2015 IEEE 14th International Conference on Machine Learning and Applications

Restricted Boltzmann machine for nonlinear system modeling

Erick De la Rosa , Wen Yu

***Abstract*— In this paper, we use a deep learning method, restricted Boltzmann machine, for nonlinear system identification. The neural model has deep architecture and is generated by a random search method. The initial weights of this deep neural model are obtained from the restricted Boltzmann machines. To identify nonlinear systems, we propose special unsupervised learning methods with input data. The normal supervised learning is used to train the weights with the output data. The modified algorithm is validated by modeling two benchmark systems.**

I. Introduction

The multi-layer perceptron (MLP) can approximate any nonlinear function to any prescribed accuracy if sufficient hidden neurons are provided [1]. The structure identification is often tackled by trial-and-error approaches like fuzzy grid [2], and data mining methods [3]. In this paper, we increase the hidden layers and decrease the hidden units keeping the model complexity while improving its modeling capacity.

Since the identification error space is unknown, the neural model can be settled down in a local minima easily if the initial weights are not suitable [4]. There are some techniques to overcome the local minima: such as noise-shaping modification [5] and combining nonlinear clustering [6]. However, these methods do not solve the key problem: wrong initial weights. In [7], the initial weights are calculated by the sensitivity ratio analysis and in [8], they are obtained by finding the support vectors of the input data. In this paper, we use deep learning techniques to find the best initial weights.

A deep neural network (DNN) has the same structure as a MLP. There are not methods to find the best structure of DNNs. [9], [10], and [11] propose some effective algorithms to find a suitable structure for DNNs: 1) Grid search, it tests all possible combinations of the structure parameters; 2) Random search, it only use a few combinations by some sampling rules obtaining similar performances than the grid search [11].

The restricted Boltzmann machines (RBM) are unsupervised learning methods. The results of [12] show that the unsupervised pre-training can drive the DNN away from the local minima for pattern recognition. For system identification, can similar results be obtained? To the best of our knowledge, there are not publications that apply deep learning for nonlinear system identification. RBMs cannot be applied to system identification directly, because the input values are non-binary as classification problem [13]. In this paper, we first construct a DNN, then we design unsupervised learning methods with RBMs. Then we use the trained weights as the initial weights of the DNN. Finally, the gradient based supervised learning method is applied. Two benchmark examples are used to show the effectiveness of the DNN.

Erick de la Rosa , Wen Yu are with Departamento de Control Automatico, CINVESTAV-IPN (National Polytechnic Institute), Mexico City, Mexico.

II. Nonlinear system identification with deep neural networks

Consider the following unknown discrete nonlinear system

$$\bar{x}(k+1) = f\left[\bar{x}\left(k\right), u\left(k\right)\right], \quad y(k) = g\left[\bar{x}\left(k\right)\right] \tag{1}$$

where $u\left(k\right) \in \Re^u$ is the input vector, $\bar{x}\left(k\right) \in \Re^x$ is an internal state vector, and $y\left(k\right) \in \Re^m$ is the output vector. f and g are nonlinear smooth functions $f, g \in C^\infty$. Denoting $Y(k) = \left[y^T\left(k\right), y^T\left(k+1\right), \cdots y^T\left(k+n-1\right)\right]^T$, $U(k) = \left[u^T\left(k\right), u^T\left(k+1\right), \cdots u^T\left(k+n-2\right)\right]^T$. If $\frac{\partial Y}{\partial \bar{x}}$ is non-singular at $\bar{x} = 0$, $U = 0$, this leads to the model (2)

$$y(k) = \Phi\left[x\left(k\right)\right] \tag{2}$$

$$x\left(k\right) = [y^T\left(k-1\right), y^T\left(k-2\right), \cdots u^T\left(k\right), u^T\left(k-1\right), \cdots]^T$$

$\Phi\left(\cdot\right)$ is an unknown nonlinear difference equation representing the plant dynamics, $u\left(k\right)$ and $y\left(k\right)$ are measurable scalar input and output, d is time delay. The nonlinear system (2) is a NARMA model. We can also regard the input of the nonlinear system as $x\left(k\right) = \left[x_1 \cdots x_n\right]^T \in \Re^n$, the output as $y(k) \in \Re^m$. We use the MLP to identify the unknown system (2)

$$\widehat{y}\left(k\right) = \phi_p\left\{W_p \phi_{p-1} \ldots \left(W_2 \phi_1\left[W_1 x\left(k\right) + b_1\right] + b_2\right) \ldots + b_p\right\} \tag{3}$$

where $\widehat{y}\left(k\right) \in \Re^m$ is the output of the neural model, $W_1 \in \Re^{l_1 \times n}$, $b_1 \in \Re^{l_1}$, $W_2 \in \Re^{l_2 \times l_1}$, $b_2 \in \Re^{l_2}$, $W_l \in \Re^{m \times l_{p-1}}$, $b_p \in \Re^m$, l_i $(i = 1 \cdots p)$ are node numbers in each layer, $\phi_i \in \Re^{l_i}$ $(i = 1 \cdots p)$ are active vector functions, p is layer number. We use linear function for the output layer ϕ_p, *i.e.*, $\phi_p = \left[\sum_1 \cdots \sum_m\right]$. The other layers use sigmoid functions as $\phi_i\left(\omega\right) = \alpha_i / \left(1 + e^{-\beta_i^T \omega}\right) - \gamma_i$, where $i = 1 \cdots p-1$, $j = 1 \cdots l_i$, α_i, β_i, and γ_i are prior defined positive constants, ω is the input vector to the sigmoid functions.

From the Stone-Weierstrass theorem, we know if the node number of one hidden layer is large enough, the neural model can approximate the nonlinear function Φ to any degree of accuracy for all $x\left(k\right)$. Instead of increasing l_i, we increase the layer number p in this paper. The deep neural structure of this paper is $p \geq 2$. The goal of the neural identification is to find a suitable structure (layer number p, node number in each layer l_i) and weights $(W_1 \cdots W_p)$ such that the identification error $e\left(k\right) = \widehat{y}\left(k\right) - y\left(k\right)$ is minimized. To find a suitable structure of the neural model, we use the random search method [14].

The next job is to find suitable weights $(W_1 \cdots W_p)$. The gradient descent method can always arrive a local minima completely depending on the initial weights $W_1 \cdots W_p$. In the following sections, we will show how to use RBMs to find the initial weights $W_1\left(0\right) \cdots W_p\left(0\right)$, and how to identify the nonlinear systems as suggested in Fig.1.

978-1-5090-0287-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ICMLA.2015.24

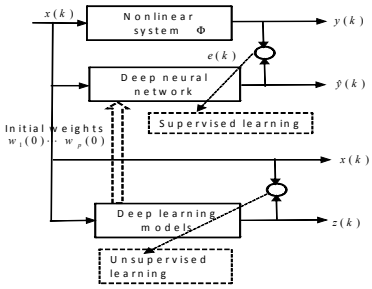

Fig. 1. The identification structure using the deep learning techniques.

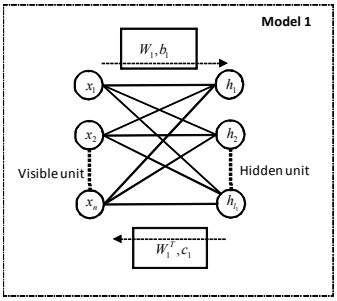

Fig. 2. The model of the restricted Boltzmann machine

III. Restricted Boltzmann machines for system identification

The RBM is a deep learning method, which trains the weights under a probability distribution by using a set of input. We describe next how to use it for system identification.

A. Restricted Boltzmann machine training

An RBM is an energy-based model, which is defined by a probability distribution. This probability distribution depends on the current configuration (or energy) of the model. The goal of the training is to maximize the likelihood of the model,

$$P(x) = \sum_{l} P(x,h) = \sum_{l} \frac{e^{-E(x,h)}}{Z} \tag{4}$$

where x is the input, l is the hidden unit, Z is a partition function defined as $Z = \sum_x e^{-E(x,h)}$, $P(x)$ is the probability distribution of x, $E(x,h)$ is the energy function defined

$$E(x,h) = -b^T x - c^T h - h^T W x \tag{5}$$

where W is the weight matrix of the hidden layer, b and c are visible and hidden bias respectively. Since the observation of all states $x(k)$ is not tractable, this may require the introduction of some not observed variables h to enhance the reconstruction capabilities. The nonlinear active functions become conditional probability transformations. In some simple cases, such as classification and dimensionality reduction, h_i are binary values, *i.e.,* $h_i \in \{0,1\}$. If W_j $(j = 1 \cdots l_i)$ denote the rows of W and W_i^T $(i = 1 \cdots n)$ its columns, the probabilistic version of the deep learning model becomes

$$\begin{aligned} P(h|x) &= \prod_{j=1\cdots l_i} P(h_j|x) = \prod_{j=1\cdots l_i} \phi\left[W_j x(k) + b\right] \\ P(x|h) &= \prod_{i=1\cdots n} P(x_i|h) = \prod_{i=1\cdots n} \phi\left[W_i^T h(k) + c\right] \end{aligned} \tag{6}$$

For each sub-model, the transformation (6) is repeated s times. It is shown in Fig.2, which shows the first model of RBM. The hidden representation of Model 1 is $h_1(k) = \phi\left[W_1 x(k) + b_1\right]$ where $h_1(k)$ is the input of RBM Model 2. The weights of RBM are updated by

$$W(k+1) = W(k) - \eta_3 \frac{\partial \log P(x)}{\partial W(k)} \tag{7}$$

In order to calculate de gradient, the concept of free energy $F(x)$ is introduced by (8).

$$F(x) = -b^T x - \sum_{i=1}^{n} \log \sum_{h_i} e^{h_i(c_i + W_i x)} \tag{8}$$

This allows to calculate de gradient of $P(x)$ as,

$$\frac{\partial \log P(x)}{\partial W(k)} = \sum P(z) \frac{\partial F(z)}{\partial W(k)} - \frac{\partial F(x)}{\partial W(k)}$$

It is difficult to determine an analytical expression for the gradient $\sum P(z) \frac{\partial F(z)}{\partial W(k)}$, because it involves the calculation of $E_P\left[\frac{\partial F(z)}{\partial W(k)}\right]$, which is the mathematical expectation over all possible combinations of the reconstructed input $z(k)$ under the distribution $P(z)$. In this paper, we estimate it with a set S which includes finite samples. The sample number is s, turning the sum into $\sum P(z) \frac{\partial F(z)}{\partial W(k)} \approx \frac{1}{s} \sum_{z \in S} \frac{\partial F(z)}{\partial W(k)} \cdot \sum P(z) \frac{\partial F(z)}{\partial W(k)} \approx \frac{1}{s} \sum_{z \in S} \frac{\partial F(z)}{\partial W(k)}$. Here the samples S are obtained with the Monte Carlo algorithm with contrastive divergence. Monte Carlo Markov chains work especially well for RBM models [15], although the approximation capability is improved when the hidden units increases [16]. In each running period, the samples S and the input $x(k)$ are used to update the weights W and a new example $x(k+1)$ is presented to the model.

B. Conditional probability for non-binary values

For nonlinear system identification, the visible units cannot be binary values. We classify the input $x(k)$ into three types: 1) interval $[0, \infty)$, for unbounded positive input; 2) $[0,1]$, for normalized input; 3) $[-\delta, \delta]$, for bounded input.

1) The visible units are in the interval $[0, \infty)$.

The conditional probability for energy function of (4) is

$$P(x_i|h) = \frac{e^{(c_i + h^T W_i) x_i}}{\int_{x_i} e^{(c_i + h^T W_i) x_i} dx_i} \tag{9}$$

Obviously, if $x_i(k) \in (-\infty, \infty)$, the integral term does not converge. Let denote $a_i = c_i + h^T W_i$ where a_i is the term applied to the visible units, see Fig.2. If $x_i(k)$ are non-negative, $x_i(k) \in [0, \infty)$, (9) becomes

$$P(x_i|h) = \frac{e^{a_i x_i}}{\int_0^\infty e^{a_i x_i} dx_i} = \frac{e^{a_i x_i}}{\frac{1}{a_i} e^{a_i x_i}|_0^\infty} \tag{10}$$

(10) has finite value if $a_i(h) < 0$, $\forall h$. The conditional probability distribution is $P(x_i|h) = -a_i(h)\, e^{a_i(h)x_i} > 0$. The visible units which have this probability distribution as are called exponential units. In order to perform the sampling process, we need to calculate the cumulative probability distribution $P_C(x_i|h) = 1 - e^{a_i x_i}$. So $P_C(x_i|h)$ always increases. The sampling process is possible by the inverse function of cumulative probability P_C^{-1}. If $o(k)$ is a value from a sampling process on a uniform distribution, then we can associate it with the cumulative density value P_C in (11) and the expected value according to the distribution $P(x_i|h)$ is $E[x_i] = -1/a_i(h)$.

$$z_i(k) = \frac{\ln(1 - o(k))}{a_i} \tag{11}$$

2) The visible and hidden units are in the interval $[0, 1]$

The positive range $[0, \infty)$ is very big. Sometimes the calculation accuracy is affective. If the input to the visible units is restricted in a special zone, normalization methods can be applied to fore the interval in $[0, 1]$. In this case, the probability distribution computation is also bounded (9) and the cummulative probability is given by (12).

$$P_C(x_i|h) = \frac{a_i}{e^{a_i} - 1}\int_0^{x_i} e^{a_i x_i} dx_i = \frac{e^{a_i x_i} - 1}{e^{a_i} - 1} \tag{12}$$

With a sample unit $o(k)$, the new value $z_i(k)$ with respect to P_C is given by (13) and the expected value of the distribution is $E[x_i] = 1/(1 - e^{-a_i}) - 1/a_i$

$$z_i(k) = \frac{\log\left[1 + o(k)\left(e^{a_i} - 1\right)\right]}{a_i} \tag{13}$$

3) The visible and hidden units are in the interval $[-\delta, \delta]$

If the input set $x(k)$ can be positive and negative, we define a interval as $[-\delta, \delta]$ for each visible unit. In this case the conditional probability is truncated exponential. (9) is

$$P(x_i|h) = \frac{e^{a_i x_i}}{\frac{1}{a_i} e^{a_i x_i}|_{-\delta}^{\delta}} = \frac{a_i e^{a_i x_i}}{e^{a_i \delta} - e^{-a_i \delta}} \tag{14}$$

The sampling process using the inverse function of the corresponding P_C and $o(k)$ in the uniform distribution and the expected value for this distribution are

$$\begin{aligned} z_i(k) &= \frac{\log\left[e^{-a_i\delta} + o(k)\left(e^{a_i\delta} - e^{-a_i\delta}\right)\right]}{a_i} \\ E[x_i] &= \delta \frac{e^{a_i\delta} + e^{-a_i\delta}}{e^{a_i\delta} - e^{-a_i\delta}} - \frac{1}{a_i} \end{aligned} \tag{15}$$

After RBM training with input data, the initial weights of neural model are obtained. For the supervised learning, the weights W_i are updated by

$$W_i(k+1) = W_i(k) - \eta_2 \frac{\partial J_2(k)}{\partial W_i(k)}, \quad i = 1 \cdots p \tag{16}$$

IV. Simulations

In this section, we use two benchmark examples proposed in [17] to show the effectiveness of our deep learning methods for nonlinear system identification.

A. First order nonlinear system

The first order nonlinear system presented by [17] is

$$y(k+1) = \frac{y(k)}{1 + y^2(k)} + u^3(k) \tag{17}$$

where $u(k)$ is a periodic input, which has different form in the training and the testing processes

$$u(k) = A \sin\left(\frac{\pi k}{50}\right) + B \sin\left(\frac{\pi k}{20}\right) \tag{18}$$

In the training stage, $A = B = 1$. In the training stage of the neural model (3), $A = 1.1$, $B = 0.9$. In the testing stage, $A = 0.9$, $B = 1.1$. The unknown nonlinear system (17) has the form of (2). $\bar{x}(k) = [y(k), u(k)]^T$, $n = 2$, $m = 1$.

We use 2000 data to train the deep learning model. We use 2000 data for the neural model (3), 200 data to test it. So $q = 2000$. The structure parameters of the neural model, layer number p and node number of each layer l_i ($i = 1 \cdots p$), are obtained by the random search method [11]. The search ranges of the parameters are: $10 \geq p \geq 3$, $6 \geq l_i \geq 2$, for the RBM (7) $0.1 \geq \eta_3 \geq 0.01$. The random search shows better performance with 4 hidden layers ($p = 5$) and $l_i = 6$ ($i = 1, 2, 3, 4$). We use it as our neural model: four hidden layers, each hidden layer has six nodes.

We compare the following four models: deep neural model with RBM (DN_RB), deep neural model with autoencoders method [18] (DN_AM), multilayer perceptron (MLP), and deep neural model with backpropagation (DN_BP). The MLP model has the same structure as [17], it has two hidden layers $(20, 10)$. The DN_BP and DN_AM models have the same structure as the deep neural model $(6, 6, 6, 6)$. However, the initial weights of DN_BP are random.

The DN_RB model uses four types of visible units to calculate the conditional probability transformation: binary, $[0, \infty)$ interval, $[0, 1]$ interval, and $[-8, 8]$ interval. For binary case $\bar{x}(k)$ is treated as a probability not as a real value. For the interval $[0, 1]$, $\bar{x}(k)$ is normalized as

$$x(k) = \frac{\bar{x}(k) - \min_k\{\bar{x}(k)\}}{\max\{\bar{x}(k)\} - \min_k\{\bar{x}(k)\}} \tag{19}$$

When $q = 2000$, the upper bound of $\bar{x}(k)$ is 8. For $[0, \infty)$ case, $x(k) = \bar{x}(k) + 8$.

The testing results of the four models are shown in Fig. 3.

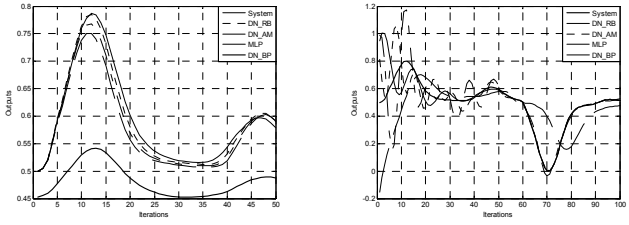

Fig. 3 First order system results. Left) Training Right) Testing

In the testing phase, we define the average error as $\frac{1}{N}\sum_{k=1}^{N} \|\hat{y}(k) - y(k)\|^2$. In this example, $N = 200$. Table

1 shows the average errors of different models in the testing phase. This table also gives the comparisons of the four types visible units for RBM. We can see that the best performance is to normalize the visible units in $[0, 1]$. It is some strange that the binary normalization has better performance than the non-normalization methods, $[0, \infty)$ and $[-8, 8]$. We believe that the normalization method is more suitable for the conditional probability transformation of RBM, and the weights are not influenced so much by exact values of the visible units.

For this example, we have the following conclusions:

1) The deep neural model of this paper has better modeling accuracy than the normal neural model as in [17]. The deep neural model has less hidden nodes (24) than MLP (30).

2) The RBM (DN_RB) has better capacity to avoid the local minima than the autoencoders method (DN_AM)

3) Comparing DN_RB and DN_AM with DN_BP, the deep learning techniques are effective in nonlinear modeling

4) After the RBM application, the weighs converge faster.

B. Second order nonlinear system

The second order nonlinear system presented by [17] is

$$y(k+1) = \frac{y(k) + y(k-1)}{1 + y^2(k) + y^2(k-1)} + u(k)^3 - u(k) \quad (20)$$

where $u(k)$ is a periodic input as described by (18). In the training stage of the DNN (3), $A = 0.8$, $B = 1.1$. In the testing stage, $A = 0.9$, $B = 0.8$. For the unknown nonlinear system (20) $\bar{x}(k) = [y(k), y(k-1), u(k)]^T$, $n = 3$, $m = 1$. We use 2000 data to train the model and 400 data to test it. The search ranges of the parameters are the same as the first order nonlinear system (17). A local minimum is in four hidden layers ($p = 5$) and $l_i = 3$.

We also use four models, DN_RB, DN_AM, MLP, and DN_BP to identify the nonlinear system (20). The visible units of DN_RB have also the same 4 types and the MLP model has two hidden layers $(20, 10)$. The training time of the deep learning models DN_RB and DN_AM are 11 seconds and 6 seconds with the Intel i5 CPU @ 2.4GHZ. The DN_RB model has more computational complexity. The training results of the deep neural model (3) are shown in Fig. 4 (left). The deep learning methods achieve good performances after 60 training samples without the presence of overfitting.

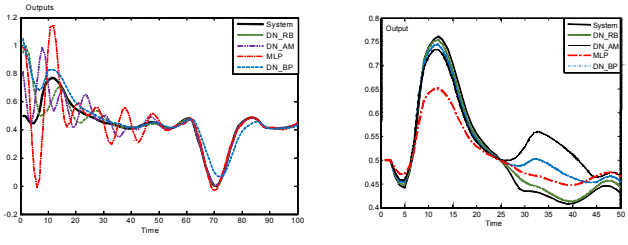

Fig. 4 Second order system results. Left) Training Right) Testing

Fig. 4 (right) and Table 1 show the testing results of different models. We obtain the similar conclusions as the first example.

Table 1. The average error in the testing phase ($\times 10^{-3}$). FOS (first order system), SOS (second order system)

Model	DN_AM	MLP	DN_BP	Binary	$[0,\infty)$	[0,1)	[-8,8]
FOS	7.477	16.158	6.652	5.972	31.546	5.915	6.862
SOS	8.569	9.123	21.345	7.738	14.124	7.234	9.456

V. Conclusions

In this paper, the neural model for nonlinear system identification has deep structure. The unsupervised learning methods with input data is used to obtain the initial weights of the neural model. The supervised learning techniques with output data are applied to train the weights. We modify the deep learning algorithms of the restricted Boltzmann machines, such that they are suitable for nonlinear system identification. Further works will develop on-line version of these algorithms.

References

[1] G.Cybenko, Approximation by Superposition of Sigmoidal Activation Function, *Math.Control, Sig Syst*, Vol.2, 303-314, 1989

[2] J. S. Jang, ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 23, 665–685, 1993.

[3] W.Yu, K.Li, X.Li, Automated nonlinear system modeling with multiple neural networks, *International Journal of Systems Science*, Vol.42, No.10,1683-1695, 2011

[4] S.Jagannathan and F.L.Lewis, Identification of Nonlinear Dynamical Systems Using Multilayered Neural Networks, *Automatica*, Vol.32, No.12, 1707-1712, 1996.

[5] S.Chakrabartty, ; R.K.Shaga, K.Aono, Noise-Shaping Gradient Descent-Based Online Adaptation Algorithms for Digital Calibration of Analog Circuits, *IEEE Transactions on Neural Networks and Learning Systems*, Volume: 24 , Issue: 4, pp. 554-565, 2013

[6] Yang Liu, Yan Liu, K.Chan, K.A.Hua, Hybrid Manifold Embedding, *IEEE Transactions on Neural Networks and Learning Systems*, Volume: 25 , Issue: 12, pp.2295 - 2302, 2014

[7] Q.Song, Robust Initialization of a Jordan Network With Recurrent Constrained Learning, *IEEE Transactions on Neural Networks*, Vol.22, No.12, pp.2460 - 2473, 2011

[8] W.Yu, X.Li, Automated Nonlinear System Modeling with Multiple Fuzzy Neural Networks and Kernel Smoothing, *International Journal of Neural Systems*, Vol.20, No.5, 429-435, 2010

[9] J. Bergstra, Y. Bengio, Algorithms for Hyper-Parameter Optimization, *Journal of Machine Learning Research*, pp 201-210. 2012

[10] S. Lawrence, C. Lee Giles, and Ah Chung, What Size Neural Network Gives Optimal Generalization, *Technical Report, University of Maryland*, 1996

[11] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization, *Journal of Machine Learning Research*, pp 281-305, 2011

[12] D.Erhan, Y.Bengio, A.Courville, P-A.Manzagol, P.Vincent, Why Does Unsupervised Pre-training Help Deep Learning?, *Journal of Machine Learning Research, vol.*11, 625-660, 2010

[13] G. E. Hinton, S. Osindero, and Y. Teh, A fast learning algorithm for deep belief nets, *Neural Computation*, vol. 18, pp. 1527-1554, 2006.

[14] R. Collobert and J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, *25th International Conference on Machine Learning*, pp. 160-167, ACM, 2008.

[15] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, A learning algorithm for boltzmann machines, *Cognitive Science*, vol. 9, pp. 147-169, 1985.

[16] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, The difficulty of training deep architectures and the effect of unsupervised pretraining, *12th International Conference on Artificial Intelligence and Statistics (AISTATS'09)*, pp. 153-160, 2009.

[17] K. S. Narendra and K. Parthasarathy, Gradient methods for optimization of dynamical systems containing neural networks, *IEEE Transactions od Neural Networks*, pp. 252-262, March 1991

[18] P.Vincent, H. Larochelle Y. Bengio and P.A. Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders, *25th International Conference on Machine Learning*, pp.1096-1103, ACM, 2008.