Electric Power Systems Research 137 (2016) 113–123

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Short-term power load forecasting with ordinary differential equation substitutions of polynomial networks

Ladislav Zjavka[b,*], Václav Snášel[a,b]

[a] Faculty of Electrical Engineering and Computer Science, Department of Computer Science, VŠB-Technical University of Ostrava, Ostrava, Czech Republic
[b] IT4innovations National Supercomputing Center, VŠB-Technical University of Ostrava, Ostrava, Czech Republic

ARTICLE INFO

Article history:
Received 17 October 2015
Received in revised form 28 February 2016
Accepted 3 April 2016

Keywords:
Power demand prediction
Week and day load cycle
Differential polynomial network
Substituting relative derivative term
Ordinary differential equation solution

ABSTRACT

The purpose of the short-term electricity demand forecasting is to forecast in advance the system load, represented by the sum of all consumers load at the same time. Power load forecasting is important for an economically efficient operation and effective control of power systems and enables to plan the load of generating units. A precise load forecasting is required to avoid high generation costs and the spinning reserve capacity. Under-prediction of the demands leads to an insufficient reserve capacity preparation and can threaten the system stability, on the other hand, over-prediction leads to an unnecessarily large reserve that leads to high cost preparations. Differential polynomial neural network is a new neural network type, which decomposes and solves the selective general partial differential equation, which can model a searched function on the bases of observed data samples. It produces an output sum combination of convergent series of selected relative polynomial derivative terms, which can substitute for an ordinary differential equation solution to describe and forecast real data time-series. Partial derivative terms of several time-point variables substitute for the time derivatives of the converted general ordinary differential equation. The operating principles of the proposed method differ significantly from other conventional neural network techniques.

1. Introduction

Accurate forecasting of the electricity demand is essential in the operation and utilization of electric power grid systems. The potential benefits of an energy demand prediction are obvious useful in the automatic power dispatch, load scheduling and energy control. Over-estimation of future load pattern can cause starting of extra generating units that will lead to a useless increase in the reserve and operating costs. Under-estimation of future load forecasts results in failures to provide the required operating reserve and stability of the system, which may lead to a breakdown in the power network. Load patterns are affected by several factors, such as weather, time, economy, electricity prices, population characteristics, social activities, geographical and environmental conditions, types of consumers. Load forecasts are prepared for different time frames and levels of detail. Forecasting methods in general might be classified into 2 basic types:

- time-series models, in which the load is a function of its past observed values, e.g. autoregressive and moving average models [1]
- multi-variate causal models, which load output is a function of some exogenous factors, especially weather and social variables. [2]

The time-series approach may perform worst on data with inherent special events (holidays, weekends) [3] however both model types can be combined [4]. Regression analysis is considered as a conventional way in power demand predictions, statistical methods are used to estimate the parameters of a function, which predicts the values of a response variable. Statistical forecasting models are represented by deterministic mathematical equations [5], e.g. exponential smoothing with interval time-series [6] may apply a weighted bootstrap predictor, which weights increase according to the similarity between past and actual situations in simultaneous predictions at multiple horizons [7] or a data preprocessing based on the pattern similarity of daily load time-series cycles [1]. Linear regression models can decompose the load into basic and weather dependent components with defined physical interpretations of an exactly described behaviour [8]. Real power time-series are difficult to describe using conventional

* Corresponding author. Tel.: +420 597329790; fax: +420 597329797.
E-mail addresses: lzjavka@gmail.com (L. Zjavka), vaclav.snasel@vsb.cz (V. Snášel).

http://dx.doi.org/10.1016/j.epsr.2016.04.003
0378-7796/© 2016 Elsevier B.V. All rights reserved.

deterministic methods due to uncertain relationships between the load patterns and factors that influence it, e.g. fluctuating weather conditions as well as other extraneous casual factors. Artificial intelligence techniques of the day-ahead load forecasting can involve chaotic dynamic non-linear models with an evolutionary hybrid computation [9] or fuzzy two-valued (IF-THEN) logic for reasoning under non-linear uncertain conditions [10]. Despite the flexibility, accuracy and simplicity to use, the artificial intelligence methods have lots of inconveniences, e.g. difficulties in the parameters and input variables selection, over-fitting, etc. [5].

Artificial neural networks (ANN) are able to model the non-linear nature of dynamic processes, using its approximation capability of any continuous nonlinear functions that offers an effective alternative to more traditional statistical regression techniques. The neuro-computing based short-term load forecasting can apply a time-lagged recurrent neural network trained with seasonal data variances [2], bagged neural networks using bootstrap learning samples obtained from the original training set to average results and reduce the over-fitting [3], random vector functional link networks with direct input–output connections that emulate a time delayed finite impulse response filter [11]. A hybrid forecasting can combine a data preprocessing using wavelet transformations with the time-series and regression analysis for the input variables selection and following step-ahead predictions by means of the Bayesian neural network [12] or the differential empirical mode decomposition and auto-regression with support vector regression models [13]. The data and regression analysis of weather and environmental parameter effects on power distribution interruptions in a region can provide inputs for the ANN model training and forecasting these events [14]. The load models could be updated to take into account a dynamic nature of the power series and formed with respect to several time-point data relations of few successive days foregoing the forecast and trained with several previous weeks corresponding day periods with the same denominations, as the daily power cycle course of each following week is of a similar nature.

$$Y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk} x_i x_j x_k + \cdots \tag{1}$$

n – number of input variables $X(x_1, x_2, \ldots, x_n)$; $A(a_1, a_2, \ldots, a_n), \ldots$ – vectors (matrixes) of parameters.

Differential polynomial neural network (D-PNN) is a new neural network type, which results from the group method of data handling (GMDH) designed by a Ukrainian scientist Aleksey Ivakhnenko [15]. The GMDH constructs in successive steps a polynomial neural network (PNN), adding one last layer a time, selecting its best-fit neurons and calculating the polynomial parameters while an improvement is possible [16]. The GMDH decomposes the Kolmogorov–Gabor polynomial (1), a discrete analogue of Volterra functional series, which can express a general connection between input and output variables of a modelled system. This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or a system of Gaussian normal equations [4]. The GMDH decomposes the complexity of a process or system into many simpler relationships each described by a low order polynomial (2) for every pair of input variables. A typical PNN maps a vector input $\boldsymbol{x}$ to a scalar output y, which is an estimate of the true function $f(\boldsymbol{x})$ [17].

$$p = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \tag{2}$$

x_i, x_j – input variables of polynomial neurons.

D-PNN combines the PNN functionality with some mathematical principles of differential equation (DE) substitutions. D-PNN decomposes a general partial DE description of a searched function model using complete multi-layer PNN structure, analogous to the GMDH does the general connection polynomial (1). A combination of producing sum series of selected relative simple and composite polynomial derivative terms substitute for the general partial DE, which derivatives give an account of data observation relations. In contrast with the ANN functionality, each neuron (i.e. substituting fraction derivative term, formed in any network layer) can be directly included in the total network output, which is calculated as the sum of all active neuron output values [18]. A theoretical background of the general DE decomposition and substitution using the extended PNN backward structures is described in Section 3, Section 4 presents applied algorithms for the D-PNN parameter selection and adjustment. Section 5 compares D-PNN models with several standard soft-computing methods in daily power load forecasting using 2 different historical data sets and the same training and prediction frame technique (of preparing input data and target output series) as described in Section 2, Section 6 resumes the applied strategy, finding and experimental results.

2. Short-term energy load forecasting

Short-term electric energy estimations of a future demand are needful for the planning of generating electricity of regional grid systems and operating power systems. An overall generation plan requires a forecast of total generation requirements and also peak demands. The inaccuracy in a forecast means that load matching is not optimized and consequently the generation and transmission systems are not being operated in an efficient manner. In order to guarantee a regular supply, it is necessary to keep a reserve. Depending on how fast it is available, it is called spinning reserve or cold reserve. Over-estimating the future load results in an unused spinning reserve, being "burnt" for nothing. An unexpected supply to the international energy network is usually not welcome. Under-estimating the future load is equally detrimental, because of high starting costs of cold reserves, buying at the last minute from other suppliers is obviously too expensive. Cooperation on the electricity grid requires from all providers to foresee the demands within a sufficient accuracy [19]. The nature of parameters that affect this problem includes many uncertainties. The accuracy of a dispatching system is influenced by various conditional input parameters (weather, time, historical data and random disturbances), which a prediction model can involve. The load at a given hour is dependent not only on the load at the previous hour, but also on the load at the same hour on the previous day, and on the load at the same hour on the day with the same denomination in the previous week. There are also many important exogenous variables that should be considered, especially weather-related variables [20].

The proposed method keynote of the power load forecasting using historical time-series, is to train a neural network with actual daily cycle similarity relations of previous weeks, concerning several consequent days with the same denomination, foregoing the 24-h prediction. Power values of several sequential days in previous weeks at the same time points form the input vector, while the following day corresponding 24-h shifted series define desired network responses in the training data set. The trained network can forecast the following day power progress, using input vector variables of 24-h shifted time-series of the same time stamps of the current week last days. In addition to the daily time point variables, the input vector can comprise specific time-series of the last day in the training and testing scheme (Fig. 1). The model can apply weather or other historical data, to eliminate a casual prediction error increase. The power demand day cycle progress is more variable in winter than summer months, which is probably influenced largely by using heating systems and temperature conditions [A].

Generally the load pattern on normal weekdays remains almost constant with small random variations from varying industrial

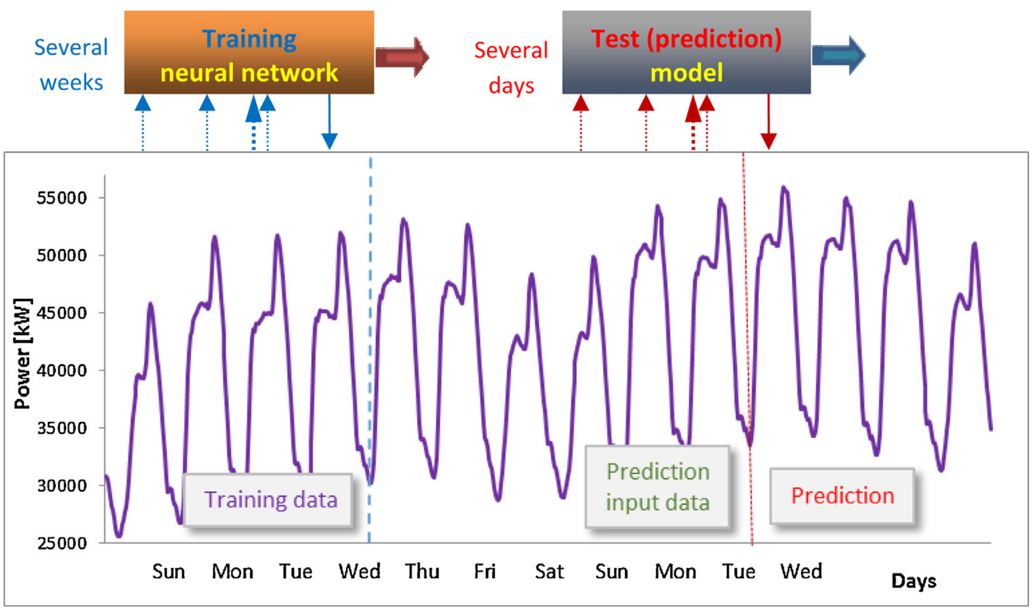

Fig. 1. Typical week cycles of the power demand (January) [A].

activities and weather conditions, so the load values for normal days are functions of the short-term historical data and weather parameters. The load on Mondays and Fridays is different from that on other weekdays due to pick-up loads on Monday mornings, and evening loads on Fridays similar to the weekend. The load pattern on Saturdays is different from rest of the weekdays as the peak load takes a dip. The shape of the load curve on Sundays is similar to that on holidays, with the peak load considerably decreasing. To allow for all these week cycle regularities it is necessary to consider not only time-series of past days but also to heed to the previous week(s) day power patterns of the same denominations.

3. General sum differential equation composition

D-PNN forms (decomposes) and solves the general partial DE (3), which can describe a system model according to data samples, by means of a sum combination of substitution derivative terms. The general DE involves also the simple form of a searched function u, which is calculated as a sum of the rest of its terms (3), i.e. its partial function derivatives. Hereby the unknown function u (3) may be expressed in the form of sum series (4), consisting of convergent series arising from partial derivative terms in respect of derivative variables (5).

$$a + bu + \sum_{i=1}^{n} c_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \cdots = 0 \tag{3}$$

$$u = \sum_{k=1}^{\infty} u_k \tag{4}$$

a, b, $\boldsymbol{c}(c_1, c_2, \ldots, c_n)$, $\boldsymbol{d}(d_{11}, dc_{12},\ldots)$, ... – parameters; $u(\boldsymbol{x})$ – searched function of n-variables; $\boldsymbol{x}(x_1, x_2, \ldots, x_n)$ – vector of n-input variables; u_k – separable partial functions of u.

Substitution partial DE terms (3) are formed according to the adapted method of integral analogues, which is a part of the similarity dimensional analysis. It replaces mathematical operators and symbols in a DE by the ratio of the corresponding variables. Derivatives are replaced by their integral analogues, i.e. derivative operators are removed and replaced by analogue or proportional signs in equations to form dimensionless characteristic groups of variables for a given system, which may be also described by data observations [21].

$$\left(\sum \frac{\partial u_k}{\partial x_1}, \sum \frac{\partial u_k}{\partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_1^2}, \sum \frac{\partial^2 u_k}{\partial x_1 \partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_2^2}\right) \tag{5}$$

The similarity theory is based on the hypothesis functional relationships exist among the non-dimensional parameters, which can describe a physical system. The Buckingham π-theorem provides a simple method for computing sets of dimensionless π units from given physical $\boldsymbol{q}$ variables, even if the (differential) equation form is still unknown. It removes extraneous information from a problem by forming dimensionless groups of variables and is the fundamental of dimensional analysis. The dimensions of a dimensional quantity q_i can be written in terms of certain fundamental dimensions $D_1, D_2, \ldots, D_n$ using a dimension monomial with real exponents r_i (6).

$$[q_i] = D_1^{r1},\ D_2^{r2}, \ldots, D_n^{rn} \quad \text{dimensionless}: \ [q_i] = D_1^0, D_2^0, \ldots, D_n^0 = 1 \tag{6}$$

The Buckingham π-theorem states if the ϕ ($\boldsymbol{q}$) is the only relationship among the q_i's and if it holds for any arbitrary choice of the units in which $q_1, q_2, \ldots, q_n$ are measured (7), then can be written in the form using $\pi_1, \pi_2, \ldots, \pi_m$ units as independent dimensionless products of the q_i's (8). If k is the minimal number of principal quantities necessary to express the dimensions of the $\boldsymbol{q}$'s, then $m = n - k$ [22].

$$\phi(q_1, q_2, \ldots, q_n) = 0 \quad \phi(\pi_1, \pi_2, \ldots, \pi_m) = 0 \tag{7}$$

Some combinations of $q_1, q_2, \ldots, q_n$ are dimensionless and can form π-terms using the dimensional matrix (6). The relationship between the original $\boldsymbol{q}$ variables can be expressed as a relationship between the various $\boldsymbol{\pi}$ groups e.g. (8).

$$\pi_i = q_1^{a1}, q_2^{a1}, \ldots, q_n^{an} \quad \pi_1 = f(\pi_2, \pi_3, \ldots, \pi_m) \tag{8}$$

where exponents a_i are rational numbers

The relationship among the non-dimensional π-variables produces new unit factors without any loss of the generality (8). The dimensional analysis is a technique for restructuring the original dimensional variables of a problem into a set of dimensionless products using the constraints imposed upon them by their dimensions. The linear model may also take a simple polynomial form. The π-functions must be invariant to a change of model units, so the arbitrary change for a dimensional variable x_i can be written as (9).

$$x_i' = \alpha_1^{D_{1i}}, \alpha_2^{D_{2i}}, \ldots, \alpha_m^{D_{mi}} x_i \tag{9}$$

α_j – unit scale change; D_{mi} – dimensionality of the ith variable; m – number of fundamental units.

If a physical model with n-dimensional variables is assumed, the invariance for all possible α-changes of m-units may be written in the form (10) and (11) for each α_j.

$$F(x_1, x_2, \ldots, x_n) = F(\alpha_1^{D_{11}} \alpha_2^{D_{21}} \ldots \alpha_m^{D_{m1}} x_1, \alpha_1^{D_{12}} \alpha_2^{D_{22}} \ldots \alpha_m^{D_{m2}} x_2, \ldots, \alpha_1^{D_{1i}} \alpha_2^{D_{2i}} \ldots \alpha_m^{D_{mn}} x_n) \tag{10}$$

$$\frac{\partial F}{\partial \alpha_j} = \frac{\partial F}{\partial x_1} \cdot \frac{\partial x_1}{\partial \alpha_j} + \frac{\partial F}{\partial x_2} \cdot \frac{\partial x_2}{\partial \alpha_j} + \cdots + \frac{\partial F}{\partial x_n} \cdot \frac{\partial x_n}{\partial \alpha_j} = 0 \tag{11}$$

For example the head loss (π_1) in a horizontal roughened pipe in turbulent flow is related to the Reynold's number (π_2-term) and 2 additional dimensionless units (12), assuming the pressure drop depends linearly on the pipe length (13).

$$Re = \pi_2 = \frac{\rho D v}{\mu} \quad \pi_1 = \frac{\Delta p}{\rho v^2} \quad \pi_3 = \frac{l}{D} \quad \pi_4 = \eta \tag{12}$$

$$\pi_1 = f(\pi_2, \pi_3, \pi_4) \quad \frac{\Delta p}{\rho g} = \left(\frac{v^2}{2g}\right)\left(\frac{l}{D}\right) f_2(Re, \eta) \tag{13}$$

l, pipe length; D, diameter; v, flow velocity; Δp, pressure drop; μ, viscosity; ρ, density; η, surface roughness.

If each variable x_i of a function u is independent of other variables, then the function u is separable and might be approximated by its partial sum functions u_k (4), i.e. its derivative terms, formed in respect of 1 or more variables. The function partial derivatives then can be expressed in the form of a product of 2 functions, where g means one (or several) – variable function of x_i only and h any complete function of all input vector variables $\mathbf{x}(x_1, x_2, \ldots, x_n)$ (14).

$$\frac{\partial u(x_1, x_2, \ldots, x_n)}{\partial x_i} = g(x_i) \cdot h(x_1, x_2, \ldots, x_n) \tag{14}$$

Polynomial fractions (12), which describe partial derivative relations of n-input variables, can substitute for the sum terms of the general partial DE solution (3). The complete input polynomials (2) replace the partial functions u_k of derivative term series (5), while the reduced polynomials of denominators represent the alterative derivative parts (15). The root function of the numerator reduces a combination degree of the input polynomial, in order to get the

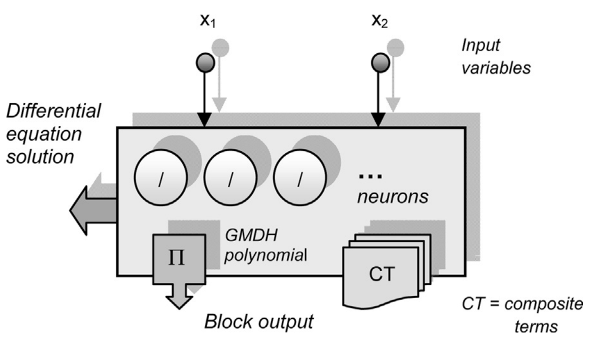

Fig. 2. D-PNN block involves simple and composite neurons (substitution DE terms) [23].

dimensionless units.

$$v_i = \frac{(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + \cdots)^{m/n}}{b_0 + b_1 x_1 + \cdots} = \frac{\partial^n f(x_1, \ldots, x_n)}{\partial x_1, \partial x_2, \ldots, \partial x_m} \tag{15}$$

n – combination degree of a complete polynomial of n-input variables; m – combination degree of a denominator derivative polynomial.

An ordinary differential Eq. (16) with only time derivatives can describe 1-variable function $s(x)$ time-series using the partial derivative term fractions (15) with specific time-step variables, analogous to the general partial DE (3) substitution.

$$a + bs + \sum_{i=1}^{n} c_i \frac{ds(x_{t_i})}{dt} + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \frac{ds(x_{t_i}, x_{t_j})}{dt} + \cdots + \sum_{i=1}^{n} cc_i \frac{d^2 s(x_{t_i})}{dt^2} + \cdots = 0 \tag{16}$$

$s(\mathbf{x_t})$ – function of independent time observations $\mathbf{x}(x_{t1}, x_{t2}, \ldots, x_{tn})$

Blocks (nodes) of the D-PNN (Fig. 2) form neurons with the same inputs, one for each polynomial derivative combination, so each neuron fraction is considered a substitution DE term (15). Each block contains a single output GMDH polynomial (2), without derivative part. Weighted neurons do not affect the block output but can be directly involved in the total network output sum calculation of a DE composition solution. Each block has 1 and neuron 2 vectors of adjustable parameters **a** and **a, b** respectively. All the neuron and block polynomial outputs cannot become negative values.

$$F\left(x_1, x_2, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \frac{\partial^2 u}{\partial x_2^2}\right) = 0 \tag{17}$$

where $F(x_1, x_2, u, p, q, r, s, t)$ is a function of 8 variables.

While using 2 input variables the 2nd order partial DE (17) involves derivative terms, formed with respect to all the GMDH polynomial (2) variables. Each block of the D-PNN forms corresponding 5 simple neurons (DE terms) of 2 single x_1, x_2 (18), 2 squared $x_1{}^2$, $x_2{}^2$ (19) and 1 combination $x_1 x_2$ (20) derivative variables, which can solve and substitute for the 2nd order partial DE (17). This type of the 2nd order partial DE is most often used to model physical or natural systems non-linearities.

$$y_1 = \frac{\partial f(x_1, x_2)}{\partial x_1} = w_1 \frac{(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2)^{1/2}}{b_0 + b_1 x_1} \tag{18}$$

$$y_3 = \frac{\partial^2 f(x_1, x_2)}{\partial x_2^2} = w_3 \frac{a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2}{b_0 + b_1 x_2 + b_2 x_2^2} \tag{19}$$

$$y_5 = \frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2} = w_5 \frac{a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2}{b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2} \tag{20}$$

y_i – substitution derivative term (neuron) output; a_i, b_i – polynomial parameters; w_i – term (neuron) weight.

4. Backward selective differential polynomial network

Multi-layer networks form composite functions (Fig. 3); the blocks preceding layers create internal functions (21), which substitute for input variables of neuron and block polynomials in

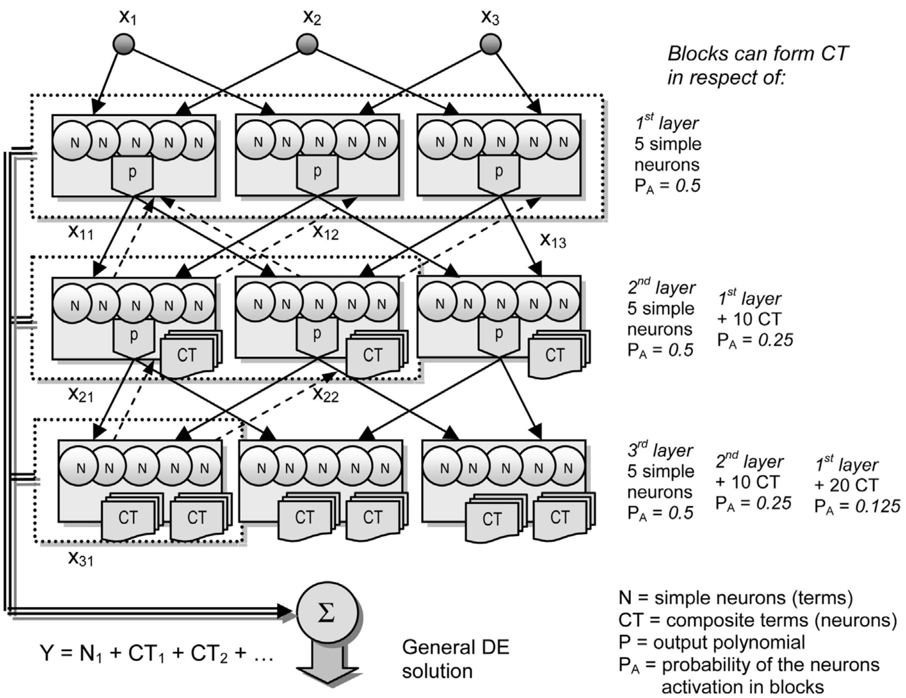
x1
x2
x3
Blocks can form CT in respect of:
1st layer 5 simple neurons PA = 0.5
N
p
x11
x12
x13
2nd layer 5 simple neurons PA = 0.5
1st layer + 10 CT PA = 0.25
CT
x21
x22
3rd layer 5 simple neurons PA = 0.5
2nd layer + 10 CT PA = 0.25
1st layer + 20 CT PA = 0.125
x31
Σ
Y = N1 + CT1 + CT2 + ...
General DE solution
N = simple neurons (terms)
CT = composite terms (neurons)
P = output polynomial
PA = probability of the neurons activation in blocks

Fig. 3. The D-PNN selects from the neurons in all layers to substitute for the general DE [23].

the next hidden layer to produce external functions (22). Substitution composite terms (CT), i.e. composite function derivatives with respect to the variables of previous layers, are calculated according to the partial derivation rules (23) and (24).

$$z_i = \varphi_i(X) = \varphi_i(x_1, x_2, \ldots, x_n) \quad i = 1, \ldots, m \tag{21}$$

$$F(x_1, x_2, \ldots, x_n) = f(z_1, z_2, \ldots, z_m) = f(\varphi_1(X), \varphi_2(X), \ldots, \varphi_m(X)) \tag{22}$$

$$\left.\begin{aligned}
\frac{\partial F}{\partial x_1} &= \frac{\partial f}{\partial z_1}\cdot\frac{\partial \varphi_1}{\partial x_1} + \frac{\partial f}{\partial z_2}\cdot\frac{\partial \varphi_2}{\partial x_1} + \cdots + \frac{\partial f}{\partial z_m}\cdot\frac{\partial \varphi_m}{\partial x_1}\\
\frac{\partial F}{\partial x_2} &= \frac{\partial f}{\partial z_1}\cdot\frac{\partial \varphi_1}{\partial x_2} + \frac{\partial f}{\partial z_2}\cdot\frac{\partial \varphi_2}{\partial x_2} + \cdots + \frac{\partial f}{\partial z_m}\cdot\frac{\partial \varphi_m}{\partial x_2}\\
&\ \ \vdots\\
\frac{\partial F}{\partial x_n} &= \frac{\partial f}{\partial z_1}\cdot\frac{\partial \varphi_1}{\partial z_n} + \frac{\partial f}{\partial z_2}\cdot\frac{\partial \varphi_2}{\partial x_n} + \cdots + \frac{\partial f}{\partial z_m}\cdot\frac{\partial \varphi_m}{\partial x_n}
\end{aligned}\right\} \tag{23}$$

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(z_1, z_2, \ldots, z_m)}{\partial z_i}\cdot\frac{\partial \varphi_i(X)}{\partial x_k} \quad k = 1, \ldots, n \tag{24}$$

$\varphi_i(\boldsymbol{x})$ – internal functions of $f(\boldsymbol{z})$ composite functions.

The blocks of the 2nd and following hidden layers form additionally composite terms (extended neurons), which substitute for composite function derivatives with respect to the (inner functions) output and input variables of back connected previous layers blocks (Fig. 3). The D-PNN with 3 input variables consists of 3-layers, which blocks can form all the possible fraction terms (neurons). For example the 1st block of the last 3rd hidden layer can form 5 simple neurons, i.e. basic substitution terms (18)–(20) of the partial DE (3) using its own 2 input variables only (Fig. 3). Additionally it creates 10 CT (a double of the neurons) with respect to the previous 2nd layer 2 blocks derivative input variables using composite function substitution derivatives products with respect to reverse outputs of the 2 back-connected blocks (25). 20 CT with respect to the 1st layer 3 blocks input variables are formed analogously (26). The total number of the CT in blocks, which involve composite functions, doubles with each previous back-connected layer, so the probability activations P_A of CT, which are formed with respect to the previous layers block input variables, must halve together with the increasing number of hidden layers they backward comprise in the network tree-structure (Fig. 3). As the couples of input variables of the internal functions $\phi_1(x_1,\ x_2)$ and $\phi_2(x_3,\ x_4)$ (15) can differ from each other, the partial derivatives (24) are calculated separately in respect of each block individual variables, thus each sum (23) consists of only 1 derivative term, which represents a single neuron [18].

The back-calculation of the composite function derivatives is well done by a recursive algorithm in the network tree-like structure (Fig. 3). The composite squared and combination DE substitution terms are also formed according to the composite function partial derivation rules, analogous to the product composition of the linear composite terms (25) and (26). The 1st block of the 3rd layer (Fig. 3) forms additional 10 CT with respect to 2 blocks input variables of the 2nd layer, $P_A = 0.25$, e.g. x_{11}:

$$y_2 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_2 \frac{(a_0 + a_1x_{21} + a_2x_{22} + a_3x_{21}x_{22} + a_4x_{21}^2 + a_5x_{22}^2)^{1/2}}{x_{22}} \cdot \frac{x_{21}}{b_0 + b_1x_{11}} \tag{25}$$

Finally it can apply 20 CT with resp. to 3 blocks input variables of the 1st layer, $P_A = 0.125$, e.g. x_1:

$$y_3 = \frac{\partial f(x_{21}, x_{22})}{\partial x_1} = w_3 \frac{(a_0 + a_1x_{21} + a_2x_{22} + a_3x_{21}x_{22} + a_4x_{21}^2 + a_5x_{22}^2)^{1/2}}{x_{22}} \cdot \frac{(x_{21})^{1/2}}{x_{12}} \cdot \frac{x_{11}}{b_0 + b_1x_1} \tag{26}$$

y_i, outputs of the substitution composite terms (CT).

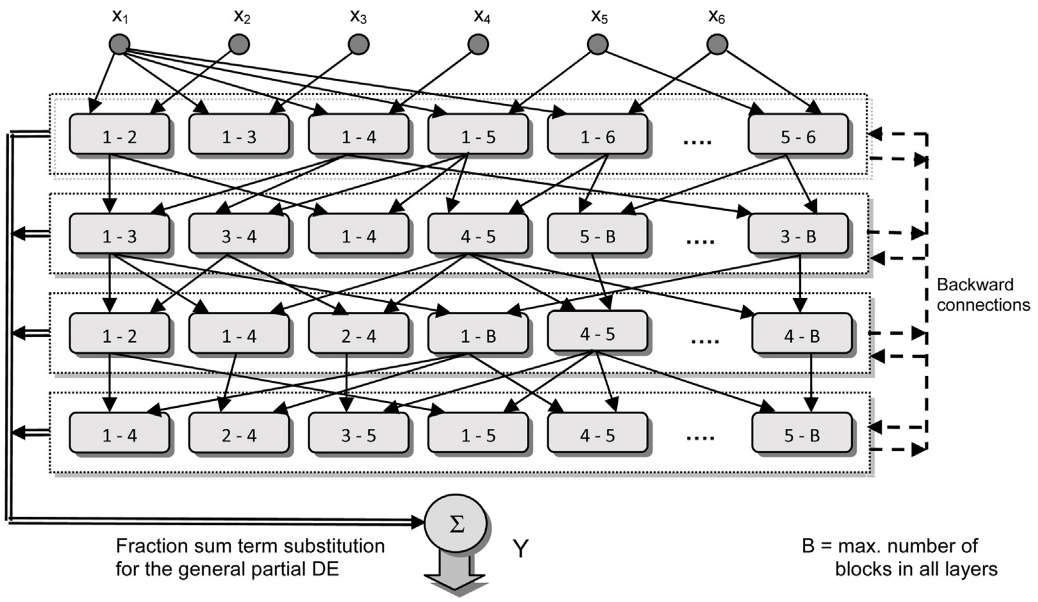

Fig. 4. N-variable D-PNN selects from (reconnects) all the 2-combination blocks in each hidden layer.

The input data are normalized to the range ⟨*0.5*, *1.5*⟩. All the neuron (18) and (25) and block (2) polynomials are divided by the number of the items (polynomial members) they include (28) to produce the outputs around the value *1.0*, which is especially useful for the polynomials calculation in next hidden layers and the parameter adjustment. The complete input polynomials (2) of neuron numerators have a constant length (28), while the number of denominators polynomial items vary according to the types of derivatives they represent (18)–(20). Only some of all the potential substitution terms (neurons) may form a DE solution although they have an adjustable term weight w_i. A proper neuron combination, which can substitute for the general DE (3), is not able to accept a disturbing effect of the rest neurons (which may also compose other partly local error sub-solutions) in the parameters optimization. The neuron selection, which is a critical initial composing phase of the DE solution search, may be performed by means of the standard particle swarm optimization (PSO) or binary PSO, able to solve effectively large binary combinatorial problems [24].

$$Y = \frac{\sum_{i=1}^{k} y_i}{k} \qquad (27)$$

k – number of active neurons (subst. DE terms).

The D-PNN total output Y is the arithmetic mean of the selected active neuron output values (27) so as to prevent a varying number of the active neurons (of a combination) from influencing the total network output sum. The raw total D-PNN output (mostly form the interval ⟨*0.9*, *1.1*⟩) must be scaled (denormalized) to a desired range of output function values. The D-PNN (and also GMDH) approximation ability of periodic functions is possible to improve by means of a sigmoidal transformation (sig) of the squared polynomial items together with their parameters in all the neuron and block polynomials (28) [25].

$$p = \frac{(a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + sig(a_4 x_i^2) + sig(a_5 x_j^2))}{l} \qquad (28)$$

l – number of polynomial items (6 in this case).

If the number of input variables increase, the number of the D-PNN *2*-combination blocks (nodes) grow exponentially in each following hidden layer (Fig. 4). Hence The D-PNN with more than 3-input variables must select the optimal block inputs in each hidden layer (Fig. 4), which reconnection process may employ the simulated annealing (SA) [26] or some proper genetic optimization techniques, analogous to the GMDH algorithm (*combinatorial explosion*) [17]. The D-PNN with an increased number of input variables need not to define the complete data relations (i.e. apply the coequal number of hidden layers), e.g. 4 layers are enough to approximate 5–6 input variable functions (Section 5).

The primary shortest reconnection phase of the SA random block input 2-combinations search, forming the optimal D-PNN skeleton structure, along with the longer-term binary PSO neurons selection is performed simultaneously together with the continuous polynomial parameter adjustment using the gradient steepest descent (GSD) method [27] (Fig. 5). The D-PNN can be trained with only a small set of input–output data samples, in a similar way to the GMDH algorithm. The training and testing error functions are calculated using the root mean squared error (RMSE) (29).

$$E = \sqrt{\frac{\sum_{i=1}^{2} (Y_i^d - Y_i)}{M}} \rightarrow \min \qquad (29)$$

Y_i^d – desired output; Y_i – estimated output for ith training vector; M, number of data samples.

5. Power load daily prediction models

The presented D-PNN was trained with 1-variable historical power load time-series of the 1st set [A] to model several day cycles data function relations. 3 consequent day power values (foregoing the forecasting day denomination) of several previous weeks (3–6) at the same uniform time points form the network input vector, corresponding to the 24-h shifted time-series of desired scalar outputs (Fig. 1). The trained network models can predict the following day 24-h power load series in respect of the current week last 3 day input values (Fig. 1). The optimal number and type of input variables were selected and tested experimentally. The mean absolute

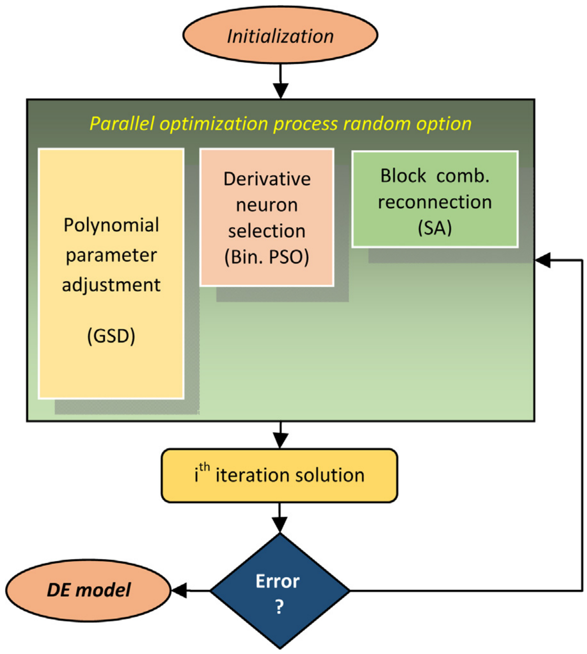
Initialization
Parallel optimization process random option
Polynomial parameter adjustment (GSD)
Derivative neuron selection (Bin. PSO)
Block comb. reconnection (SA)
i^th iteration solution
Error ?
DE model

Fig. 5. 3-parallel gradually finishing processes of the D-PNN selection and parameter adjustments.

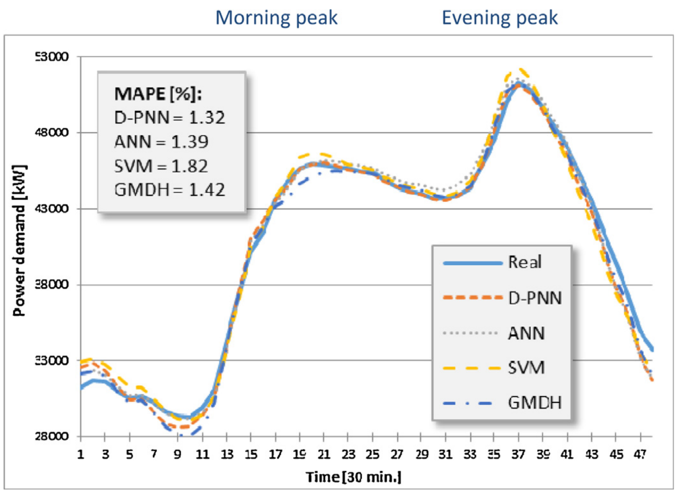

Fig. 6. National Grid – 18.2.2013 (Monday), RMSE: D-PNN = *682.3*, ANN = *711.2*, SVM = *881.2*, GMDH = *664.3*.

percentage error (MAPE) is the measure of accuracy of a method estimating time-series values (30).

$$MAPE = \frac{100\%}{t}\sum_{i=1}^{t}\left|\frac{A_i - F_i}{A_i}\right| \qquad (30)$$

t – number of time points; A_i – actual value; F_i – forecasted value.

The maximal load absolute error (MLAE) is an accuracy of the daily maximal power load prediction (31).

$$MLAE = |A_{Max} - F_{Max}| \qquad (31)$$

A_{Max} – maximal actual value; F_{Max} – maximal forecasted value.

A power demand daily period typically consists of 2 peak-hours, morning (midday) and evening peak consumptions (Fig. 6). Comparisons were done with 1 or 2-layer ANN containing 15–25 neurons in each hidden layer using the sigmoidal activation function and standard back-propagation algorithm with momentum, support vector machine (SVM) for regression using the dot kernel

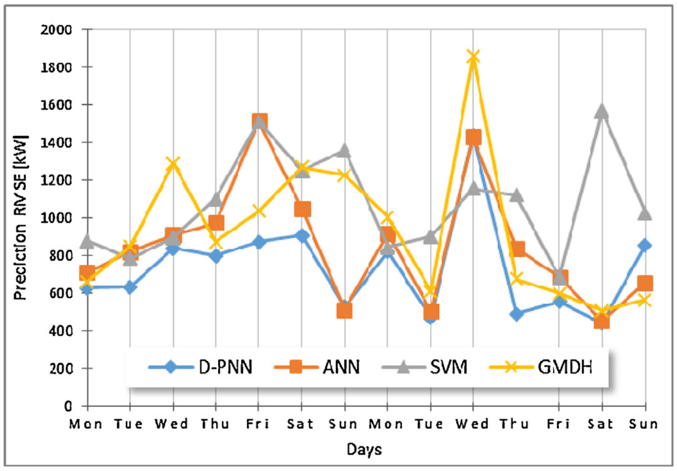

Fig. 7. 2-week daily power load prediction RMSE (18.2.2013 to 3.3.2013), average RMSE: D-PNN = *763.7*, ANN = *857.7*, SVM = *1079.7*, GMDH = *932.7* [A].

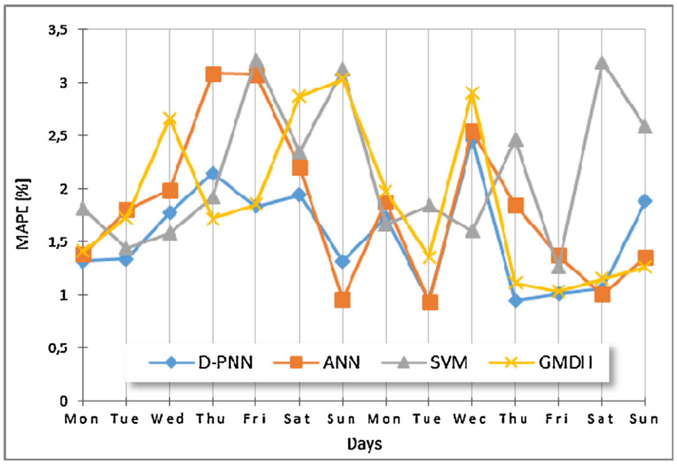

Fig. 8. 2-week daily power load prediction MAPE (18.2.2013 to 3.3.2013), average MAPE: D-PNN = *1.56*, ANN = *1.82*, SVM = *2.15*, GMDH = *1.87* [A].

[28] [D] and GMDH Shell, a professional data science forecasting software. An alternative SVM using the radial kernel, primarily designed for the pattern recognition (but possible to use also for the regression), was additionally tested [29] [E]. The applied SVM ε parameter in the loss function was optimized.

Most of day models get with very good approximations (Fig. 6) however some of them are less accurate, influenced likely by a sudden break in weather conditions or other non-considered factors. Sundays and Saturdays power models result as a rule in higher inaccuracies, which are induced by more variable power demands on weekends and holidays. Several additional time-series input variables (2–3), close to the last 3rd day time-point (3–6 h) can improve the forecasts for the free days (Fig. 1), so the input vectors consist of 5–6 input variables in this case. Historical local temperature (relative humidity or other weather variables) time-series [C] can extend the data set (input vectors), however their application did not benefit the models forecasts in these experiments. The cause rests likely in the applied averaged electric load measurements that cover substantial parts of England and Wales such that the effect of meteorological data is dispersed to a vaster region [A] and cannot influence positively the model predictions. Figs. 7–10 and Table 1 compare the described 4 neuro-computing techniques, which applied the same daily-similarity (time-point data relations) prediction method (Fig. 1) as the D-PNN.

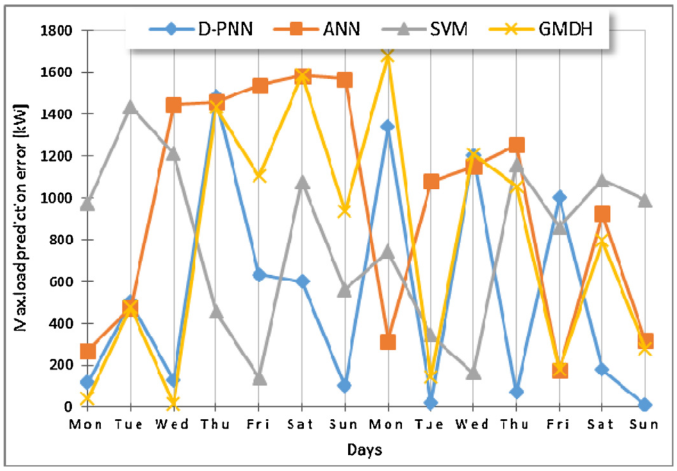

Fig. 9. 2-week daily maximal power load absolute prediction errors (18.2.2013 to 3.3.2013), Average MLAE: D-PNN = *538.1*, ANN = *967.9*, SVM = *799.9*, GMDH = *781.3* [A].

Table 1
Total average power load prediction RMSE, MAPE and MLAE (18.2.2013 to 3.3.2013) [A].

Model/error	RMSE (kW)	MAPE (%)	MLAE (kW)
D-PNN	763.7	1.56	538.1
ANN	857.7	1.82	967.9
SVM	1079.7	2.15	799.9
GMDH	932.7	1.87	781.3

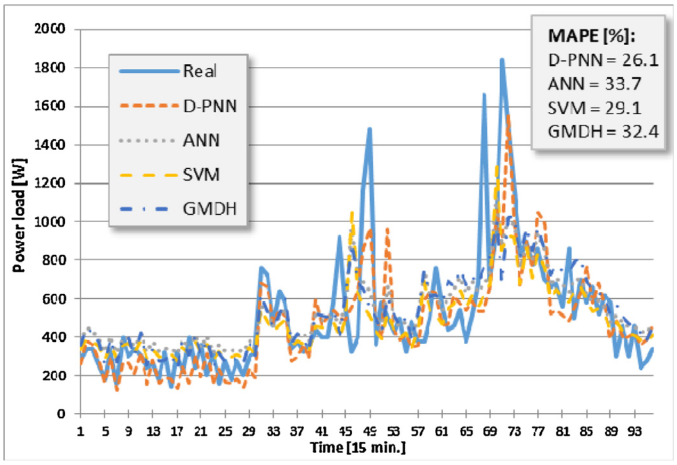

Fig. 12. ANEX: 25.1.1994 (Tuesday), RMSE: D-PNN = *212.3*, ANN = *232.0*, SVM = *245.0*, GMDH = *240.9*.

The applied 2nd data set comprises measured electric load profiles of Canadian detached houses (Fig. 11) [B]. The average electric draw of a house (in Watts) involves much more variable time behaviour (of steep pulses), evoked likely by lower customer total electricity demands and also some additional specific factors (Fig. 12). The Sundays and holidays are mostly peak consumption days in this case and their models also embody usual higher prediction errors. The load forecast models get with any worst results because of more unstable and time-variable average resident power demands; most of them result in average error

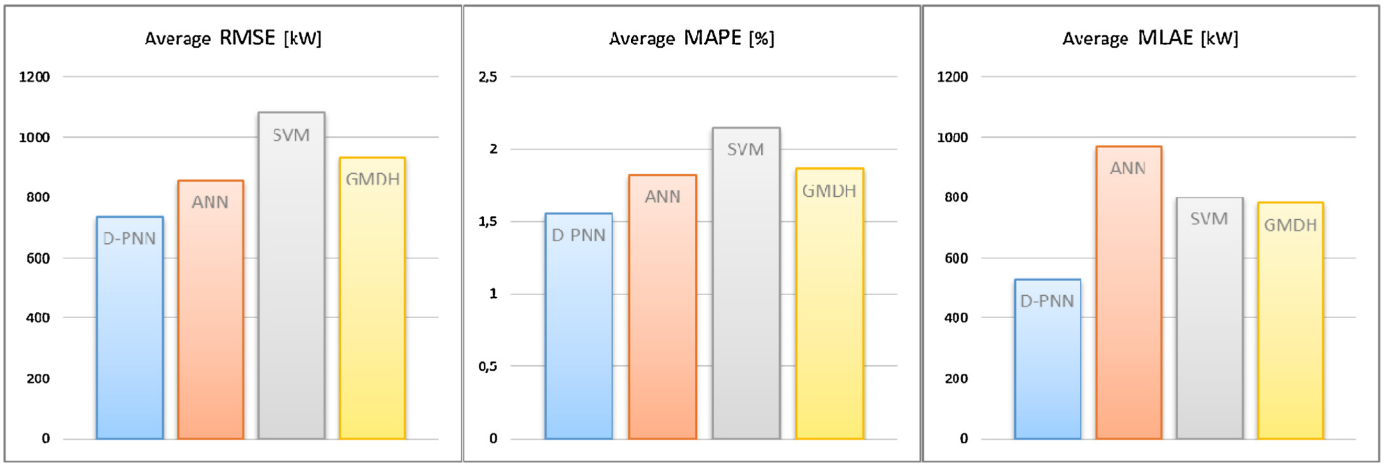

Fig. 10. 2-week average power load prediction errors (18.2.2013 to 3.3.2013) [A].

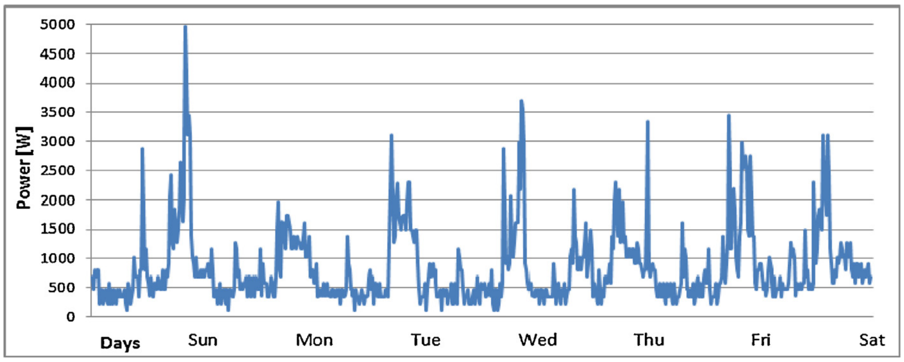

Fig. 11. A typical winter week period of the average domestic load profiles [B].

Table 2
Total average power load prediction RMSE, MAPE and MLAE (25.1.1994 to 7.2.1994) [B].

Model/error	RMSE (W)	MAPE (%)	MLAE (W)
D-PNN	334.2	37.9	1060.4
ANN	356.8	44.9	1265.6
SVM	355.2	36.4	1122.2
GMDH	356.4	42.7	1280.4

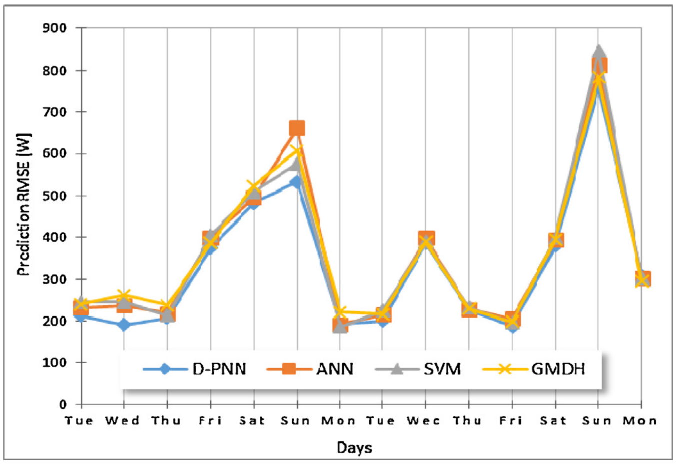

Fig. 13. 2-week daily power load prediction RMSE (25.1.1994 to 7.2.1994), average RMSE: DPNN = *334.2*, ANN = *356.8*, SVM = *355.2*, GMDH = *356.4* [B].

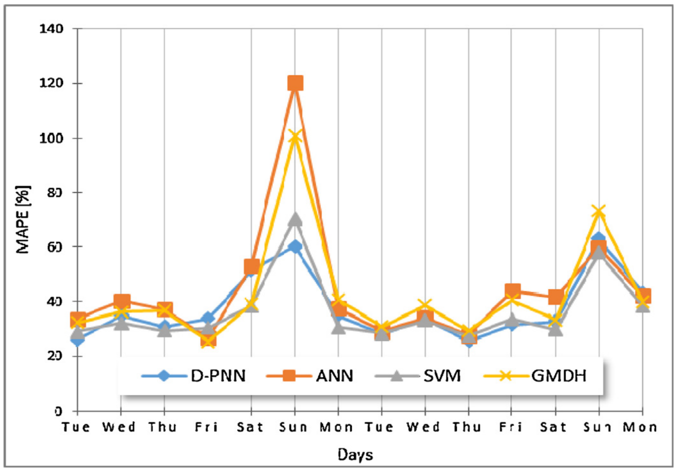

Fig. 14. 2-week daily power load prediction MAPE (25.1.1994 to 7.2.1994), Average MAPE: DPNN = *37.9*, ANN = *44.9*, SVM = *36.4*, GMDH = *42.7* [B].

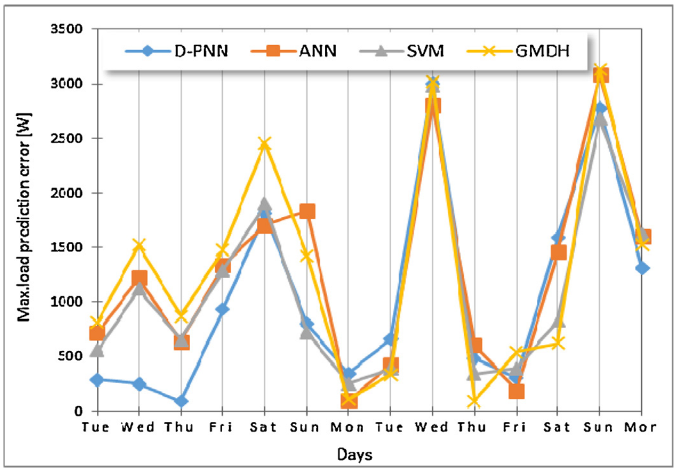

Fig. 15. 2-week daily maximal power load absolute prediction errors (25.1.1994 to 7.2.1994), average MLAE: DPNN = *1060.4*, ANN = *1265.6*, SVM = *1122.2*, GMDH = *1280.4* [B].

estimations however several ones embody higher inaccuracies. The application of historical local meteorological data (temperature) included with the power load time-series in the input vectors did not reduce the average prediction errors (Table 2) again. The combined inputs yield better results only in same day predictions, despite of the average electric draw measurements [B] are bounded to a local area (Quebec) only. The cause rests probably in the dynamic high-frequency character of the applied averaged power load time-series (Fig. 12) in this case.

Figs. 13–16 and Table 2 show the RMSE, MAPE and MLAE comparisons of all the applied 4 methods in 2-week daily power load forecasts [B], which are very similar and fail on the same days. The real prediction models could be backward tested for corresponding days of the last not trained week or there is necessary to reserve a validation (test) interval of the last 4–6 h measurements in the training data set. 3–6 weeks training periods seem to define optimal learning schemes for all the applied techniques. It is also necessary to consider legal holidays, the data of which competent (last) week working days (or alternatively Sundays) might replace in some weeks to eliminate the different type of day load cycles in respect of the rest of weeks in the training interval.

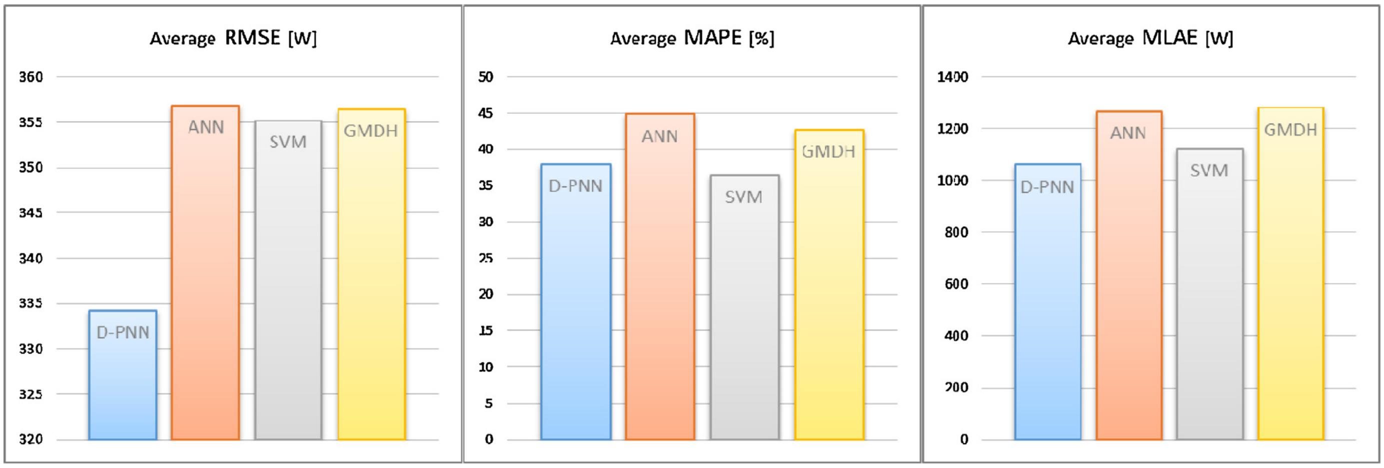

Fig. 16. 2-week average power load prediction errors (25.1.1994 to 7.2.1994) [B].

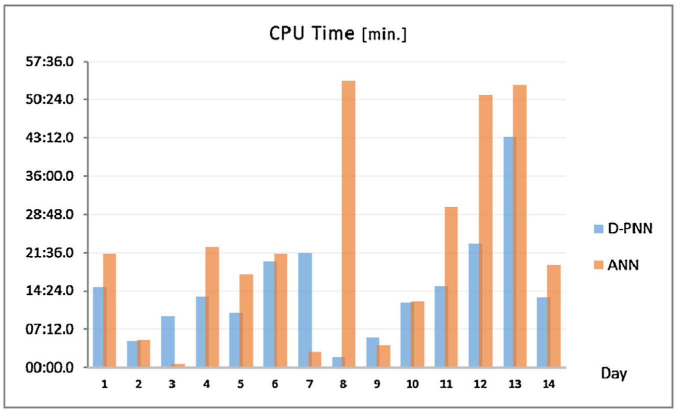

Fig. 17. The average CPU time of daily prediction models: D-PNN = *15:04*, ANN = *22:38* [A].

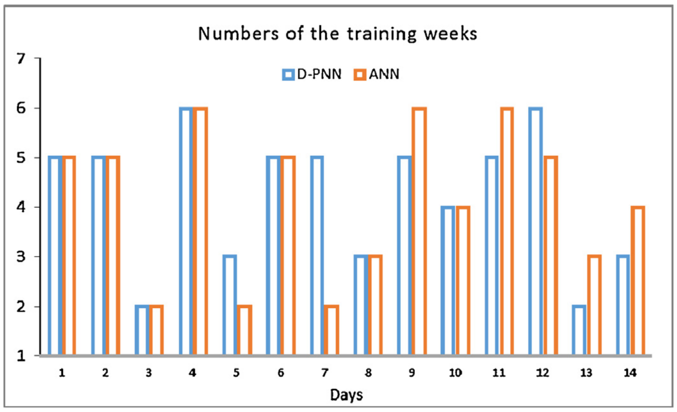

Fig. 18. The optimal numbers of the training weeks of D-PNN and ANN daily models [A].

The computational time required for the D-PNN and ANN model daily solutions using the 1st data set [A] is compared in Fig. 17. (PC Dell, 2.4 GHz, Intel core i7), it is in the order of minutes (around 1–5 min) in case of the 2nd data set [B]. The Fig. 18 compares optimal numbers of the training weeks applied by both the presented methods. SVM and GMDH models require less computational time (the parameters are calculated), ranging in the order of minutes [A] and tens of seconds [B] respectively. The ANN (less commonly D-PNN) adjustment may appear very time-consuming in some day forecasts (Fig. 17) [A], which is caused primarily by the ANN back-propagation algorithm nature (using the GSD) and the D-PNN iterative more complicated (incomplete) algorithms of the parallel processes in the parameter adjustment and neuron (block) selection that require an extra synchronization and optimization.

6. Conclusions

The proposed daily power load prediction is based on time-point data relations of subsequent day power cycles in past week periods, which ordinary differential equation solutions (of 1-variable time-series models) can describe properly. The power history affinities of several consecutive day load cycles, foregoing to the current week forecasted day denomination, primarily influence the model solutions accuracy, as no other relevant factors (e.g. weather data) were taken into account. 24-h shifted (3 day) time-points of 3–6 previous weeks define an optimal learning scheme for the networks, which trained models after apply the current week corresponding last 3-day input power series to predict the following day 24-h load values. The most important problem of the described simple prediction technique (Fig. 1) is to estimate the optimal number of training week cycles (i.e. model initialization time), which might be determined in practice with respect to the previous day model best testing results (and forecasts), as the subsequent day power cycles (time-point data relations) feature mostly quite a similar nature. Another way to determine the correct number of training week cycles is to detect the maximal backward testing error for the corresponding days of the last not trained week and then apply only the competent training period, i.e. the remaining (3–6) weeks which follow the searched error maxima week or employ some methods of a detailed data analysis [12]. The presented power load prediction models of working days apply only 3 input variables of 3 consequent day cycles (at the same 24-h shifted time-points) while the holiday models additionally combine the previous daily-similarity relations with the last day time-series (5–6 inputs in total). An increased number of input variables did not yield better average results, the applied input parameters were determined experimentally (by a trial and error method). Optimal model solutions might additionally apply appropriate tools for the parameter estimation and available convenient input exogenous factors (especially weather and environmental conditions), which can fairly lower the appearance of unexpected prediction errors however their applications did not reduce the average prediction errors in the presented 2 data set experiments. The new D-PNN method extends the GMDH polynomial neural network complete structure to form (define) and solve the general partial differential equation with sum series of polynomial relative substitution terms, which can analogously solve and substitute for a converted ordinary differential equation to model relations of day time-point variables of 1-variable historical power load time-series. The D-PNN selective derivative series regression is contrary to the conventional soft-computing approach, which applications are subjected to a fixed interval of absolute values of variables.

Notations

u, s	searched (modelled) functions (u_k – partial functions)
x_i, q_i	input variables
π	dimensionless units
$f(z)$	composite function
p	polynomial output
y_i	substitution fraction term output
Y	total (network) output
a, b	polynomial parameters
w_i	weights of terms
P_A	probability of neurons activation
sig	sigmoidal function
E	error function
ANN	artificial neural network
D-PNN	differential polynomial neural network
GMDH	group method of data handling
PNN	polynomial neural network
SVM	support vector machine (for regression)
DE	differential equation
Block	polynomial network node
Neuron	substituting fraction derivative term
CT	composite term (a neuron substituting for composite function derivatives)
SA	simulated annealing
PSO	particle swarm optimization
GSD	gradient steepest descent (method)
RMSE	root mean squared error
MAPE	mean absolute percentage error
MLAE	maximal load absolute error

Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project “IT4Innovations excellence in science – LQ1602”.

References

[1] G. Dudek, Pattern-based local linear regression models for short-term load forecasting, Electr. Power Syst. Res. 130 (2016) 139–147.
[2] S.M. Kelo, S.V. Dudul, Short-term Maharashtra state electrical power load prediction with special emphasis on seasonal changes using a novel focused time lagged recurrent neural network based on time delay neural network model, Expert Syst. Appl. 23 (2011) 1554–1564.
[3] A.S. Khwaja, Improved short-term load forecasting using bagged neural networks, Electr. Power Syst. Res. 125 (2015) 109–115.
[4] E.E. Elattar, J.Y. Goulerma, Generalized locally weighted GMDH for short term load forecasting, IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 42 (2012) 345–356.
[5] A.R. Khan, A. Mahmood, A. Safdar, Z.A. Khan, N.A. Khan, Load forecasting, dynamic pricing and DSM in smart grid: a review, Renew. Sustain. Energy Rev. 54 (2016) 1311–1322.
[6] K. Garcia-Ascanio, C. Mate, Electric power demand forecasting using interval time series, Energy Policy 38 (2010) 715–725.
[7] A. Antoniadis, X. Brossat, J. Cugliari, J.-M. Poggi, A prediction interval for a function-valued forecast model: application to load forecasting, Int. J. Forecast. (2016).
[8] O. Hyde, P.F. Hodnett, An adaptable automated procedure for short-term electricity load forecasting, IEEE Trans. Power Syst. 12 (1) (1997) 84–93.
[9] C. Unsihuay-Vila, A.C. Zambroni, J.W. Marangon-Lima, P.P. Balestrassi, Electricity demand and spot price forecasting using evolutionary computation combined with chaotic nonlinear dynamic model, Electr. Power Energy Syst. 32 (2010) 108–1116.
[10] R. Mamlook, O. Badran, E. Abdulhadi, A fuzzy inference model for short-term load forecasting, Energy Policy 37 (2009) 1239–1248.
[11] Ye Rena, P.N. Suganthana, N. Srikanthb, G. Amaratunga, Random vector functional link network for short-term electricity load demand forecasting, Inf. Sci. (2016).
[12] M. Ghayekhloo, M.B. Menhaj, M. Ghofrani, A hybrid short-term load forecasting with a new data preprocessing framework, Electr. Power Syst. Res. 119 (2015) 138–148.
[13] G.-F. Fan, L.-L. Peng, W.-C. Hong, F. Sun, Electric load forecasting by the SVR model with differential empirical mode decomposition and autoregression, Neurocomputing 173 (2016) 958–970.
[14] A.I. Sarwat, Weather-based interruption prediction in the smart grid utilizing chronological data, J. Modern Power Syst. Clean Energy (2015) 1–8.
[15] A.G. Ivakhnenko, Polynomial theory of complex systems, IEEE Trans. Syst. Man Cybern. 1 (4) (1971) 364–378.
[16] S. Dipti, Energy demand prediction using GMDH networks, Neurocomputing 72 (2008) 625–629.
[17] N.Y. Nikolaev, H. Iba, Adaptive Learning of Polynomial Networks. Genetic and evolutionary computation, Springer, New York, 2006.
[18] L. Zjavka, V. Snášel, Constructing ordinary sum differential equations using polynomial networks, Inf. Sci. 281 (2014) 462–477.
[19] G.A. Darbellay, M. Slama, Forecasting the short-term demand for electricity, Int. J. Forecast. 16 (2000) 71–83.
[20] H.S. Hippert, C.E. Pedreira, R.C. Souza, Neural networks for short-term load forecasting – a review and evaluation, IEEE Trans. Power Syst. 16 (1) (2001) 1239–1248.
[21] D. Randall, Dimensional Analysis, Scale Analysis, and Similarity Theories, 2012, September.
[22] W.D. Curtis, J. David Logan, W.A. Parker, Dimensional analysis and the pi theorem, Linear Algebra Appl. 47 (1982) 117–126.
[23] L. Zjavka, Numerical weather prediction revisions using the locally trained differential polynomial network, Expert Syst. Appl. 44 (2016) 265–274.
[24] S. Lee, S. Soak, S. Oh, W. Pedrycz, M. Jeon, Modified binary particle swarm optimization, Progr. Nat. Sci. 18 (2008) 1161–1166.
[25] L. Zjavka, W. Pedrycz, Constructing general partial differential equations using polynomial and neural network, Neural Netw. 73 (2016) 58–69.
[26] D. Bertsimas, J. Tsitsiklis, Simulated annealing, Stat. Sci. 8 (1) (1993) 10–15.
[27] N.Y. Nikolaev, H. Iba, Polynomial harmonic GMDH learning networks for time series modeling, Neural Netw. 16 (2003) 1527–1540.
[28] T. Joachims, Making large-scale SVM learning practical, in: Advances in Kernel Methods – Support Vector Learning, MIT Press, 1999.
[29] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (2011), 27:1–27:27.

Further reading

[A] National Grid, U.K. Electricity Transmission, http://www2.nationalgrid.com/UK/Industry-information/Electricity-transmission-operational-data/Data-explorer/.
[B] ANEX 54, Measured Canadian Occupant-driven Electrical Load Profiles, http://www.iea-annex54.org/annex42/data.html.
[C] Weather Underground, Historical weather and forecasts, www.wunderground.com/history/.
[D] MySVM, http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html.
[E] LibSVM, http://www.csie.ntu.edu.tw/~cjlin/libsvm.