Deep Neural Network Modeling for Big Data Weather Forecasting

James N. K. Liu, Yanxing Hu, Yulin He, Pak Wai Chan and Lucas Lai

Abstract The coming of the big data era brings the opportunities to greatly improve the forecasting accuracy of weather phenomena. Specifically, weather change is quite a complex process that is affected by thousands of variables. In the traditional computational intelligence models, we have to select the features from variables according to some fundamental assumptions, thus the correctness of these assumptions may crucially affect the prediction accuracy. Meanwhile, the principle of big data is to let data speaking, which means, when the volume of data is big enough, the hidden statistical disciplines in domain data will be revealed by the data set itself. Therefore, if massive volume of weather data is employed, we may be able to avoid using assumptions in the models, and we have the opportunity to improve the weather prediction accepted by learning the correlations hidden in the data. In our investigation, we employ a new computational intelligence technology called stacked Auto-Encoder to simulate hourly weather data in 30 years. This method can automatically learn the features from massive volume of data set via layer-by-layer feature granulation, and the large size of the data set can make sure that the complex deep model does avoid the overfitting problem.

J. N. K. Liu (✉) · Y. Hu · L. Lai
Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong
e-mail: james.liu@polyu.edu.hk

Y. Hu
e-mail: csyhu@comp.polyu.edu.hk

L. Lai
e-mail: caskclai@yahoo.com

Y. He
College of Mathematics and Computer Science, Hebei University, Baoding, China
e-mail: yulinhe@ieee.org

P. W. Chan
Hong Kong Observatory, 134A Nathan Road, Kowloon, Hong Kong
e-mail: pwchan@hko.gov.hk

W. Pedrycz and S.-M. Chen (eds.), *Information Granularity, Big Data, and Computational Intelligence*, Studies in Big Data 8,
DOI: 10.1007/978-3-319-08254-7_19,
© Springer International Publishing Switzerland 2015

The experimental results demonstrate that using the new represented features in the classical model can obtain higher accuracy in time series problems.

Keywords Weather forecasting · Big data · Deep Neural Network

1 Introduction

From the beginning of human's history, people never cease their efforts on predicting the trend of weather changes. Every step forward of weather forecasting technology has great academic and practical significance. This is not only because of the changes of climate that may greatly impact people's daily life, but also the fact that the advancing investigation of weather forecasting can reflect the progress of human's ability to know the earth.

Many significant research efforts are utilized to develop weather forecasting methods including computational intelligence technologies that have been accepted as appropriate means for weather forecasting and reported encouraging results since 1980s [6, 7, 17, 19, 21, 32]. However, the coming of the big data era brings the opportunities to improve the forecasting accuracy of weather phenomena in advance. Some conventional difficulties in weather forecasting tasks are expected to be solved with big data/large volume of weather information. Specifically, for weather forecasting tasks, the variation tendency of atmospheric phenomenon is quite unstable and complex, therefore, thousands of related variables are changing every second so that a small change of a certain variable may greatly affect the weather condition [16]. Unfortunately, the number of variables which can be handled in a certain model is limited. Especially, for computational intelligence models, if too many variables are employed, the overfitting problem is very difficult to be avoided with smaller number of training samples [3]. Accordingly, some fundamental assumptions are required, and the accuracy of the forecasting results highly depends on the correctness of initial condition of assumptions [15, 33].

The conception of "big data" refers to the increasing volume of the data sets that used to analyze problems in different research domains [37]. Combined with statistical methods and computational intelligence technologies, big data has brought a revolution to many traditional research fields including the meteorology, genomics, connectomics, complex physics simulations, and biological and environmental research [30], etc. The principle of big data is to "let data speaking", which means, when the volume of data is big enough, the hidden relevance in data set will be revealed via the statistical disciplines [1, 9, 31, 38]. Therefore, if massive weather data is employed, we may avoid using assumptions in our model, and we have the opportunity to directly analyze the correlations hidden in the weather data. In so doing, the generalization of the models and accuracy of the results are expected to be improved ultimately.

The true significance of the term "big data" not only concentrates on larger size of the data sets, but also refers to the suitable strategy to process the obtained large data set. Computational intelligence models, particularly Neural Networks (NNs), are good tools to discover the statistical rules hidden in the big data sets and have obtained some successful reputation in the previous big data applications [12, 34]. In computational intelligence field, a very prevalent conventional conception was that shallow and simple models, e.g. Support Vector Machines (SVMs) and Single Layer Feedforward Networks (SLFNs) can provide better performance than complex and deep models, e.g. NNs with deep (multi-layer) architectures in the big data environment [3, 35]. Meanwhile, previous NNs with multi-layer architecture have their own inferiorities including (1) huge computational complexity; (2) a complex NN model with too many parameters is inevitable to the overfitting problem. Nevertheless, the studies since 2006 undertaken by Hinton [13, 14] and followed by other researchers hold on the opposite conception: (1) NNs with deep structure may provide a superior learning capacity [3, 18]; (2) the newest proposed Deep Neural Network (DNN) approach, also well-known as Deep Learning (DL), employs a so-called layer−wise unsupervised pre−training mechanism to solve the training difficulties efficiently [5] and (3) particularly, in big data environment, despite the number of parameters in DNN is more than that of shallow models, the overfitting problem can also be avoided because of the huge amount of data samples [28].

Compared with simple and shallow models, NNs with deep architecture can provide a higher learning ability. Although the back-propagation NNs with three layers have been proved that can theoretically approximate any nonlinear functions with arbitrary precision [10], functions that can be compactly represented by a deep architecture might be required to handle an exponential number of computational elements (parameters) to be represented by a depth architecture. More precisely, functions that can be compactly represented by a depth k architecture might be requiring an exponential number of computational elements to be represented in a depth $k-1$ architecture. Since the number of computational elements one can afford depends on the number of training examples available for tuning or selecting them, the consequences are not only computational but also statistical: poor generalization may be expected when using an insufficiently depth architecture for representing some functions [3, 5].

The core technologies in DNN is the layer−wise unsupervised pre−training mechanism, by such training method, the original information in the raw data can be represented [5] or granulated [24, 25]. By such granulation, the raw data in original feature space may be mapped into a new feature space, and the principle for such mapping is an information granula of interesting [2]. In a larger data environment, the significance of granulation becomes more important, we need some approaches for mining the knowledge in big data sets. With the granulation, the hidden relevance in the data set maybe extracted and represented layer by layer [5, 24]. Such a feature representation may greatly improve the performance of traditional computational intelligence models.

In our investigation, we apply a multi-layer model to predict the weather change in the next 24 hour with a big data set. The massive data involving millions

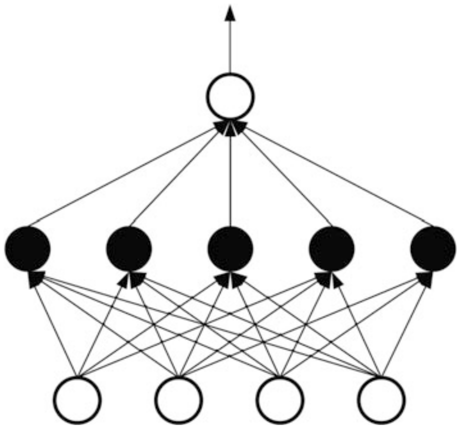
Fig. 1 A typical shallow feedforward network with one hidden layer

of weather records is provided by The Hong Kong Observatory (HKO).[1] Our training method is to use the latest proposed greedy layer-wise unsupervised pre-training algorithm followed by a supervised fine-tuning. In detail, we choose a revised autoencoder algorithm to build the network, the DNN is used to learn the features from the larger volume of raw data, and we will evaluate the learned features according to the prediction accuracy. The contribution and significance of our investigation demonstrates that: compared with the classical models, NN with Deep architectures can improve the prediction accuracy in weather forecasting field; moreover, the positive results show the potential of DNN model in big data; last but not least, DNN has won some encouraging results in research field including Computer Vision [14], Speech Recognition [23], Natural Linguistic Programming [4] and Bioinformation, our investigation will show that DNN also has great potential in time series problems, especially in weather forecasting domain [22].

This chapter is organized as follows: in Sect. 2 we will introduce some background knowledge, mainly including how to train a DNN model layer by layer. Section 3 briefly discusses the DNN model for weather data simulation. Section 4 discusses the experiments along with some comparative analysis. The last section gives the conclusion and future work.

2 Background Knowledge

In this section, some background knowledge is presented. Specifically, we mainly introduce the greedy layer−wise unsupervised pre−training approach and the stacked Auto-Encoder based DNN.

1 http://www.hko.gov.hk/contente.htm.

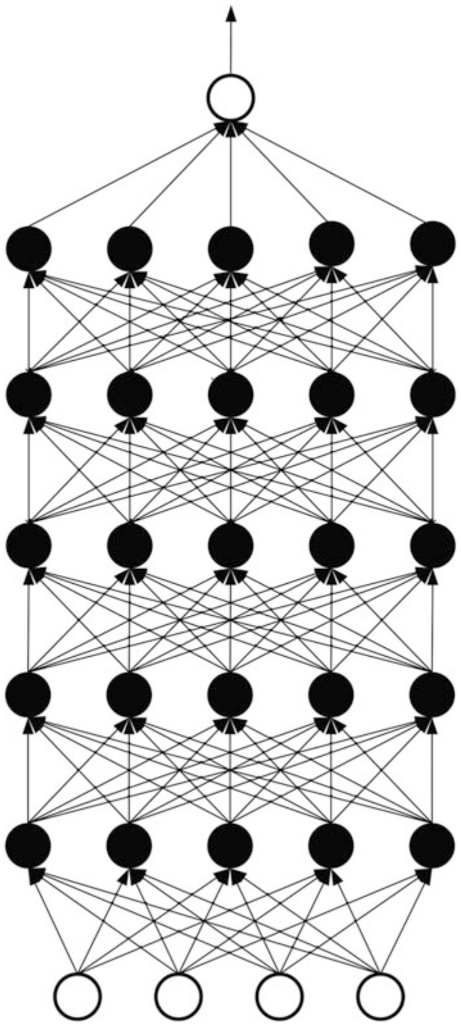
Fig. 2 A traditional NN with deep architecture (simply adding extra hidden layers to the shallow model), that shows complex structure, hard to train, and easily overfitting

2.1 *Greedy Layer-Wise Unsupervised Pre-training: Auto−encoder Granulation*

The essential challenge in training a NN with deep architecture is to deal with the strong dependencies that exist during training between the parameters across layers [11]. In previous investigations, researchers found that simply adding layers to a classical shallow Feedforward Network cannot overwhelm the mentioned challenge. Figure 1 shows the architecture of the classical SLFNs and Fig. 2 gives the earlier model of NN with multi-layer architecture.

Training deep architectures involves a potentially intractable non-convex optimization problem, and there were no inadequate algorithms for training fully-connected deep architectures until Hinton et al. introduced a learning algorithm that greedily trains one layer at a time in 2006 [14]. Shortly after, strategies for building deep architectures from related variants were proposed by Bengio [22] and Ranzato [29]. They solved the training problem of DNN in two phases: for the first phase, unsupervised pre-training, all layers are initialized using this layer-wise unsupervised learning signal; for the second phase, fine-tuning, a global training criterion (a prediction error, using labels in the case of a supervised task) is minimized. Such training approach is called the greedy layer−wise unsupervised pre−training, and DNN training with this mechanism since then has been applied with success in many fields, which is widely known as the term "Deep Learning".

Several unsupervised training models have been proposed and investigated since 2006. These models are categorized into the family of greedy layer−wise unsupervised pre−training approaches and employed to build the deep architecture of NN, e.g. Restricted Boltzmann Machines (RBMs), Stacked Auto-Encoder, CNNs, etc. According to Andrew NG in 2007, the selection of different greedy layer-wise unsupervised pre-training approaches in DNN gives little effect to the final result [8]. Therefore, with the consideration of the attribute type of the weather data, i.e., the collected data are all real numbers, in this investigation, we choose the Stacked Auto-Encoder to build the deep architecture of our NN model.

The Stacked Auto-Encoder, as its name suggests, is a stacked architecture NN that applies Auto-Encoder in each layer. In computational intelligence field, NN means a network of neurons with different architectures (e.g., NN in Figs. 1 and 2). A single "neuron" is a computational unit that taken as input vector $X = x_1, x_2, \ldots, x_n$ (and a "+1" intercept term), and outputs $h_{W,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b)$, where $f : \Re \mapsto \Re$ is called the activation function, and W is the weight matrix that stands for the connection among different neurons in the network. In most of cases, sigmoid function $f(z) = \frac{1}{1+\exp(-z)}$ is chosen as the activation function. A typical Auto-Encoder tries to learn a function $h_{w,b}(x) \approx x$. In other words, it is trying to learn an approximation to the identity function, so as to output $\hat{x}$ that is similar to x. The identity function seems a typically trivial function trying to learn; but by placing constraints on the network, such as by limiting the number of hidden neurons, we can discover interesting structure about the data [29], e.g., for a data set, suppose that the original samples are collected from a 100-dimensional feature space, i.e. $x \in \Re^{100}$, set that there are 50 hidden neurons in the hidden layer, based on the requirement $h_{w,b}(x) \approx x$, the network is forced to learn a compressed representation of the input. That is, given only the vector of hidden unit activations $a^{(2)} \in \Re^{50}$, it must try to reconstruct the 100-dimensional input x. An illustration of Auto-Encoder is shown in Fig. 3. If the inputs were completely random, each x_i comes from an I.I.D. Gaussian independent of the other features, then this compression task would be very difficult. But if there is a certain structure hidden in the data, for example, if some of the input features are correlated, such as

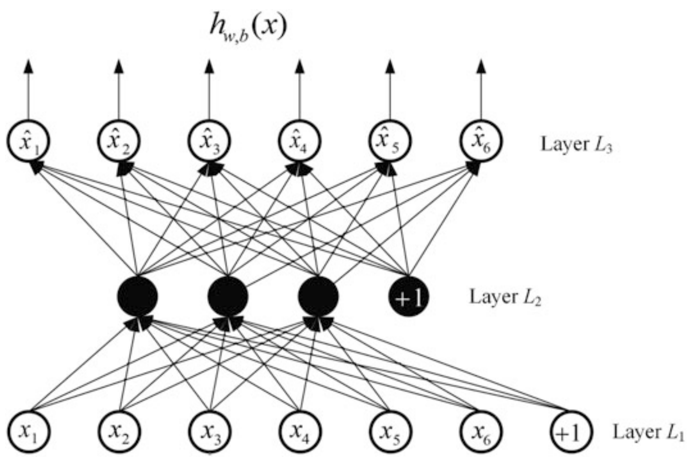

Fig. 3 An illustration of auto-encoder algorithms. Layer L_1 is the input layer, and L_3 is the output layer. Via hidden layer L_2, we hope to represent the information x in layer L_1, so that the output $\hat{x}$ in L_3 can approximate the raw data x

in the feature space of time series analysis, then this algorithm will be able to discover some of those correlations.

The loss function of Auto-Encoder is:

$$J(W,b) = \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}\left(x^{(i)}\right) - x^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \qquad (1)$$

where m is the number of training samples. The objective of the Auto-Encoder is to minimize Eq. (1) in order to make sure that the output $h_{W,b}(x^{(i)})$ can approximate the raw data $x^{(i)}$ as far as possible. The second term in Eq. (1) is a regularization term (also called a weight decay term) controlled by the weight decay parameter λ that tends to decrease the magnitude of the weights, and helps prevent overfitting. We can minimize Eq. (1) by *gradient descent* to compute the configuration of the network.

In some special cases, the number of hidden neurons is large (perhaps even greater than the number of dimensions of input vectors), we can still discover interesting structure by imposing other constraints on the network. In particular, if we impose a **sparsity** constraint on the hidden neurons, then the autoencoder will still discover interesting structure in the data, even if the number of hidden neurons is large. To achieve this, we would like to define a ρ as sparsity parameter, typically a small value close to zero. In other words, if we use $a_j(x)$ to denote the activation of the jth hidden neuron when the network is given a specific input x, we hope that the average activation $\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j(x^{(i)})\right]$ of each hidden neuron to be close to ρ. Then a revised loss function is employed as:

$$J_{\text{sparse}}(W,b) = J(W,b) + \beta \sum_{j=1}^{n} \text{KL}\left(\rho || \hat{\rho}_j\right), \tag{2}$$

where n is the number of the hidden neurons, $J(W,b)$ is defined in Eq. (1), and β controls the weight of the sparsity penalty term. The term $\hat{\rho}_j$ (implicitly) depends on W, b, and the last term $\text{KL}(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$ is the Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean $\hat{\rho}_j$.

2.2 The Deep Neural Networks and Layer by Layer Granulations

In a stacked Auto-Encoder based DNN, for each layer, we use an Auto-encoder to train the parameters in this layer, and then combined these layers together. Specifically, in the training process of each layer, as shown in Fig. 3, the input vectors have to pass three layers, and the vectors in hidden layers (layer L_2, and for simplicity, we call the vectors in layer L_2 as the transformed vectors of the initially input vectors) are representations of the input vectors and can be used to reconstruct the input vectors. Thus, in every layer of the DNN, the input of the current layer is the output of the previous layer, then we train the input data via an Auto-Encoder, and use the transformed vectors as the output of the current layer. Figure 4 shows the detailed mechanism of stacked Auto-Encoder based DNN.

Observing the NN in Fig. 4, we can find that the principle of layer-wise unsupervised pre-training based DNN is to map the raw data into a new feature space layer by layer. For computational intelligence model, the selection of features can greatly impact the accuracy of models. Therefore, how to select features is one of the core and universal problems. The DNN provides us a method to learn features instead of manually selection like we did previously. In DNN, data in raw feature space can be mapped into a new space and regularized in each layer. Thus, in each layer, input features can be reconstructed and new features can be generated; finally, these generated features can be applied by the model in the top layer. In big data environment, the requirement of learning feature is more important, and the larger size of the data can improve the quality of the generated features and guarantee the avoidance of overfitting.

The DNN based on stacked Auto-Encoder can be also seen as a branch or an extension of Granular Computing(GC) [26]. Information granules can be regarded as collections of objects that exhibit some similarity in terms of their properties or functional appearance. Actually, early in 2001, Pedrycz has already applied NN model to process granulation problem [26]. Generally, information granules defined in some space X can be treated as a mapping:

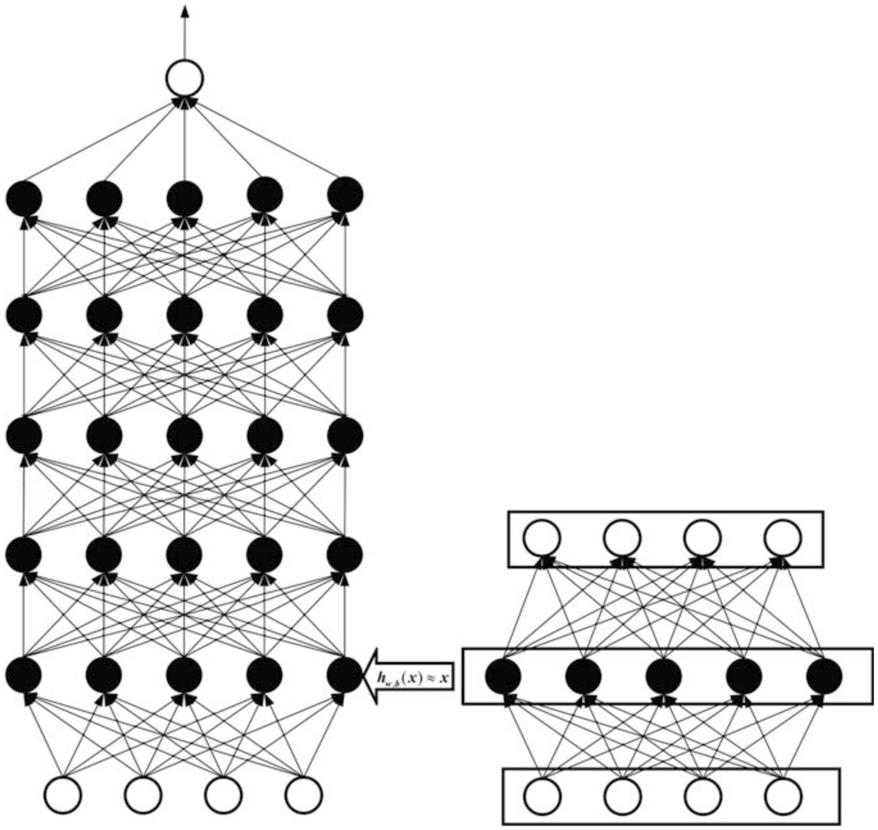

Fig. 4 A DNN with Stacked Auto-Encoder method, by which each layer is greedily pre-trained with an unsupervised Auto-Encoder algorithm and to learn a nonlinear transformation of its input (the output of the previous layer) that captures the main variations in its input, i.e. $h_{W,b}(x) \approx x$

$$A : X \to \zeta(x) \tag{3}$$

where A is an information granule of interest, ζ denotes a formal framework of information granules. From this definition, we can see that the granulation of a universe involves grouping of similar elements into granules to form coarse-grained views of the universe [39]. The only difference between classical GC and the Auto-Encoder (or layer-wise unsupervised pre-training algorithms) is that, GC is to group and reconstruct samples, but Auto-Encoder is to group and reconstruct the feature of the samples, therefore, Deep NN based on stacked Auto-Encoder can be considered as a branch or an extension of GC, which granulates the information of the raw data layer by layer.

2.3 Support Vector Regression

The principle of layer-wise unsupervised pre-training based DNN is to learn the features, therefore, we have to choose the model applied in the top layer, or output layer to process the learned features. In our experiment, SVM is employed.

Based on the Structural Risk Minimization (SRM) principle, SVM method seeks to minimize an upper bound of generalization error instead of the empirical error as in other NNs. Additionally, SVM models generate the regression function by applying a set of high-dimensional linear functions. The Support Vector Regression (SVR) function is formulated as follows:

$$y = w\phi(x) + b \tag{4}$$

where $\phi(x)$ is called the feature, which is nonlinear and mapped from the input space $\Re^n$. y is the target output value we want to estimate. The coefficients w and b are estimated by minimizing:

$$R = \frac{1}{2}\|w\|^2 + \frac{1}{n}C\sum_{i=1}^{n} L_\varepsilon(d_i, y_i) \tag{5}$$

where:

$$L_\varepsilon(d, y) = \begin{cases} |d - y| - \varepsilon, |d - y| \geq \varepsilon \\ 0, \text{otherwise} \end{cases} \tag{6}$$

Equation (5) is the risk function consisting of the empirical error and a regularization term that is derived from the SRM principle. The term $\frac{1}{n}\sum_{i=1}^{n} L_\varepsilon(d_i, y_i)$ in Eq. (5) is the empirical error (risk) measured by the ε-insensitive loss function (ε-insensitive tube); in the meanwhile, the term $\frac{1}{2}\|w\|^2$ is the regularization term. The constant $C > 0$ is taken as the regularized constant that determines the trade-off between the empirical error (risk) and the regularization term. Increasing the value of C will add importance to the empirical risk in the risk function. ε is called the tube size of the loss function and it is equivalent to the accuracy approximation placed on the training data points. Both C and ε are user-prescribed parameters.

Then the slack variables ζ and ζ^* which represent the distance from the real values to the corresponding boundary values of ε-insensitive tube are introduced. With these slack variables, Eq. (5) can be transformed to the following constraint based optimization:

Minimize:

$$R(w, \zeta, \zeta^*) = \frac{1}{2}ww^T + C^*\left(\sum_{i=1}^{n}(\zeta + \zeta^*)\right) \tag{7}$$

Subject to:

$$\begin{aligned} & w\phi(x_i) + b_i - d_i \leq \varepsilon + \zeta_i^* \\ & d_i - w\phi(x_i) - b_i \leq \varepsilon + \zeta_i \\ & \zeta_i, \zeta_i^* \geq 0, \quad i = 1, 2, \ldots, n \end{aligned} \tag{8}$$

Finally, by introducing the Lagrangian multipliers and maximizing the dual function of Eq. (4), it can be changed to the following form:

$$\begin{aligned} R(\alpha_i - \alpha_i^*) = & \sum_{i=1}^{n} d_i(\alpha_i - \alpha_i^*) - \varepsilon \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) \\ & - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha_i - \alpha_i^*) \times \left(\alpha_j - \alpha_j^*\right) (\Phi(x_i) \cdot \Phi(x_k)) \end{aligned} \tag{9}$$

with the constraints:

$$\sum_{j=1}^{n} (\alpha_i - \alpha_i^*) = 0, \quad 0 \leq \alpha_i \leq C, \quad 0 \leq \alpha_i^* \leq C, \quad i = 1, 2, \ldots, n \tag{10}$$

where α_i and α_i^* are Lagrangian multipliers which satisfy $\alpha_i \times \alpha_i^* = 0$, the general form of the regression estimation function can be written as:

$$f(x, \alpha_i, \alpha_i^*) = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x, x_i) + b \tag{11}$$

$K(x_i \cdot x)$ is called the kernel function. It is a symmetric function $K(x_i \cdot x) = (\Phi(x_i) \cdot \Phi(x))$ satisfying Mercer's conditions. When the given problem is a non-linear problem in the primal space, we may map the sample points into a high-dimensional feature space where the linear problem can be performed. Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid are four main kernel functions in use. As we discussed above, in most of the time series forecasting problems, the SVR employs RBF kernel function to estimate the nonlinear behavior of the forecasting data set because RBF kernels tend to give good performance under general smoothness assumptions.

3 Weather Prediction with Deep Neural Networks

The main task of our investigation is to employ DNN based on stacked Auto-Encoder to predict weather informations, more specifically, we hope to predict two kinds of weather information, temperatures and wind speed, in the next few hour. Univariate time series regression is the most fundamental and most widely applied model in short-term weather forecasting. Generally speaking, for a certain variable, the objective of univariate time series regression is to find the relationship between its status in a certain future time point and its status in a series of past time points, and estimate its future status via:

$$v_t = f(v_{t-1}, v_{t-2}, \ldots, v_{t-n}) \tag{12}$$

To obtain f, earlier investigations employ models such as Linear Regression, Generalized Linear Model, Autoregressive Integrated Moving Average Mode, etc. After the computational intelligence technologies become a hot research field, investigators found that some intelligence technologies models, such as NNs, SVMs, and fuzzy models could provide higher generalization on some complex, nonlinear, and unstable domains including weather forecasting tasks.

Weather data has some particularities. More specifically, there is season-to-season, and year-to-year variability in the trend of weather data. The cycle could be multi-month, multi-season or multi-year, and the main difficulty of investigations is to capture all the possible cycles.

In our investigation, we will simply input the data sets into our model. The architecture of the applied model is as the NN in Fig. 3. The input n-dimensional vector is composed by the status in $(t-1)$th, $(t-2)$th,…, $(t-n)$th time points, we try to use the DNN to represent these status, and employ a SVR to estimate the status in tth time point. Since our data set is quite large, we hope the seasonal cycles can be captured via the massive volume of data by the superior learning ability of the multi-layer structured NN.

4 Experimental Results and Analysis

From the discussion in previous sections, we can see that the key point of employing a DNN is to learn the features, or granulate raw data into a new feature space via a multi-layer NN. Therefore, our experiments concentrate on the evaluation of the learned features: two types of weather data, with very large data sets, are employed and simulated, and the comparison of results is conducted between models using raw features and models with represented features.

4.1 Data Collection

The HKO has provided great support to our investigation. Based on our collaboration with HKO, a massive volume of high quality real weather data could be applied in our experiment. Two types of historical weather data sets, the wind speed data and temperature data are employed in our model. The time range of the data sets is almost 30 year long, which covers the period from January 1, 1983 to

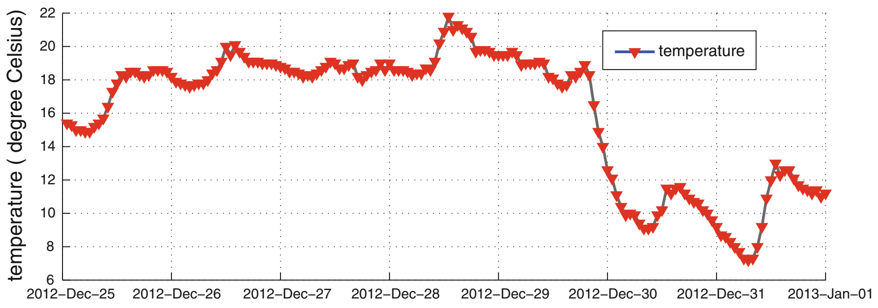

Fig. 5 The distribution of temperature data in the last week of the data set

December 31, 2012. In detail, the number of temperature records is more than 260,000, and the total number of records in the wind speed experiment is more than 1,560,000.

Compared with the temperature data which has only one dimension (measured in degree Celsius), the wind speed data has two dimensions: the polar coordinate for the wind direction (measured in degree angle) and the speed (measured in meters per second). Moreover, in the raw data set, for a certain time points, the direction of the air motion is not stable, i.e. the wind direction at that time point is not fixed. Such condition is denoted as ''variable'' in the raw data. Therefore, according to the requirement of our algorithm, we have to do some pre-processing on the data sets.

4.2 Data Pre-processing

Compared with the temperature data which is a scalar quantity only having one dimension (as Fig. 5), the wind speed data (in a fixed horizontal plane) is a vector quantity that has two dimensions in the polar coordinate (as Fig. 6), i.e. the angle to show its direction and the speed to measure the velocity in this direction: the polar coordinate and the speed [27]. However, since our model focuses on univariate time series problems, we have to transform the data set to satisfy the model's requirement. According to the physical significance of the two dimensions, we denote the angle as θ and the speed as v to obtain:

$$v^0 = \cos\theta \cdot v \tag{13}$$

where v^0 is the vector components of the wind speed in 0° angle direction (as Fig. 7). Thus, what we actually simulate is the time series of the speed component of the air motion in 0 degree angle direction. Moreover, there are about 3 % wind speed data with the direction valued as "variable", for such condition, we consider it as a missing value in the data set and use the average value of the wind direction in its previous time point and its next time point to replace the value "variable".

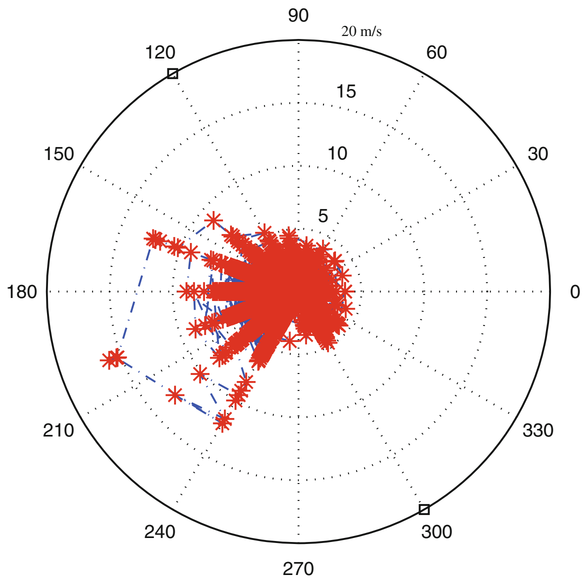

Fig. 6 The distribution of wind speed data in polar coordinate

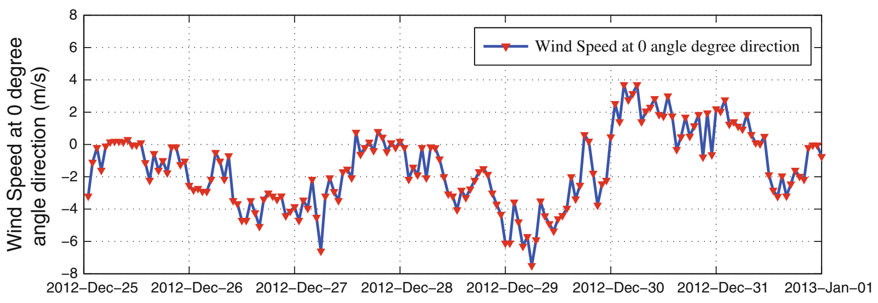

Fig. 7 The distribution of wind speed at a fixed direction in the last week of the data set

4.3 Evaluation Criteria

Three criteria are applied in our investigation to evaluate the prediction performance of the used models.

Normalized Mean Squared Error (NMSE) measures the deviation between the actual values and the predicted values. The smaller the values are, the closer the predicted values to the actual values. The formula of NMSE is:

$$\mathrm{NMSE} = 1/\left(\delta^2 n\right)\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{14}$$

where

$$\delta^2 = 1/(\mathrm{n}-1)\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{15}$$

Directional symmetry (DS) indicates the percentage correctness of the predicted direction. The formula of DS is:

$$\mathrm{DS} = (100/n) \times \sum_{i=1}^{n} d_i \tag{16}$$

where

$$d_i = \begin{cases} 1, (x_i - x_{i-1})(\hat{x}_i - \hat{x}_{i-1}) \geq 0 \\ 0, \text{otherwise} \end{cases} \tag{17}$$

We also employ R^2 value to evaluate the whole simulation ability of our model, the R^2 indicates how well our model can explain the raw data set. The R^2 value of a model is:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{18}$$

where $SS_{tot} = \sum_{i=1}^{n} (\hat{x}_i - x_i)^2$ and $SS_{tot} = \sum_{i=1}^{n} (\hat{x}_i - \bar{x})^2$.

4.4 Experimental Results and Discussion

In our first experiment, we use a 4-layer DNN model to predict the temperature in the next time point. More specifically, we tried to use the 7-day hourly temperature data to forecast the temperature in the next 24 hour, The NN model is with a single input layer, two stack-organized Auto-Encoder layers, and the top layer which used SVR to output the prediction results. In this experiment, since the sparsity is less considered in weather forecasting, we adjusted the value of β to a relatively small value, and set the number of hidden nodes as 84. The experiments were conducted in a CPU cluster with 6 Intel i7 processors with MatLab 2012a. The experiment is based on the 10-fold cross-validation, thus in each cycle, the training set has more than 230,000 randomly selected records, and about 26,000 samples are selected as training set. The result is compared with classical SVR. Note that the parameter in the classical SVR is set as same as in the top SVR layer of our model. Table 1 and Fig. 8 give the result.

From Fig. 8 we can observe that both SVR and DNN can simulate the real data very well after training with a big data set, the predicted results are almost coincide

Table 1 The comparison of temperature prediction by SVR and DNN

Model	NMSE	DS	R^2
Classical SVR	2.179e−2	0.75	0.872
DNN with SVR in top layer	8.117e−3	0.79	0.915

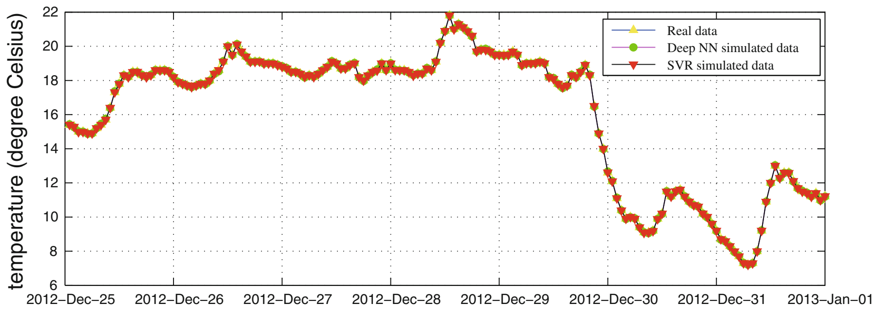

Fig. 8 The results of temperature prediction for the date in the last week

with the real data. Such an encouraging result demonstrates the advantage of big data for univariate time series problem: the massive volume of the data set can help us to approximate the statistical discipline with a higher accuracy.

Results shown in Table 1 demonstrate the positive role of the DNN in the temperature simulation task. As we discussed in above, both of the two models have the same configuration in SVR part, the only difference is that in the DNN, we represent/granulate the raw features, and use the new features to train the SVR in the top layer. We can see that the DNN model greatly reduces the NMSE (the NMSE in SVR model has already been very small, but after the feature representation/granulation, the NMSE becomes even smaller), and the higher R^2 value also proved that, with the represented features via a DNN, the SVR model in the top layer can learn the raw data much better.

From Figs. 5 and 7, we may observe that compared with wind speed data, the fluctuation of temperature data (in short time) is much more smooth. Therefore, temperature simulation is not a challenging task. actually, in the training process of our experiment, the classical SVR model takes more time than DNN. This phenomenon may be attributed to two aspects: (1) the temperature data is very smooth so that the convergence of the NN is very quick [4]; (2) the represented features can reveal the principle of the given data set and consequently improve the performance of SVR [20, 36]. Therefore, the shortening of the training time gives two things: (1) the new feature space that reconstructed via DNN has positive effect for time series tasks; (2) the significance of our positive results obtained in temperature data set is limited.

In the second experiment, we change the data set. The wind speed data is employed. Since the change of wind speed data is much more unstable, the

Table 2 The comparison of wind speed prediction by SVR and DNN

Model	NMSE	DS	R^2
Classical SVR	0.3721	0.72	0.851
DNN with SVR in top layer	0.2522	0.83	0.871

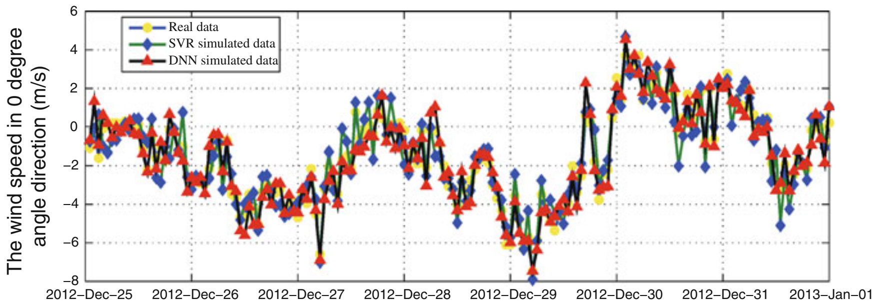

Fig. 9 The results of temperature prediction for the date in the last week

simulation of wind speed data is more difficult and has more academic and practical significance. We made some modifications on our model in the second experiment: we added two Auto-Encoder layers in the model in order to improve the learning ability of the network. The results are shown in Table 2 and Fig. 9.

Figure 9 shows the simulation results of the wind speed data in the last week. From Fig. 9, we can observe that after training with a big volume of wind speed data, the model can capture the main trend of changes, and the DNN can give a better performance than simply using SVR. Inspecting the criteria in Table 2, DNN can return a lower NMSE and higher R^2 value. Note that the DS value is greatly improved when the data is simulated with the DNN model, this maybe caused by the fact that features generated via DNN may have the largest possible variation, and such fact shows that the principle of DNN may be considered as an advanced form of Principal Component Analysis (PCA).

Our experiments only make comparison between Classical SVR and Stacked Auto-Encoder DNN with SVR in the top layer. Actually, some other models can also be applied to deal with weather data related time series problem. However, the main objective of our investigation is to attest the models' performance with the new represented/granulated features. The results demonstrate that compared with the raw features, the obtained features can explain the principle of the raw data set better. What is more, the DNN can be combined with many other models, and the obtained features can be employed to improve the performances of most models in computational intelligence field.

5 Conclusion and Future Work

Big data may bring revolutions to many research fields including weather forecasting. In this chapter, we explore an approach that using computational intelligence technologies to process massive volume of data. The proposed DNN model may granulate the features of the raw weather data layer by layer, and experimental results show that the new obtained features can improve the performances of classical computational intelligence models.

The contribution of our investigation is significant: we give an approach that using computational intelligence method to learn features with big data, and our experiments demonstrate that the DNN algorithm also has the potential to address time series problems.

The main future work of our investigation is that, we will try to employ our model on some more difficult weather data, such as rain fall data set; and moreover, we will continue exploring the theoretical principle of computational intelligence, especially, we will try to give the mathematical explanation of the DNN.

Acknowledgment The authors would like to acknowledge the partial support of the CRG grants G-YL14, G-YM07 of The Hong Kong Polytechnic University.

References

1. Aronova, E., Baker, K.S., Oreskes, N.: Big science and big data in biology: from the international geophysical year through the international biological program to the long term ecological research (LTER) network, 1957–present. Hist. Stud. Nat. Sci. **40**(2), 183–224 (2010)
2. Bargiela, A., Pedrycz, W.: Granular Computing: An Introduction. Springer, Berlin (2003)
3. Bengio, Y.: Learning deep architectures for AI, vol. 2. Now Publishers Inc. (2009)
4. Bengio, Y.: Deep learning of representations for unsupervised and transfer learning. J. Mach. Learn. Res. Proc. Track **27**, 17–36 (2012)
5. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. Adv. Neural Inf. Process. Syst. **19**, 153 (2007)
6. Chen, S.M., Hwang, J.R.: Temperature prediction using fuzzy time series. IEEE Trans. Syst. Man Cybern. B: Cybern. **30**(2), 263–275 (2000)
7. Chen, S.M., Tanuwijaya, K.: Multivariate fuzzy forecasting based on fuzzy time series and automatic clustering techniques. Expert Syst. Appl. **38**(8), 10594–10605 (2011)
8. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics, pp. 215–223 (2011)
9. Condie, T., Mineiro, P., Polyzotis, N., Weimer, M.: Machine learning for big data. In: Proceedings of the 2013 International conference on Management of Data, pp. 939–942. ACM (2013)
10. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. **2**(4), 303–314 (1989)
11. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**, 625–660 (2010)

12. Herrero, J., Valencia, A., Dopazo, J.: A hierarchical unsupervised growing neural network for clustering gene expression patterns. Bioinformatics **17**(2), 126–136 (2001)
13. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
14. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
15. Kissinger, C.R., Gehlhaar, D.K., Fogel, D.B.: Rapid automated molecular replacement by evolutionary search. Acta Crystallogr. D Biol. Crystallogr. **55**(2), 484–491 (1999)
16. Kuligowski, R.J., Barros, A.P.: Localized precipitation forecasts from a numerical weather prediction model using artificial neural networks. Weather Forecast. **13**(4), 1194–1204 (1998)
17. Kwong, K., Wong, M.H., Liu, J.N., Chan, P.: An artificial neural network with chaotic oscillator for wind shear alerting. J. Atmos. Oceanic Technol. **29**(10), 1518–1531 (2012)
18. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 609–616. ACM (2009)
19. Liu, J., Kwong, K.M., Chan, P.W.: Chaotic oscillatory-based neural network for wind shear and turbulence forecast with lidar data. IEEE Trans. Syst. Man Cybern. C: Appl. Rev. **42**(6), 1412–1423 (2012)
20. Liu, J.N., Hu, Y.: Application of feature-weighted support vector regression using grey correlation degree to stock price forecasting. Neural Comput. Appl., 1–10 (2013)
21. Maier, H.R., Jain, A., Dandy, G.C., Sudheer, K.P.: Methods used for the development of neural networks for the prediction of water resource variables in river systems: current status and future directions. Environ. Model Softw. **25**(8), 891–909 (2010)
22. Miller, S.M., Geng, Y., Zheng, R.Z., Dewald, A.: Presentation of complex medical information: Interaction between concept maps and spatial ability on deep learning. Int. J. Cyber Behav. Psychol. Learn. (IJCBPL) **2**(1), 42–53 (2012)
23. Mohamed, A.R., Dahl, G.E., Hinton, G.: Acoustic modeling using deep belief networks. IEEE Trans. Audio Speech Lang. Process. **20**(1), 14–22 (2012)
24. Pedrycz, W., Bargiela, A.: Granular clustering: a granular signature of data. IEEE Trans. Syst. Man Cybern. B: Cybern. **32**(2), 212–224 (2002)
25. Pedrycz, W., Skowron, A., Kreinovich, V.: Handbook of Granular Computing. Wiley, New York (2008)
26. Pedrycz, W., Vukovich, G.: Granular neural networks. Neurocomputing **36**(1), 205–224 (2001)
27. Pielke, R.A.: Mesoscale Meteorological Modeling. Academic Press, US (2002)
28. Raina, R., Madhavan, A., Ng, A.Y.: Large-scale deep unsupervised learning using graphics processors. ICML **9**, 873–880 (2009)
29. Ranzato, M., Huang, F.J., Boureau, Y.L., Lecun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07, pp. 1–8. IEEE (2007)
30. Reichman, O., Jones, M.B., Schildhauer, M.P.: Challenges and opportunities of open data in ecology. Science (Washington) **331**(6018), 703–705 (2011)
31. Rushton, J.P., Irwing, P.: A general factor of personality (GFP) from two meta-analyses of the big five: and. Personality Individ. Differ. **45**(7), 679–683 (2008)
32. Sánchez Reinoso, C., Cutrera, M., Battioni, M., Milone, D., Buitrago, R.: Photovoltaic generation model as a function of weather variables using artificial intelligence techniques. Int. J. Hydrogen Energy **37**(19), 14781–14785 (2012)
33. Sundqvist, H., Berge, E., Kristjánsson, J.E.: Condensation and cloud parameterization studies with a mesoscale numerical weather prediction model. Mon. Weather Rev. **117**(8), 1641–1657 (1989)
34. Suzuki, K., Zhang, J., Xu, J.: Massive-training artificial neural network coupled with laplacian-eigenfunction-based dimensionality reduction for computer-aided detection of polyps in CT colonography. IEEE Trans. Med. Imaging **29**(11), 1907–1917 (2010)

35. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3360–3367. IEEE (2010)
36. Wang, X., He, Q.: Enhancing generalization capability of SVM classifiers with feature weight adjustment. In: Knowledge-Based Intelligent Information and Engineering Systems, pp. 1037–1043. Springer, Berlin (2004)
37. White, T.: Hadoop: The Definitive Guide. O'Reilly (2012)
38. Wu, X., Zhu, X., Wu, G.Q., Ding, W.: Data mining with big data. IEEE Trans. Knowl. Data Eng. **26**(1), 97–107 (2014)
39. Yao, Y.: Information granulation and rough set approximation. Int. J. Intel. Syst. **16**(1), 87–104 (2001)