \begin{pmatrix} n & \sum_i x_i & \sum_i x_i^2 \\ \sum_i x_i & \sum_i x_i^2 & \sum_i x_i^3 \\ \sum_i x_i^2 & \sum_i x_i^3 & \sum_i x_i^4 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \tag{22}$$

solving the system of equations by an algebraic inverse matrix method,

$$\mathbf{a} = \mathbf{A}^{-1} \cdot \mathbf{v} \tag{23}$$

for the actual quadratic problem, the coefficients are a_0, a_1, and a_2. In addition, with standardised error $s_{y/x}$, and coefficient of determination $r^2 = (s_t - s_r)/s_t$.

3 Nonlinear Interpolation

For known precise data set points, a basic procedure is to fit a series of curves crossing through each point directly. This inter-point values estimation is known as interpolation. Let us state the general form polynomial

$$f_n(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \cdots + b_n(x - x_0)\ldots(x - x_{n-1}) \tag{24}$$

where the coefficients are obtained from the next expression,

$$b_0 = f(x_0); \quad b_1 = f(x_1, x_0); \quad b_2 = f(x_2, x_1, x_0); \quad b_n = f(x_n, x_{n-1}, \ldots, x_0) \tag{25}$$

The divided differences are described by,

$$f(x_i, x_j) = \frac{f(x_i) - f(x_j)}{x_i - x_j} \tag{26}$$

likewise, the second divided difference is deduced in the next expression,

$$f(x_i, x_j, x_k) = \frac{f(x_i, x_j) - f(x_j - x_k)}{x_i - x_k} \tag{27}$$

In such a way, the n^{th} finite divided difference is

$$f(x_n, x_{n-1}, \ldots, x_1, x_0) = \frac{f(x_n, x_{n-1}, \ldots x_1) - f(x_{n-1, x_{n-2}, \ldots x_0})}{x_n - x_0} \tag{28}$$

Such differences are useful to evaluate the coefficients of Eq. (25) substituted in Eq. (24) to obtain the interpolating polynomial, as defined next:

$$\begin{aligned} f_n(x) = f(x_0) + (x - x_0)f(x_1, x_0) + (x - x_0)(x - x_1)f(x_2, x_1, x_0) + \cdots + \\ (x - x_0)\cdots(x - x_{n-1})f(x_n, \ldots, x_0) \end{aligned} \tag{29}$$

which is known as the Newton's interpolation polynomial of divided differences. To avoid the divided differences calculus, the Newton interpolation is algebraically reformulated to state the Lagrange interpolation, which is concisely represented by

$$f(x) = \sum_{i=0}^{n} [L_i(x) y_i]; \qquad \forall \qquad L_i(x) = \prod_{\substack{j=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i} \tag{30}$$

or directly written as

$$f(x) = \sum_{i=0}^{n} \left[\left(\prod_{j=0}^{n} \frac{x - x_i}{x_j - x_i} \right) y_i \right] \tag{31}$$

The general model produces a polynomial equation that fits data of degree $n-1$.

$$y_i(x_i) = a_o + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n \tag{32}$$

4 Splines Fitting

Spline functions are concatenation of different same degree polynomials allowing to fit data segments properly [10]. Yielding better error minimisation that traditional polynomial interpolations [5]. Given a function $f(x)$ that is continuously derivable and divided in n segments along the interval $[t_0, t_n]$ and interpolates data set. Thus, $f(t_i) = y_i,\ \forall\ 0 \leq i \leq n$.

$$f(x) = \begin{cases} f_1(x), & x \in [t_0, t_1] \\ f_2(x), & x \in [t_1, t_2] \\ \vdots & \\ f_{n-1}(x), & x \in [t_{n-1}, t_n] \end{cases} \tag{33}$$

where the k^{th} degree polynomial $f_i(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$ is a part of the whole function $f(x)$. The function f(x) continuity at an arbitrary point s may be defined by the next limits condition,

$$\lim_{x \to s^+} f(x) = \lim_{x \to s^-} f(x) = f(s) \tag{34}$$

For a quadratic function that is continuously derivable for the interval $[t_0, t_a]$, and interpolate the data. Thus if $f'(x)$ is continuous $z_i \equiv f'(t_i)$, and

$$f_i(x) = \frac{z_{i+1} - z_i}{2(t_{i+1} - t_i)}(x - t_i)^2 + z_i(x - t_i) + y_i \tag{35}$$

If so, we must prove the next three conditions $f_i(t_i)$, $f_i'(t_i) = z_i$, and $f'(t_{i+1}) = z_{i+1}$. These define the function f_i only for the interval $[t_i, t_{i+1}]$ as in Eq. (35).

$$z_{i+1} = -z_i + 2\left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i}\right); \qquad (0 \leq i \leq n). \tag{36}$$

And, following the Subbotin condition for interpolating beyond the interval extremes $[a, b]$, and the inner points in the sub-intervals, $f(\tau_i) = y_i\ \forall\ 0 \leq i \leq n+1$, where $\tau_0 = t_0$, $\tau_i = 0.5(t_i + t_{i+1})$, $(1 \leq i \leq n)$, $\tau_{n+1} = t_n$ with the form,

$$f_i(x) = y_{i+1} + \frac{1}{2}(z_{i+1} - z_i)(x - \tau_{i+1}) + \frac{1}{2h_i}(z_{i+1} - z_i)(x - \tau_{i+1})^2 \tag{37}$$

where $h_i = t_{i+1} - t_i$, the coefficients z_i are found by solving the next tri-diagonal system,

$$\begin{cases} 3h_0 z_0 + h_0 z_1 = 8(y_1 - y_0) \\ h_{i+1} z_{i-1} + 3(h_{i-1} + h_i) z + h_i z_{i+1} = 8(y_{i+1} - y_i) \quad (1 \leq i \leq n-1) \\ h_{n-1} z_{n-1} + 3h_{n-1} z_n = 8(y_{n+1} - y_n) \end{cases} \tag{38}$$

5 Trigonometric Functions

Non linear periodical functions are fitting methods through calculation of trigonometric function models [6], and are capable to basically approximate any function. The general Fourier model is provided next (39),

$$f(x) = \alpha_0 + \alpha_m \cos(m\omega x) + \beta_m \sin(m\omega x) \tag{39}$$

Through expressions (40) the Fourier series independent coefficient is calculated.

$$\alpha_0 = \frac{1}{T} \int_0^T f(x) dx \tag{40}$$

and the subsequent Fourier coefficients are calculated by

$$\alpha_m = \frac{2}{T} \int_0^T f(x) \cos(m\omega_0 x) \mathrm{d}x; \qquad \beta_m = \frac{2}{T} \int_0^T f(x) \sin(m\omega_0 x) \mathrm{d}x \tag{41}$$

The function is expressed with the numerical coefficients of the Fourier series.

$$\begin{aligned} f(x) = \frac{1}{T} \int_{x=0}^T f(x) dx + \left(\frac{2}{T} \int_{x=0}^T f(x) \cos(m\omega_0 x) \mathrm{d}x \right) \cos(m\omega_0 x) \\ + \left(\frac{2}{T} \int_{x=0}^T f(x) \sin(m\omega_0 x) \mathrm{d}x \right) \sin(m\omega_0 x) \end{aligned} \tag{42}$$

6 Taylor Series

Taylor's theorem provides a way of expressing a function as a power series w.r.t. the independent variable $\mathbf{x}$, but only applied to those functions that are continuous and differentiated within the $\mathbf{x}$-range of interest [6]. The theorem establishes that any smooth function is approximated by a multivariate polynomial of the series,

$$f(x_0, x_1, \ldots, x_n) \sim \sum_{k=0}^{\infty} \frac{1}{k!} \left(\sum_{i=1}^{n} (x_i - h_i) \frac{\partial}{\partial x_i} \right)^k f(x_0, x_1, \ldots, x_n) \tag{43}$$

for n variables given in a vector $\mathbf{x} \in \mathbb{R}^n$, such that $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$. And the k degree Taylor expansion is developed,

$$f(\mathbf{x}) \approx f(\mathbf{x}) + \left((x_1 - h_1)\frac{\partial}{\partial x_1} + (x_2 - h_2)\frac{\partial}{\partial x_2} + \cdots + (x_n - h_n)\frac{\partial}{\partial x_n} \right)^k f(\mathbf{x}) \tag{44}$$

For instance, the expansion of Taylor series for two variables x_1 and x_2, and for $h_{1,2} = 0$, with $k = 2$ degrees (second order derivative) is developed next.

$$f(x_1, x_2) \approx f(x_1, x_2) + \left(x_1\frac{\partial}{\partial x_1} + x_2\frac{\partial}{\partial x_2} \right)^2 f(x_1, x_2) \tag{45}$$

by algebraically expanding the two degree binomial,

$$f(x_1, x_2) \approx f(x_1, x_2) + \left(x_1^2\frac{\partial^2}{\partial x_1^2} + 2x_1x_2\frac{\partial}{\partial x_1 \partial x_2} + x_2^2\frac{\partial^2}{\partial x_2^2} \right) f(x_1, x_2) \tag{46}$$

thus, the second order derivative approximation is

$$f(x_1, x_2) \approx f(x_1, x_2) + x_1^2\frac{\partial^2 f(x_1, x_2)}{\partial x_1^2} + 2x_1x_2\frac{\partial f(x_1, x_2)}{\partial x_1 \partial x_2} + x_2^2\frac{\partial^2 f(x_1, x_2)}{\partial x_2^2} \tag{47}$$

7 Multivariate Non-linear Functions

A multivariate series develops polynomials that may be used to discriminate nonlinear separable multi-class in hyper-dimension spaces [4,13] by the next general expression

$$f(x) = a_0 + \sum_{i_1}^{n} w_{i_1} x_{i_1} + \sum_{i_1}^{n}\sum_{i_2}^{n} w_{i_1 i_2} x_{i1} x_{i2} + \sum_{i_1}^{n}\sum_{i_2}^{n}\sum_{i_3}^{n} w_{i_1 i_2 i_3} x_{i_1} x_{i_2} x_{i_3} + \ldots \tag{48}$$

where a_0 is the independent coefficient; n determines the space dimension. The polynomial degree is established by the number of summed terms. $\sum_i w_i$ determines the coefficients, and $\sum_i x_i$ are the independent variables at i^{th} dimension. This approach makes particularly complex to determine all coefficients value, because multi-dimension numerical recursive methods have to be used, assuming that the independent variables are initially known values.

For instance, a linear polynomial in three dimension

$$f(x) = w_0 + \sum_{i=1}^{3} w_{i1} x_{i1} = w_0 + w_1 x_1 + w_2 x_2 \tag{49}$$

Another example, for a quadratic polynomial in 2D,

$$f(x) = w_0 + \sum_{i=1}^{2} w_{i1} x_{i1} + \sum_{i_1=1}^{2}\sum_{i_2=1}^{2} w_{i_1 i_2} x_{i_1} x_{i_2} \tag{50}$$

hence by developing the series and algebraically arranging,

$$f(x) = w_0 + w_1 x_1 + w_2 x_2 + w_{11} x_1^2 + w_2 x_2^2 + x_1 x_2 (w_{12} + w_{21}) \tag{51}$$

8 Multi-layer Artificial Neural Networks

The ANN are discriminant functions native multidimensional and non linear that are trained by linear models [15–18]. Once the synapses parameter converged during training, the feed forward model to be used is particularly a fast and simple expression [11]. Thus, let us define the weighting sum of inputs and weights in the first layer as $net_j = \mathbf{w}^\top \cdot \mathbf{f}_j$. Where the first neurons layer outputs toward the hidden layer is $y_j = g(net_j)$. For the hidden layer $net_k = \mathbf{w}_k^\top \cdot \mathbf{y}$, and the hidden layer outputs [13,14],

$$\mathbf{z} = \mathbf{f}(\mathbf{w}_k^\top \cdot \mathbf{f}(\mathbf{w}_j^\top \cdot \mathbf{x})) \tag{52}$$

The general squared absolute differences error ε of the output neurons layer vector $\mathbf{z}_t$ and their output model vector $\mathbf{z}^o$ is stated by

$$\varepsilon(\mathbf{w}) = \frac{1}{2} \|\mathbf{z}^o - \mathbf{z}_t\| \tag{53}$$

The general back-propagation (BP) learning rule has fundamentals on the gradient descendent. The weights vector $\mathbf{w}$ recursively changes its direction until minimise the error. Thus,

$$\Delta \mathbf{w} = -\eta \frac{\partial \varepsilon}{\partial \mathbf{w}} \tag{54}$$

where η is the learning factor that by numerical approximations represents the magnitude of the change of direction of $\mathbf{w}$. Such that $\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta \mathbf{w}_k$. Thus, by using the chain rule for the hidden neurons layer,

$$\frac{\partial \varepsilon}{\partial w_{kj}} = \frac{\partial \varepsilon}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = -D_k \frac{\partial net_k}{\partial w_{kj}} \tag{55}$$

where the sensitivity of the hidden neurons k is $D_k = -\frac{\partial \varepsilon}{\partial net_k}$ and, the output of a single neuron, D_k is

$$D_k = -\frac{\partial \varepsilon}{\partial net_k} = -\frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} = (z_k^o - z_k) f'(net_k) \tag{56}$$

and given the activation function $f(\cdot)$, and its derivative $f'(\cdot)$ as a sigmoid for the hidden layer,

$$f(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + \mathrm{e}^{-\mathbf{wx}}}; \quad \text{and} \quad f'(\mathbf{w} \cdot \mathbf{x}) = \frac{\mathrm{e}^{\mathbf{w} \cdot \mathbf{x}}}{(1 + \mathrm{e}^{-\mathbf{w} \cdot \mathbf{x}})^2} \tag{57}$$

defining $net_k = \mathbf{w}^\top \cdot \mathbf{y}$, thus $\frac{\partial net_k}{\partial w_{kj}} = y_j$. Therefore, the learning rule for the hidden layer to the outputs is $\Delta w_{kj} = \eta D_k y_i = \eta (z_k^o - z_k) f'(net_k) y_j$, and

$$\frac{\partial \varepsilon}{\partial w_{ji}} = \frac{\partial \varepsilon}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}} \tag{58}$$

the error rate of change

$$\frac{\partial \varepsilon}{\partial y_j} = \frac{\partial}{\partial y_j}\left(\frac{1}{2}\sum_{k=1^c}(z_k^o - z_k)^2\right) \tag{59}$$

thus, algebraically arranging previous expression,

$$\frac{\partial \varepsilon}{\partial y_j} = -\sum_{k=1}^{c}(z_k^o - z_k) f'(net_k) w_{ij} \tag{60}$$

the hidden layer sensitivity is defined as

$$D_j = f(net_j)\sum_{k=1}^{c} w_{ij} D_k \tag{61}$$

Hence, the learning rule for the weights $\mathbf{w}$ of the input layer toward the hidden layer,

$$\Delta w_{ji} = \eta \mathbf{w}^\top \cdot \mathbf{x} f'(net_j)\mathbf{x} \tag{62}$$

9 Conclusion and Results

This manuscript presented an analysis of numerical fitting methods purposed to be used for solving highly non linear discriminant functions in classification problems of machine learning. We found that numerous problems present non bijective discriminant paths. A partial solution that alleviates the problem is to firstly transform the path of discriminant points by geometric rotations for subsequent model fitting. Some fitting models are depicted in Fig. 1 applied to

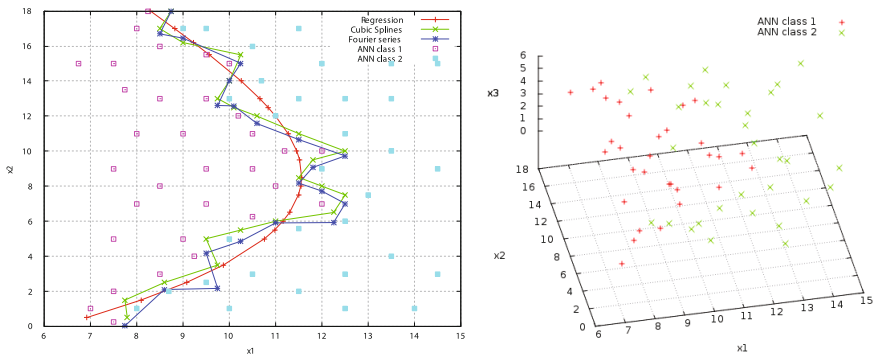

Fig. 1. Left: Different nonlinear fitting methods. Right: Same data set in 3-dimension labelled by a multi-layer Perceptron ANN. Classification performance: regression 99.8%, interpolation 99.82%, cubic splines 100%, Fourier series 99.95%, and multi-layer ANN 100%.

the same data set. Where only interpolation Splines, Fourier series, and multidimensional ANNs are methods capable to over-fit a set of discriminant threshold points that are sequentially nonlinear discontinuous. The rest of the methods yielded small degrees of classification errors.

Regressions and interpolations are numeric fitting methods that are more suitable for off-line systems with prior data. The former performs numerical computation for matrix inversion particularly for massive data. The latter, requires reduced numbers of data representing their trend, requiring symbolic calculation, hence computation becomes more complex than polynomial regressions. Although, regressions and interpolations fit to non linearity, they may fail tracking discontinuities. Thus, a more sophisticated and accurate fitting approach is by using splines. Fitting curves using the Fourier series may work better than polynomials, and even could fit major non linear discontinuities. But yielding complex and long trigonometric functions, limiting mathematical solutions. To diminish this problem, trigonometric/exponential models may further be transformed and approximated to polynomials by the Taylor series. Solving exact parameters for highly multivariate functions becomes too complex. An alternative solution is the multi-layer ANN, where highly dimensional spaces do not represent any problem. ANN overcome numerous discontinuities by increasing the number of hidden layers. Nevertheless, ANN are tided to off-line training for deep learning.

References

1. Jabbar, H.K., Khan, R.Z.: Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). Comp. Sci., Comm. Instr. Dev. 163–172 (2015)
2. Tetko, I.V., Livingstone, D.J., Luik, A.I.: Neural network studies. 1. Comparison of overfitting and overtraining. J. Chem. Inf. Comput. Sci. **35**, 826–833 (1995)
3. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2001)
4. Riley, K.F., Hobson, M.P., Bence, S.J.: Mathematical Methods for Physics and Engineering, 3rd edn. Cambridge University Press, Cambridge (2006)
5. Cheney, W., Kincais, D.: Numerical Mathematics and Computing, 6th edn. Cengage, Boston (2011)
6. Kreyszig, E.: Advanced Engineering Mathematics, 3rd edn. Limusa Wiley, Mexico City (2009)
7. Van, D.A., Rubin, W.M., Rubin, V., Verbeek, H.M.W., Van Dongen, B.F., Kindler, E., Gunther, C.W.: Process mining: a two-step approach to balance between underfitting and overfitting. Softw. Syst. Model. **9**(1), 87–111 (2010)
8. Loughrey, J., Cunningham, P.: Using early-stopping to avoid overfitting in wrapper- based feature selection employing stochastic search (2005)
9. Schaffer, C.: Overfitting avoidance as bias. Mach. Learn. **10**(2), 153–178 (1993)
10. Ruppert, D., Carroll, R.J.: Spatially-adaptive penalties for spline fitting. J. Statist. **42**, 205–224 (2000)
11. Lawrence, S., Giles, C.L., Tsoi, A.C.: Lessons in neural network training: overfitting may be harder than expected. In: Proceedings of the 14th National Conference on AI, USA, pp. 540–545 (1997)

12. Raskutti, G., Wainwright, M.J., Yu, B.: Early stopping and non-parametric regression: an optimal data-dependent stopping rule. J. Mach. Learn. Res. **15**(1), 335–366 (2014)
13. Bishop, C.M.: Neural Networks for Patter Recognition. Oxford University Press, Oxford (2005)
14. Freeman, J.A., Skapura, D.M.: Neural Networks, Algorithms, Applications and Programming Techniques. Computation and Neural Systems Series. Addison-Wesley, Reading (2002)
15. Wen, U.P., Lan, K.M., Shih, H.S.: A review of Hopfield neural networks for solving mathematical programming problems. Eur. J. Oper. Res. **198**(3), 675–687 (2009)
16. Caruana, R., Lawrence, S., Giles, L.: Overfitting in neural nets: backpropagation, conjugate gradient, and early stopping. In: Advances in Neural Information Processing Systems, pp. 402–408 (2011)
17. Kazushi, M.: Avoiding overfitting in multilayer perceptrons with feeling-of-knowing using self-organizing maps. Biosystems **80**(1), 37–40 (2005)
18. Gaurang, P., Amit, G., Parth, S., Devyani, P.: Determination Of over-learning and over-fitting problem in back propagation neural network. Int. J. Soft Comput. **2**(2), 40–51 (2011)