Differential polynomial neural network applications

Ladislav ZJAVKA

This project is funded by Structural Funds of the European Union (ESF) and state budget of the Czech Republic
Biologically inspired neuro-computing

Weighted combined signals are summed in the cell body and passed through a dynamic periodic activation function - an excited neural cell generates series of time-delayed output pulses in response to its input signals.

Sum of weighted input signals gets into an activation function (sigmoid). No input combinations are considered.

Artificial neuron (non-linear perceptron)
GMDH polynomial neural network

**Group Method of Data Handling** - Alexey Ivakhnenko in 1969 (Ukraine).

\[ y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \]

General connection between inputs \( \mathbf{x} \) and output \( y \) variables is expressed by the Kolmogorov-Gabor polynomial:

\[ y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{m} a_{ijk} x_i x_j x_k + ... \]

\[ A(a_1, a_2, ..., a_m), ... - \text{vectors of parameters} \]

GMDH decomposes the complexity of a process into many simpler relationships each described by a low order processing polynomial of 2 variables. Each neuron fits its output to the desired value \( y_t = f(\mathbf{x}) \) for each input vector \( \mathbf{x} \) from the training set.
General differential equation composition

\[ a + bu + \sum_{i=1}^{n} c_i \cdot \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \cdot \frac{\partial^2 u}{\partial x_i \partial x_j} + ... = 0 \quad u = \sum_{k=1}^{\infty} u_k \]

\[ u(x_1, x_2, \ldots, x_n) - \text{searched function of all input variables} \]

**Method of integral analogues** replaces operators and symbols of a DE by ratio of corresponding variables, derivatives are replaced by analogous proportion signs.

\[ u_i = \frac{(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + \ldots)^{m/n}}{b_0 + b_1 x_1 + \ldots} = \frac{\partial^m f(x_1, \ldots, x_n)}{\partial x_1 \partial x_2 \ldots \partial x_m} \]

\[ m,n \text{– combination degree of denominator and numerator} \]

Ordinary differential equations can involve derivatives in respect to time \( t \) variable:

\[ a + bf + \sum_{i=1}^{m} c_i \cdot \frac{df(t, x_i)}{dt} + \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \cdot \frac{d^2 f(t, x_i, x_j)}{dt^2} + \ldots = 0 \]

\( f(t, x) - \text{function of time } t \text{ and independent variables } x(x_1, x_2, \ldots, x_m) \)

**2nd order partial differential equation (DE) solution :**

\[ F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial x \partial y}, \frac{\partial^2 u}{\partial y^2}\right) = 0 \]

\[ \left(\sum \frac{\partial u_k}{\partial x_1}, \sum \frac{\partial u_k}{\partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_1^2}, \sum \frac{\partial^2 u_k}{\partial x_1 \partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_2^2}\right) \]
Block of derivative neurons

The network block consists of basic and compound neurons, one for each polynomial fraction, defining a sum partial derivative term of a 2\textsuperscript{nd} order DE solution. Neurons do not affect the block output but can direct participate in the total network output calculation.

Neurons = \textit{linear, combination and square} derivative sum terms of a DE solution.

\[
y_1 = \frac{\partial f(x_1, x_2)}{\partial x_1} = w_1 \left( a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 \right)^{\frac{1}{2}} \cdot 1.5 \cdot (b_0 + b_1 x_1)
\]

\[
y_3 = \frac{\partial^2 f(x_2, x_2)}{\partial x_1 \partial x_2} = w_3 \left( a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 \right) \cdot 2.3 \cdot (b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2)
\]

\[
y_4 = \frac{\partial^2 f(x_1, x_2)}{\partial x_1^2} = w_4 \left( a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 \right) \cdot 2.7 \cdot (b_0 + b_1 x_1 + b_2 x_2^2)
\]

Total network output:

\[
Y = \sum_{i=1}^{k} y_i
\]

\[k = \text{amount of active neurons}\]
Stressor – susceptibility interaction model

Modeling the failure probability of electronic circuits using 2 input variables.
Multi-layered networks form composite polynomial functions.

Composite function derivation rules:

\[
F(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_m) = (\phi_1(x), \phi_2(x), \ldots, \phi_m(x)) \quad k=1, \ldots, n
\]

\[
\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(y_1, y_2, \ldots, y_m)}{\partial y_i} \cdot \frac{\partial \phi_i(X)}{\partial x_k}
\]

3\textsuperscript{rd} layer 1\textsuperscript{st} block neuron outputs in respect to 2\textsuperscript{nd} and 1\textsuperscript{st} layer block variables (compound terms):

\[
x_{31} = a_0 + a_1x_{21} + a_2x_{22} + a_3x_{21}x_{22} + a_4x_{21}^2 + a_5x_{22}^2
\]

\[
y_6 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_6 \frac{x_{31}^{1/2}}{1.6 \cdot x_{22}} \cdot \frac{x_{21}^{1/2}}{1.5 \cdot (b_0 + b_1x_{11})}
\]

\[
y_{16} = \frac{\partial f(x_{21}, x_{22})}{\partial x_1} = w_{16} \frac{x_{31}^{1/2}}{1.6 \cdot x_{22}} \cdot \frac{x_{21}^{1/2}}{1.6 \cdot x_{12}} \cdot \frac{x_{11}^{1/2}}{1.5 \cdot (b_0 + b_1x_{11})}
\]
Solar power plant output models

Training with previous 1 or 2 day 10-minute time-series (i.e. data samples). 3 time-series of the solar illuminance variable form the input vector of a power output estimation at the end-time (3rd) variable.
Relative humidity forecast models

3 weather variables - **wind speed**, **temperature** and **sea level pressure** of a locality form input vector.

Training with previous 1 or 2 days hourly data series (24 or 48 hours i.e. data samples).
Power quality monitoring systems

Power factor is the ratio of active power [W] to the apparent power [VA]:

\[
\text{Power factor} = \cos(\varphi) = \frac{\text{Active Power (P)}}{\text{Apparent Power (S)}} = \frac{\text{resistance (R)}}{\text{impedance (Z)}}
\]

2 variables - voltage and current of 3 time-series form 6 variables of input vectors totally. A combination of partial and ordinary differential equations is defined by multi-parametric function relation with time-series.

Mean square error = 0.000124

Power demand (utilization) modelling.
Parallel process of selection and adjustment

1\textsuperscript{st} phase – neuron (DE term) selection

2\textsuperscript{nd} phase parameter adjustment.
Differential polynomial neural network

- generates a sum of simple or compound fractional polynomial derivative terms (differential equation composition)
- is trained only with a small set of input-output data samples (GMDH), larger data sets are problematic
- model complexity increases proportionally along with raising amount of input variables (GMDH)
- combines the polynomial neural network functionalities with some math techniques of differential equation solutions
- searching space contains a great amount of local error solutions (term selection and parameter adjustment is complicated)
- relative data processing allows to describe a wide range of applied test interval input values
- evokes combinatorial explosion (GMDH)
This project is funded by Structural Funds of the European Union (ESF) and state budget of the Czech Republic