A Deep and Stable Extreme Learning Approach for Classification and Regression*

Le-le Cao, Wen-bing Huang, and Fu-chun Sun

Tsinghua National Laboratory for Information Science and Technology (TNList),
Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, P.R. China
{caoll12,huangwb12}@mails.tsinghua.edu.cn,
fcsun@mail.tsinghua.edu.cn

Abstract. The random-hidden-node based extreme learning machine (ELM) is a much more generalized cluster of single-hidden-layer feed-forward neural networks (SLFNs) whose hidden layer do not need to be adjusted, and tends to reach both the smallest training error and the smallest norm of output weights. Deep belief networks (DBNs) are probabilistic generative modals composed of simple, unsupervised networks such as restricted Boltzmann machines (RBMs) or auto-encoders, where each sub-network's hidden layer serves as the visible layer for the next. This paper proposes an approach: DS-ELM (a deep and stable extreme learning machine) that combines a DBN with an ELM. The performance analysis on real-world classification (binary and multi-category) and regression problems shows that DS-ELM tends to achieve a better performance on relatively large datasets (large sample size and high dimension). In most tested cases, DS-ELM's performance is generally more stable than ELM and DBN in solving classification problems. Moreover, the training time consumption of DS-ELM is comparable to ELM.

Keywords: extreme learning machine (ELM), deep belief networks (DBNs), classification, regression, deep-and-stable ELM.

1 Introduction

In the research field of machine learning, the capability of classification and regression is often evaluated from perspectives such as accuracy, time cost, stability, and statistical significance. The research introduced in this paper will focus on the performance of learning machines with respect to accuracy and stability of both classification and regression tasks. In particular, the neural networks approaches are our major focus. On one hand, Extreme Learning Machine (ELM) [1–3] is within our scope because of its simple architecture with proven potential in solving classification and regression problems [1, 4]; on the other hand, we emphasize deep neural networks, specifically on deep belief networks

* This work was supported by grants from China National Natural Science Foundation under Project 613278050 and 61210013.

(DBNs) composed of several layers of restricted Boltzmann machines (RBMs), which "seek to learn concepts instead of recognizing objects" [5].

Both ELM and DBNs have gained widespread popularity these years; and many sound and successful applications built upon ELM and DBNs have been reported. Generally speaking, ELM has high scalability and less computational complexity, while DBNs are known to have good modeling ability for higher-order and highly non-linear statistical structure in the input [6]. It is commonly accepted that the first layers of DBNs are expected to extract relatively low-level features out of the input space while the upper layers are expected to gradually refine previously learnt concepts to generate more abstract ones. Hence, it is natural to think of the possibility of combining ELM and DBNs, so we can have advantages from both methodologies in one.

Because the output of the higher DBN layers can easily be used as the input of a supervised classifier, Ribeiro et al. [5] used an ELM classifier for classify the deep concepts and lower the training cost of DBNs by applying adaptive learning rate technique and Graphics Processing Units (GPU) implementation of DBNs [7]. The recognition rate of their proposed approach (named DBN-ELM) is competitive (and often better) than other successful approaches in well-known benchmarks. The so called DBN-ELM approach is making use of full unsupervised learning power of DBN (as an auto encoder), the output of which is fed into a typical ELM classifier with randomly generated hidden neurons. This method tends to consume noticeable more time than a standalone ELM classifier; and the performance is not stable, meaning the test accuracy of a single trial (given exactly the same training and testing split) might very well likely be different from others trials. Although this kind of performance fluctuation is acceptable and limited within a certain interval, this phenomenon makes it mandatory to carry out multiple trials and perform statistical significance analysis.

The authors of [8] proposed a method called multilayer ELM (ML-ELM) using ELM as an auto encoder for learning feature representations. ML-ELM is composed of ELM auto encoders (ELM-AE) which performs layer-by-layer unsupervised learning like a typical DBN. ELM-AE can be regarded as a special case of ELM, where the input is equal to output. In a nutshell, the proposed method [8] initialize ELM-AE hidden layer weights within a deep structure in a random manner, adjust the weights using layer-wise unsupervised training, and finally fine-tune the entire network with BP algorithm. Although the reported testing accuracy of ML-ELM outperforms DBNs and ELMs, the problem of performance fluctuation still exists as in DBN-ELM [5].

In short, DBN-ELM [5] concatenates a DBN with an ELM (on top layer as a supervised classifier), while ML-ELM [8] stacks multiple ELM-AE together to form a deep network. Because of the random hidden nodes in both approaches, the performance (testing accuracy) is not stable. One of our objectives is to propose a new way of assembling DBN and ELM together, obtaining an ELM learning machine with deep structure, which is expected to have a relatively stable performance. DBN-ELM was only tested towards image reconstruction and classification tasks in [5]. ML-ELM was tested merely on MINIST data set which is commonly used for testing deep network performance. The other objective of our research is testing our approach on classification (binary and multiple class) and regression datasets to obtain a full picture of its performance.

2 Extreme Learning Machine

Extreme learning machine (ELM) [1–3] was developed specifically for single-hidden-layer feed-forward neural networks (SLFNs) at the very beginning. Huang et al. then "generalized" it to a kind of SLFN which may not be neuron alike [9, 10]. ELM was also extended to kernel learning in [4] showing that ELM can make use of various feature mappings such as random hidden nodes and kernels. ELM tends to reach both the smallest training error and the smallest norm of output weights [1, 4, 11]. It has been proved in [4] that ELM can achieve fast learning speed and good generalization performance on both regression and classification tasks. ELM is fast, which may be attributed to its single-hidden-layer structure requiring no iterative process. ELM also requires less human intervention and supervision, which makes it an efficient algorithm especially when facing large datasets where training time and easy parameter tuning are critical.

The decision function of ELM for generalized SLFNs is shown in the following equation. For simplicity, we take the case of one output node as an example.

$$f_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x) = \beta \cdot h(x) \tag{1}$$

where L denotes the number of hidden-layer node; β_i represents the weight connecting the i-th hidden node and the output; notation $G(a_i, b_i, x)$ is the activation function of the i-th hidden node; $h(x) = [G(a_1, b_1, x), \cdots, G(a_L, b_L, x)]^T$ is the output vector of the hidden layer with respect to the input x [11]. $h(x)$ maps the feature dimension(s) from N to L. It is worth mentioning that parameters for hidden node (i.e. $\{a_i, b_i\}_{i=1\cdots L}$) can be randomly generated obeying any continuous probability distribution [4, 11]. As a result, ELM could generate the hidden node parameters before seeing the training data. As long as the output functions of hidden neurons are nonlinear piecewise continuous, neural networks with random hidden neurons attain both universal approximation and classification capabilities, and the changes in finite number of hidden neurons and their related connections do not affect the overall performance of the networks. Equation (1) is equivalent to $H\beta = T$, where

$$H = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \cdots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix}, \beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}, T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix} \tag{2}$$

As a result, the hidden layer output matrix H is also called ELM feature space [4] mapped from input layer to hidden layer. The i-th column of H is the output of the i-th hidden node with respect to inputs $x_1, x_2, \cdots, x_N$. Given a training set $\aleph = \{(x_i, t_i) | x_i \in R^n, t_i \in R^m, i = 1, \cdots, N\}$, hidden node output function $G(a, b, x)$, and the number of hidden nodes L, the output weight β can be calculated by equation (3) [4]:

$$\beta = H^{\dagger}T = \begin{cases} H^T(\frac{I}{C} + HH^T)^{-1}T, & \text{when training set is not huge} \\ (\frac{I}{C} + H^T H)^{-1}H^T T, & \text{when training set is huge} \end{cases} \tag{3}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of hidden layer output matrix H [1]. The positive value $\frac{I}{C}$ is added to the diagonal of $H^T H$ or HH^T to make the resulting solution stabler and obtain better generalization performance [4, 12]. Unlike traditional gradient-based learning algorithms facing several issues like local minima, improper learning rate and overfitting, etc, ELM tends to reach the solutions straightforward without such trivial issues [13].

3 Deep Belief Networks

DBNs are probabilistic generative modals, or alternatively a kind of deep neural network, composed of multiple latent variables (hidden units). DBNs were initially introduced in [14], addressing three problems that exist in traditional deeply layered neural networks: (1) large demand for training examples; (2) time consuming to reach convergence; (3) prone to local optima [15]. Recent key findings on neocortex of mammal brain such as [16] motivated the emergence of deep learning machine even further. Many researchers have affiliated the fact that DBNs can reach an equivalent modeling capability as SLFNs using a lot less nodes in each hidden layer.

DBNs can be viewed as a composition of simple, unsupervised networks such as restricted Boltzmann machines (RBMs) [14] or autoencoders [17], where each sub-network's hidden layer serves as the visible layer for the next. As is illustrated in Fig. 1 (a), each layer tries to model the distribution of its input. Every RBM has a layer containing visible nodes v that represent the data and a layer containing hidden nodes h that learn to represent input features capturing higher-order correlations in the data. [5] The topology of DBNs depicts a joint distribution based on observation input v and multiple hidden units $h_1, h_2, \cdots, h_L$:

$$P(v, h_1, h_2, \cdots, h_L) = (\prod_{k=0}^{L-2} P(h_k | h_{k+1}))P(h_{L-1}, h_L) \tag{4}$$

The key idea behind DBNs is that the weights, W, connecting two layers have no connections within a layer. This matrix of symmetrically weighted connections is learned by an RBM which defines both $p(v|h, W)$ and the prior distribution over hidden vectors, $p(h|W)$, so the probability of generating the visible vector, v, can be written as:

$$p(v) = \sum_h p(h|W)p(v|h, W) \tag{5}$$

By starting with the data vector on the visible units and alternating several times between sampling from $p(h|v, W)$ and $p(v|h, W)$, it is easy to get the learning weights W. The learning algorithm for DBNs proposed by Hinton et al. [14, 18] has two training phases: (1) a greedy learning algorithm for transforming representations (unsupervised learning), and (2) Back-Fitting with the up-down algorithm (fine-tune). The term "epochs" is used to represent iterations or sweeps of unsupervised pre-training (per layer) and supervised fine-tune.

4 Proposed Approach

This section presents a new machine learning approach named deep and stable extreme learning machine (DS-ELM) inspired by the ELM and DBN methodology. Our overall intention is to use a quick-and-dirty DBN to generate a relatively stable feature space H that is fed into an ELM to calculate the output weights. The details of DS-ELM approach are explained in two steps below (Fig. 1):

Step 1. Setup a DBN structure fed with input vector x; and perform a quick-and-dirty training based on the pre-defined DBN structure [cf. Fig. 1(a)]. The term "quick-and-dirty" means that both "unsupervised pre-training" and "supervised fine-tune" are accomplished within only a few iterations rather than sufficient iterations. Because the separating hyper-plane of ELM feature space goes through the origin in theory [4, 11]; hence bias b is not needed in training this quick-and-dirty DBN.

Step 2. The nodes in the top hidden layer can be viewed equal to hidden nodes in a typical ELM network; those hidden layer output matrix H is feature space of the input vector. the feature space H initiated via Step 1 with help of a DBN is then fed into a typical ELM solver to calculate the output weights β with equation (3). [cf. Fig. 1(b)]

By integrating a DBN with an ELM in this manner, DS-ELM adds the following potential advantages to a standalone ELM:

- **Auto-abstraction of deep concepts.** Most of the classification and regression problems have input examples which are usually represented by a set of manually extracted features. In many cases, the challenging nature of many problems lie on the difficulty of extracting features such as "behavioral characteristics like mood, fatigue, energy, etc." [5] A typical example is object image classification problem which is especially challenging due to the fact that same object might appear differently because of pose and illumination conditions; the low level visual features are far detached from the semantics of the scene, making it problem-prone when used to infer object presence. [19] Human-crafted features [20] is very hard to embody complex functions hidden in input data, but the unsupervised pre-training of DBNs allows learning those complex functions by mapping the input to the output directly. Specifically speaking, the bottom layers are expected to extract and represent low-level features from the input data while the upper layers are expected to gradually refine previously learnt concepts [5].

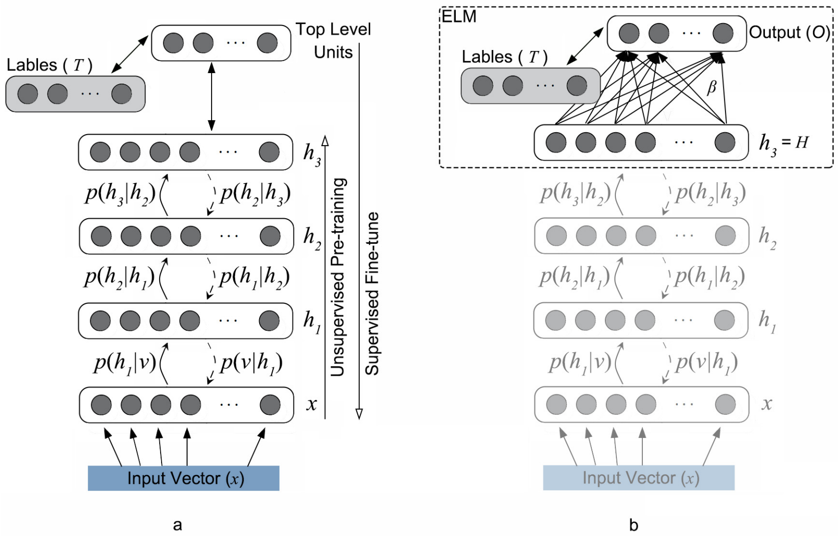

Fig. 1. An example (three hidden layers) of two-step training process of DS-ELM. (a) Step 1: initialize ELM feature space H with a quick-and-dirty DBN. (b) Step 2: calculate output weights β from H with a typical ELM solver.

- **Stable feature space and performance.** The parameters for ELM hidden node (i.e. $\{a_i, b_i\}_{i=1\cdots L}$) are randomly generated obeying any continuous probability distribution [4, 11]. Hence the ELM feature space H (or called feature mapping matrix) defined in equation (2) does not stays the same even for the same input data. According to ELM universal approximation capability (of approximating any target continuous function) theorems [9, 3], we can prove the classification capability of ELM [4]; but the performance of ELM with random hidden nodes is not quite stable. The most straightforward impact is on the test accuracy of a single trial (given exactly the same training and testing split) which might very likely be different from others trials. Although this kind of performance fluctuation is acceptable and controlled within certain limits, this behavior makes it necessary to carry out groups of trials and perform statistical significance analysis. DS-ELM, on the other hand, stabilize the ELM performance (test accuracy) by initialize ELM feature space H with a quick-and-dirty DBN.

5 Experiments and Analysis

In order to extensively verify the performance of DS-ELM, a variety type of real-world data was chosen for each problem category (regression, binary classification, and multi-category classification). Seen from Table 1, the simulations involve 17 datasets ranging from small to large data size; and from low to high dimension. Fixed training/testing division is applied for all datasets.

Most of the simulated experiments of DS-ELM, ELM[1], and DBN[2] are carried out with Matlab 2012b (maci64) run on Intel Core i7, 2.3-GHz CPU with 16-GB, 1333-MHz RAM and 250-GB, SATA 6GB/s SSD. A few datasets (i.e. Shuttle, CTslice [21], and Protein [22]) require larger memory, so we have to execute algorithms for these datasets with a Matlab 2013a installation on a high-performance server with dual Xeon E7-4820 2.266GHz CPU and 4x64G RAM.

Table 1. Dimension and size of selected benchmark data sets: binary classification problems (noted as "bincls"), multiple-category classification problems (noted as "multicls"), and regression problems (noted as "reg")

Size / Dim.	Small	Large
Low	**bincls**: Diabetes, Liver [21] **multicls**: Iris, Segment [21], Vowel [23] **reg**: Pyrim, Housing [21]	**bincls**: Mushroom, Musk2 [21] **multicls**: Shuttle [21] **reg**: Abalone [21]
High	**bincls**: Leukemia [24] **multicls**: DNA [21] **reg**: Crime [21, 26]	**bincls**: Gisette [25] **multicls**: Protein [22] **reg**: CTslice [21]

The code of ELM classifier is originally obtained from [4]. The source code is then adjusted to fit in the needs of data pre-processing and feature preparation. The toolbox containing DBN implementation is retrieved in accordance with [27]. Our implementation of DS-ELM approach is a combination of ELM [4] and DBN [27] based on the procedure explained in section 4. In all simulations, Sigmoidal hidden layer activation function is used, and 20 trials are executed for each dataset.

The experimental results for classification problems and regression problems are put together in Table 2 and 3 respectively. The best results among the three tested approaches are highlighted in bold. Generally speaking, the three approaches are capable of achieving similar generation performance for most tested datasets.

Table 2 shows the performance comparison of ELM, DBN, and DS-ELM for classification problems. It can be seen from binary classification tests that (1) DS-ELM tends to obtain the lowest standard deviation for five out of six datasets; (2) although ELM achieved best testing rate for four out of six datasets, DS-ELM has the best testing rate for Mushroom and Gisette datasets which contain the biggest number of training/testing samples. Observed from multi-category classification simulations, we found that (1) the performance of DS-ELM is more stable (with the smallest "Dev" value) than the other two approaches in all tested datasets; (2) for DNA (high dimension and medium size) and Protein (high dimension and large size) datasets, DS-ELM method achieved better testing rate compared to the other two methods.

[1] ELM: `http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html`

[2] DeepLearnToolbox: `https://github.com/rasmusbergpalm/DeepLearnToolbox`

Table 2. Performance comparison of ELM (random hidden nodes), DBN, and DS-ELM approaches: classification problems

Datasets	ELM			DBN			DS-ELM		
	Tesing Rate(%)	Training Time(s)	Dev (%)	Tesing Rate(%)	Training Time(s)	Dev (%)	Tesing Rate(%)	Training Time(s)	Dev (%)
Diabetes	**76.85**	0.14	2.16	73.33	1.2436	6.83	75.88	0.369	**1.07**
Liver	**71.12**	0.189	3.28	68.06	1.739	5.42	70.1	0.2501	**1.83**
Mushroom	99.82	2.23	0.28	88.83	214.34	1.69	**99.9**	2.6515	**0.03**
Musk2	**95.34**	27.231	0.32	85.08	57.103	**0**	93.81	29.281	0.16
Leukemia	**81.25**	3.56	3.97	79.81	32.192	4.53	80.78	3.166	**1.95**
Gisette	92.5	98.199	1.12	90.6	3041.5	2.49	**93.04**	127.08	**0.32**
Iris	**100**	0.0031	**0**	97.91	0.1969	2.36	**100**	0.0093	**0**
Vowel	**90.74**	0.018	1.89	88.45	0.2093	1.9	89.7	0.0602	**0.24**
Segment	**98.07**	0.84	0.94	96.52	139.26	0.92	97.88	2.69	**0.91**
Shuttle	**99.75**	2.8303	0.03	98.99	1894.9	0.02	99.7	9.592	**0.01**
DNA	92.86	1.495	0.4	92.79	2207.8	0.36	**93.61**	13.119	**0.21**
Protein	83.13	20.26	0.65	84.03	1511.3	0.75	**84.78**	30.81	**0.2**

Table 3. Performance comparison of ELM (random hidden nodes), DBN, and DS-ELM approaches: regression problems

Datasets	ELM			DBN			DS-ELM		
	RMSE	Training Time(s)	Dev (%)	RMSE	Training Time(s)	Dev (%)	RMSE	Training Time(s)	Dev (%)
Pyrim	0.1317	0.089	**0.0332**	0.2023	1.592	0.2916	**0.1091**	0.159	0.0346
Housing	**0.0802**	0.239	**0.0103**	0.0838	11.042	0.0599	0.084	0.5753	0.0164
Abalone	**0.073**	1.4166	**0.0031**	0.0785	87.08	0.0438	0.0775	1.7391	0.0099
Crime	0.1381	0.719	0.0642	0.1418	116.37	0.1094	**0.1368**	1.7729	**0.06**
CTslice	3.3055	232.28	0.0125	3.8021	2092.8	0.2665	**3.278**	562.31	**0.0083**

For regression problems (Table 3), DS-ELM also showed a slight advantage over ELM on relatively large datasets (i.e. Crime and CTslice). However, we do not observe an obvious pattern of better standard deviation. It should also be noted that parameter tuning for DBN is a time-consuming task comparing to ELM; it is probably one of the reasons that DBN test results seems sub-optimal than the other approaches. In order to achieve good results, DBN need more careful and data-centric parameter tuning activities. All simulation results showed that ELM uses least training time while DBN tends to consume a lot more time (over hundreds of times more in many situations) to train. DS-ELM approach often has a comparable training time consumption to ELM; of course, the scale of its training time depends on the level of "quick-and-dirty"-ness (number of epochs) of the embodied deep network. Our parameter tuning activities also justify the fact that DS-ELM is not sensitive to the network structure; but it is not quantitatively measured yet in our research.

6 Conclusions

DS-ELM is a training schema that combines DBN and ELM. The essence of DS-ELM is using a quick-and-dirty DBN to generate a relatively stable feature space H which is, in turn, fed into an ELM to calculate the output weights. From our experimental results, we summarize the three key findings below:

(1). DS-ELM tends to achieve better testing rate over ELM and DBN on relatively large datasets (i.e. large number of samples and high dimension).
(2). DS-ELM is generally more stable (with smaller standard deviation value) than ELM and DBN in solving classification problems (for both binary and multiple category cases).
(3). DS-ELM approach often has a similar training-time cost as ELM, as long as its embodied deep network merely requires a few (usually one or two) epochs for unsupervised pre-training (per layer) and supervised fine-tune.

References

1. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: theory and applications. Neurocomputing 70(1), 489–501 (2006)
2. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedingsof the 2004 IEEE International Joint Conference on Neural Networks, vol. 2, pp. 985–990. IEEE (2004)
3. Huang, G.-B., Chen, L., Siew, C.-K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions on Neural Networks 17(4), 879–892 (2006)
4. Huang, G.-B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42(2), 513–529 (2012)
5. Ribeiro, B., Lopes, N.: Extreme Learning Classifier with Deep Concepts. In: Ruiz-Shulcloper, J., Sanniti di Baja, G. (eds.) CIARP 2013, Part I. LNCS, vol. 8258, pp. 182–189. Springer, Heidelberg (2013)
6. A.-r. Mohamed, G., Hinton, G.: Understanding how deep belief networks perform acoustic modelling. In: 2012 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP), pp. 4273–4276. IEEE (2012)
7. Lopes, N., Ribeiro, B.: Gpumlib: An efficient open-source gpu machine learning library. International Journal of Computer Information Systems and Industrial Management Applications 3, 355–362 (2011)
8. Kasun, L.L.C., Zhou, H., Huang, G.-B., Vong, C.M.: Representational learning with extreme learning machine for big data. IEEE Intelligent Systems (2013)
9. Huang, G.-B., Chen, L.: Convex incremental extreme earning machine. Neurocomputing 70(16), 3056–3062 (2007)
10. Huang, G.-B., Chen, L.: Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16), 3460–3468 (2008)
11. Huang, G.-B., Ding, X., Zhou, H.: Optimization method based extreme learning machine for classification. Neurocomputing 74(1), 155–163 (2010)
12. Huang, G.-B., Wang, D.H., Lan, Y.: Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics 2(2), 107–122 (2011)

13. Li, M.-B., Huang, G.-B., Saratchandran, P., Sundararajan, N.: Fully complex extreme learning machine. Neurocomputing 68, 306–314 (2005)
14. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural computation 18(7), 1527–1554 (2006)
15. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning-a new frontier in artificial intelligence research [research frontier]. IEEE Computational Intelligence Magazine 5(4), 13–18 (2010)
16. Lee, T.S., Mumford, D.: Hierarchical bayesian inference in the visual cortex. JOSA A 20(7), 1434–1448 (2003)
17. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al.: Greedy layer-wise training of deep networks. Greedy layer-wise training of deep networks 19, 153 (2007)
18. Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum 9(1), 926 (2010)
19. Wang, G., Hoiem, D., Forsyth, D.: Building text features for object image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 1367–1374. IEEE (2009)
20. Bengio, Y.: Learning deep architectures for AI. Foundations and trends in Machine Learning 2(1), 1–127 (2009)
21. Bache, K., Lichman, M.: UCI repository of machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2013), `http://archive.ics.uci.edu/ml`
22. Shevade, S.K., Keerthi, S.: A simple and efficient algorithm for gene selection using sparse logistic regression. Bioinformatics 19(17), 2246–2253 (2003)
23. Duarte, M.F., Hen Hu, Y.: Vehicle classification in distributed sensor networks. Journal of Parallel and Distributed Computing 64(7), 826–838 (2004)
24. Xing, E.P., Jordan, M.I., Karp, R.M., et al.: Feature selection for high-dimensional genomic microarray data. ICML 1, 601–608 (2001)
25. Guyon, I., Gunn, S.R., Ben-Hur, A., Dror, G.: Result analysis of the nips 2003 feature selection challenge. In: NIPS, vol. 4, pp. 545–552 (2004)
26. Redmond, M., Baveja, A.: A data-driven software tool for enabling cooperative information sharing among police departments. European Journal of Operational Research 141(3), 660–678 (2002)
27. Palm, R.B.: Prediction as a candidate for learning deep hierarchical models of data. Technical University of Denmark, Palm (2012)