A Survey of Randomized Algorithms for Training Neural Networks

Le Zhang, P.N. Suganthan*

School of Electric and Electronic Engineering,
NanYang Technological University,
Singapore, 639798; Email: epnsugan@ntu.edu.sg

Abstract

As a powerful tool for data regression and classification, neural networks have received considerable attention from researchers in fields such as machine learning, statistics, computer vision and so on. There exists a large body of research work on network training, among which most of them tune the parameters iteratively. Such methods often suffer from local minima and slow convergence. It has been shown that randomization based training methods can significantly boost the performance or efficiency of neural networks. Among these methods, most approaches use randomization either to change the data distributions, and/or to fix a part of the parameters or network configurations. This article presents a comprehensive survey of the earliest work and recent advances as well as some suggestions for future research.

Keywords: randomized neural networks, recurrent neural networks, convolutional neural networks, deep learning

1. Introduction

Inspired by biological Neural network, artificial neural network (ANN) are a family of non-parametric learning methods for estimating or approximating functions that may depend on a large number of inputs and outputs. Typically, training protocol of an ANN is based on minimizing a loss function defined on the desired output of the data and actual output of the ANN through updating the parameters. Classical approaches usually tune the parameters based on the derivatives of the loss function. However, much of the power of ANN comes from the nonlinear function in the hidden units used to model the nonlinear mapping between the input and output. Unfortunately, this kind of architecture loses the elegance of finding the global minimum solution with respect to all the parameters of the network since the loss function depends on the output of nonlinear neurons. Thus, the optimization turns out to be nonlinear least square problem which is usually solved iteratively. In this case, the error function has to be back propagated backwards to serve as a guidance for tuning the parameters [30]. Due to this, it is widely acknowledged that these training methods are very slow [38] and may not converge to a single global minimum because there exist many local minima [53, 29] and also the resulting neural network is very weak in the real world noisy situations. These weaknesses of this family of methods naturally limit the applicability of gradient-based algorithms for training neural networks. Randomization based methods remedy this problem by either randomly fixing the network configurations (such as the connections) or some parts of the network parameters (while optimizing the rest by a closed form solution or an iterative procedure), or randomly corrupt the input data or the parameters during the training. Remarkable results have been achieved in various network structures, such as single hidden layer feed forward network [68], RBF neural networks [9], deep neural network with multiple hidden layers [31], convolutional neural network [43] and so on.

A main goal of the paper is to show a role and a place of randomized methods in optimization based neural networks' learning. In Section 2, we present some early work on this line of research on perceptron and standard feed-forward neural network with random parameters in the hidden neuron. Another piece of important work is Random Vector Functional Link Network, which is described in Section 3. Randomization based learning in RBF, recurrent neural network and deep neural network are presented in Section 4, Section 5, and Section 6, respectively. We also offer some details on other scenarios such as evolutionary learning in Section 7. In Section 8, we point out some research gap in the literature of randomization algorithm for neural network training. Conclusions are presented in the last section.

2. Early works on Perceptron and standard Feed-forward Neural Network with randomization

The earliest attempt in this research area was the “perceptron” presented in [64] and extended in [65, 10]. Generally speaking, a perceptron consists of a retina of sensor units, associator and response units. The sensor units are connected to the associator units in “random and many to many” manner. The associator units may connect to other associator units and/or response units. When a stimulus (or the input data) is presented to the sensor units, impulses are conducted from the activated sensor units to the associator units. The associator units are activated once the total arrived signals exceed a threshold. In this case, an impulse from the associator will be send to the units which are connected with it. In perceptron, the weights between the sensor units and the response units can be regarded as randomly selected from $\{1, 0\}$, while the weights between the associator units and the response units are achieved by reinforcement learning.

In [68], the authors investigated the performance of a standard feed-forward neural network (SLFN) which is demonstrated in Fig. 1. In this work, the weights between the input layer and hidden layer are randomly generated and kept fixed. The author reported that the weights between the output layer and hidden layer are of more importance and the rests may not need to be tuned once they are properly initialized.

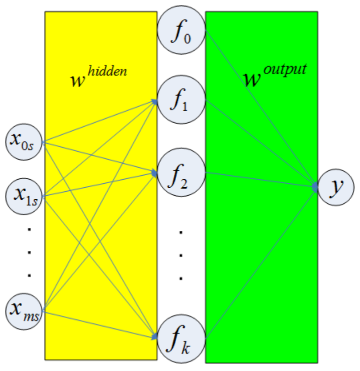

Figure 1: The structure of SLFN in [68]. x means the input feature. The arrows within the yellow rectangle represents the random weights (w^{hidden}) which connect the input feature to the hidden neurons. Those arrows within the green rectangle are the output weights (w^{output}) which need to be optimized. x_{0s} and f_{0s} can be regarded as the bias term in the input and hidden layer. y is the desired output target.

For a given classification problem with limited training data, there are numerous solutions with different parameter setting which is statistically acceptable. In this case, training become much easier because the learning set is only needed to make a rough selection in the parameter space. Setting the parameters in the hidden neurons randomly helps to remove the redundancy of the solution in parameter space and thus makes the solution less sensitive to the resulting parameters compared with other typical learning rule such as back-propagation. In [68], the weights in hidden neurons are set to be uniform random values in $[-1, +1]$ and they suggest to optimize this range in a more appropriate range for the specified application. An alternative choice is to set the hidden neurons to act as “correlators”, which means to fix the weights in hidden neuron with a random subset of the training data.

In [68], the network's output layer weights are optimized by minimizing the following squared error:

$$\epsilon^2 = \sum_{i=1}^{N}(y_i - \sum_{j=0}^{k} w_j f_{ij})^2 \tag{1}$$

where N means the number of data samples and k is the number of hidden neurons. f_{ij} is the activation values of the j^{th} neuron on the i^{th} data sample (f_{i0} is the bias). y_i is the target of the i^{th} data sample.

Denote by

$$F_i = [f_{i0}, f_{i1}, ...f_{ik}]^T \tag{2}$$

is the concatenated vector of activation values of the hidden layer for the i^{th} data sample and the bias. The optimal output layer weight vector, W^{output} , can be easily derived by:

$$W^{output} = R^{-1}P; \quad R = \sum_{i=1}^{N} F_i F_i^T; \quad P = \sum_{i=1}^{N} y_i F_i^T; \tag{3}$$

The above optimization problem can also be regarded as a linear regression which can be solved in a single step.

In [1], general dynamics of complex systems are realized by a feed-forward neural network with random weights. The outputs of the network are feedback to the inputs to generate a time series. By investigating the percent of the systems that exhibit chaos, the distribution of largest Lyapunov exponents, and the distribution of correlation dimensions, they show the probability of chaos approaches unity and the correlation dimension is typically much smaller than the system dimension as the system become more complex due to increasing inputs and neurons.

In [23],"Jacobian Neural Network" (JNN) is proposed which is a polynomial-time randomized algorithm that has probability 1 to give an optimal network. In JNN, the number of hidden neurons can also be learned. They consider a linear combination of two networks where one of them is randomly generated and the other can be analytically achieved from the first random neural network.

3. Random Vector Functional Link Neural Network

Another piece of pioneering work on training neural network with randomization can be found in [56]. The so-called Random Vector Functional Link Neural Network (RVFL) model can be regarded as a semi-random realization of the Functional Link neural networks, where the basic architecture is demonstrated in Fig. 2. The rationale behind

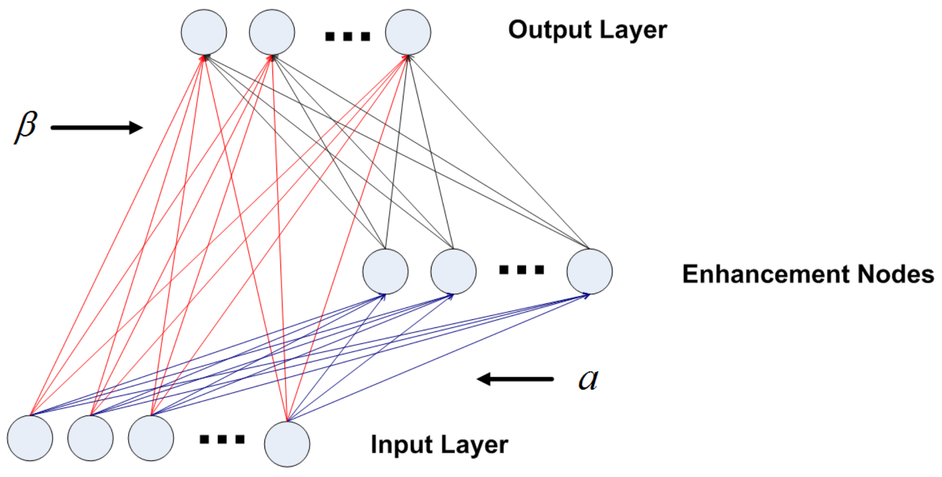

Figure 2: The structure of RVFL. The input features are firstly transformed into the enhanced features by the enhancement nodes where parameters within them are randomly generated. All the original and enhanced features are connected to the output neurons

this architecture is to improve the generalization ability of the network with some enhanced features which can be achieved by some transformation followed by nonlinearity on the original features. The weights a_{ij} from the input to the enhancement nodes are randomly generated such that the activation function $g(a_j^t x + b_j)$ is not saturated most of the time. For RVFL, only the output weights β_j need to be optimized. Suppose the input data has k features and there are J enhancement neurons, there are in total $k + J$ inputs for each output node. Learning is achieved by minimizing the following expression:

$$E = \frac{1}{2N} \sum_{i=1}^{N} (t^i - B^t d^i)^2 \tag{4}$$

where B^t consists of the weight values $\beta_j, j = 1, 2, ...k + J$. There are N input data in total. E is quadratic with respect to each $k + J$ dimensional vector β_j, indicating that the unique minimum can be found in no more than $k + J$ iterations of the learning procedure such as the conjugate gradient method. For simplicity, let $o_p = \sum \beta_j x_{pj}$ be the output of the p^{th} data, then the changes in the weight are set to be $\Delta\beta_{pj} = \eta(t_p - o_p)x_{pj}$. In the $(k + 1)^{th}$ iteration, the weights are updated as $\beta_j(k+1) = \beta_j(k) + \sum_p \Delta\beta_{pj}$. The learning procedure iterates until a stopping criterion is met. An alternative solution within a single learning step can be achieved by the Moore-Penrose pseudo-inverse [37, 55]. In this case, the weight B can be achieved by $B = td^+$, where + represents the Moore-Penrose pseudo inverse.

In [37], some theoretical justifications can be found for RVFL as well as other neural networks with hidden nodes implemented as product of univariate functions or radial basis functions. They formulate the problem as a limit-integral representation of the function to be approximated. Then the limit-integral representation is approximated by using the Monte-Carlo method. It has been proven that with weights and biases from input layer to hidden layer sampled from uniform distribution with a proper range, RVFL is efficient universal approximator for continuous functions on bounded finite dimensional sets. Indeed the overall error of the approximation can be bounded by sum of the error of approximating the function by the integral and the error of the approximating the integral by the Monte-Carlo methods.

A comprehensive evaluation of RVFL was conducted [77]. The authors show that the direct link from the input to output layer plays a key role in the classification ability of RVFL. Moreover, it is advantageous to tune the range of the distribution of the randomized parameters. An application of RVFL for optimal control can be found in [55]. In [15], the author shows the maximum number of hidden nodes is $N - r - 1$ for an RVFL with a constant bias to learn a mapping within a given precision based on an $n-$ dimensional $N-$pattern data set, where r is the rank of the data set. An online learning method is also proposed. Furthermore, a robust weighted least square method is investigated in order to eliminate outliers. In [16], a dynamic stepwise updating algorithm was proposed when a new enhancement node or a new data was added based on the same pseudo-inverse solution. In [16] , authors also proposed several methods to refine the model to deal with the small singular value problem. It is widely accepted that the small singular values may be caused by noise in the data or round-off errors during computations. Small singular values will result in very large weights which will further amplify the noise in test data. Some potential solutions in [16] include: 1). Investigating an upper bound on the weights, 2).Cutting off the singular values and investigating the relation between the cutoff values and the performance of the network in terms of prediction error, 3).Orthogonal least squares learning method or a regularization method or cross-validation methods. These ideas are further explained in [44].

In [36], authors report that compared with RVFL, Multilayer perceptron (MLP) gives a closer approximation to a given function f. Moreover, the functional dependence of the approximation error on the complexity of the model is in both cases of $\frac{1}{\sqrt{N}}$, where N is the number of the hidden neurons. Thus, both the RVFL and MLP are efficient approximators which avoid an exponential increase in n. In the same work, by combining the RVFL with expectation maximisation (EM) method, the author proposed the GM-RVFL method and improvements can be observed. In [2], the author trains a pool of decorrelated RVFL within the negative correlation learning framework to obtain an ensemble classifier. In [46], two algorithms for training RVFL are proposed where training data is distributed throughout a network of agents.

4. Radial Basis Function Network

Another neural network architecture is radial basis function network (RBF network), which was brought to attention in the neural network community by [12, 52, 59], became popular because of its efficiency and ease of training. The RBF network shares a similar architecture as the standard multilayer feedforward neural network. The difference lies in the inputs of the hidden neuron of an RBFnetwork which is the sum of distance between the input pattern and the "center" of the basis function, instead of the weighted sum of the input as in an ANN. In [12], the author demonstrates that the RBF network is sufficent to represent arbitrary nonlinear transformation as determined by a finite training data set of input-output patterns, where the centers of the RBFs can be selected from the training data or randomly generated while randomly generated centres may not reflect the data distribution, thereby leading to a poorer performance. To be more specific, suppose the training data is $(x_i, y_i), i = 1, 2, 3, ...N$, the activation function of the j^{th} hidden neuron for the t^{th} sample is $s_j(x_t) = \sum_{i=1}^{N} \phi(||x_i - c_i||)$, where c_i is the center of this RBF. Thus, the problem can be formulated by the following linear equations:

$$
\begin{aligned}
Y &= \Phi\beta \\
where\ Y &= [y_1, y_2, ...y_N]' \\
\Phi &= \begin{bmatrix} \phi_{11} & \cdots & \phi_{1N} \\ \vdots & \ddots & \vdots \\ \phi_{N1} & \cdots & \phi_{NN} \end{bmatrix} \\
\phi_{ij} &= \phi(\|x_i - c_j\|)
\end{aligned}
\tag{5}
$$

Given the existence of the inverse of matrix Φ, the weights β which connect the RBFs to the output node can be achieved by $\beta = \Phi^{-1}Y$. Micchelli proved [58] that for all N and for a large function of ϕ, the matrix Φ is non-singular if the data points are all distinct.

The above equations offer an efficient solution to the case where the number of RBFs is equal to number of distinct data samples. In practice, one may be interested in obtaining the same performance with minimum model complexity, that is–with minumum number of RBFs. In this case, the matrix Φ may no longer be square. With the "minimum number of RBFs" constraints, the problem usually becomes over-specified (which means the number of the RBFs is smaller than the number of training samples). Thus, a unique inverse on longer exists and we shall adopt a minimum norm least square method by employing the Moore-Penrose pseudo-inverse, Φ^+ of Φ.

5. Recurrent Neural Networks

Recurrent neural network (RNN) is another type of neural networks. Different from feedforward neural network, where activations is piped through the network from the input neurons to output neurons, an RNN has at least one cyclic path of synaptic connections. One typical structure of RNN can be found in Fig. 3.

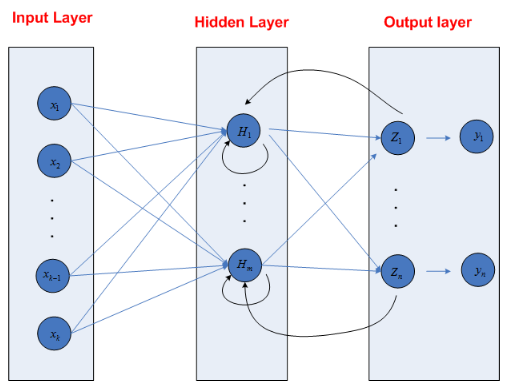

Figure 3: The structure of RNN.

A random RNN is a set of N fully connected neurons. The weights that connect the neurons are randomly initialized. The states of the neurons are $X(t) = x_i(t), i = 1, ..., N$, where each x_i is set proportional to the firing frequency of the $i - th$ neuron. The state dynamics can be modeled by the following discrete time RBF equations:

$$\forall t > 0, \forall i = 1, ..., N, x_i(t+1) = f\left(\sum_{j=1}^{N} w_{ij}x_j(t) + I_i - \Theta_i\right); \tag{6}$$

where f is the activation function, I is N dimensional constant vector and Θ is randomly generated. Time varying extensions of I can be found in [9]. There exist many random RNNs in the literature. Among them, Liquid State Machine [50] randomly fixes the input weights and internal weights, while the output weights are learned by the well-known recursive least squares (RLS) algorithm. Liquid state machine uses spiking activation functions. The Echo State Network [39] works in a fairly similar manner as the liquid state machine. But, the echo state uses continuous models as activation in neurons. Readers are referred to [71, 3] for various extensions along this line of research.

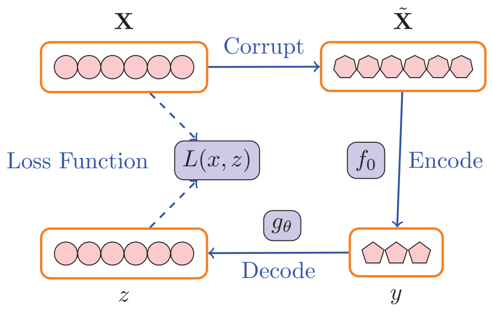

Figure 4: Basic structure of a denoising autoencoder

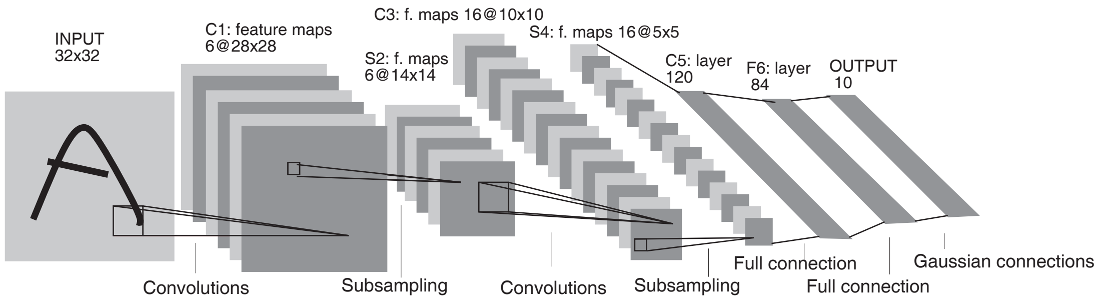

Figure 5: Details of Lenet-5. (Figure adapted from [43]).

6. Deep neural networks and Convolutional neural networks

Deep structures have become a hot research topic since the work in [31]. In deep learning, deep structures with multiple layers of non-linear operations are able to learn high-level abstractions. Such high-level abstraction is the key factor leading to the success of many state-of-the-art systems in vision, language, and other AI-level tasks. Complex training algorithms combined with carefully chosen parameters (i.e, learning rate, mini-batch size, number of epochs) can lead to a deep neural network (DNN) with high-performance.

Autoencoder [31, 7], which is a basic block of recent deep neural networks, can be decomposed into two parts: encoder and decoder. An encoder is a deterministic mapping that maps the input x to a hidden representation y through: $f_\Theta(x) = s(Wx + b)$ where $\Theta = \{W, b\}$, and s is a non-linear activation function such as the sigmoid. In the decoder, the hidden representation is then mapped back to reconstruct the input x. This map is achieved by $g'_\Theta(y) = s(W'y + b')$. Figure 4 shows the structure of a denoising autoencoder. In a denoising autoencoder, firstly the input is corrupted by some random noise and the autoencoder aims to reconstruct the "clean" input. In [72], the author proposed the stacked denoising autoencoders which were trained locally to denoise corrupted versions of their inputs. It was shown on a benchmark of classification problems to yield significantly lower classification error than the stacked autoencoder. These works clearly establish the value of using a randomization based denoising criterion as a tractable unsupervised objective to guide the learning in deep neural networks.

Convolutional Neural Network (CNN) [43] is another neural network model which has been successfully applied to solve many tasks such as digit, object and speech recognition. CNN combines three architectural ideas to ensure some degrees of shift, scale and distortion invariances: shared weights, sub-sampling and local receptive fields. A classic CNN, Lenet-5, is shown in Figure 5.

CNN is composed of alternating convolutional and pooling layers. The ideas of "shared weights" and "local receptive fields" are involved in the convolutional layers. A filter of a pre-defined size (e.g. 5×5 or 7×7) is convolved with the input to obtain the feature map. In order to boost the performance of a CNN, a common approach is to let the network learn a "over-complete" set of feature maps. We can easily stack this architecture into deep architectures by setting the output of one pooling layer to be the input of another convolutional layer. In this case, inputs of each filter in the higher layer can be randomly connected to the output of the lower pooling layer. This kind of randomization based connection is well studied in [43, 25].

In [40], the authors show that the exact values of the filters in CNN is less important than the architecture. They report that a two-stage system with random filters can yield satisfactory results, provided that the proper non-linearities and pooling layers are used. This surprising finding has also been verified by [57], where thousands of convolutional pooling architectures on a number of object recognition tasks are evaluated and the random weights are found to be only slightly worse than those with pretrained weights. The work in [67] further address this phenomenon and found certain convolutional pooling architectures can be inherently frequency selective and translation invariant even with random weights. Thus, a practical method is proposed for fast model selection. They show that the performance of single layer convolutional square pooling networks with random weights is significantly correlated with the performance of such architectures after pretraining and finetuning. Thus, the use of random weights for architecture search will improve the performance of the state-of-the-art systems. Randomization method can also be employed in the pooling layer of the CNN to improve the performance significantly. In [19], selecting local receptive fields is proposed where some features are randomly selected firstly, then the local receptive fields are selected to make sure that each feature within it is most similar to each other according to a pairwise similarity metric. In [76], the authors proposed to randomly pick the activation within each pooling region according to a multinomial distribution, given by the activations within the pooling region. Such randomization based pooling instead of conventional deterministic pooling operations achieve state-of-the-art performance on several benchmark datasets.

As regressors, neural network models the conditional distribution of the output variables Y given the input variables X. Multi-modal conditional distribution is proposed in [70] by using stochastic hidden variables rather than deterministic ones. A new Generalized EM training procedure using importance sampling is proposed to train a stochastic feedforward network with hidden layers composed of both deterministic and stochastic variables to efficiently learn complicated conditional distributions. They achieve superior performance on synthetic and facial expressions datasets compared to conditional restricted boltzmann machines and mixture density networks.

In [8], the authors report empirically and theoretically that randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid for deep neural networks. Compared with neural networks configured by a pure grid search, random search over the same domain is able to find models that are on par with or even better within a small fraction of the computation time. With the same computational budget, random search may find better models by effectively searching a larger as well as less promising configuration space. Gaussian process analysis of the function from hyper-parameters to validation set performance is employed. It reveals that for most data sets only a few of the hyper-parameters really matter. However, different hyper-parameters may be important on different data sets, which makes the commonly adopted grid search approach a poor choice for configuring algorithms for new data sets. In [47], the authors proposed to multiply error signals by random synaptic weights in back-propagation for deep neural networks. They demonstrated that this new mechanism performs as quickly and accurately as back-propagation on a variety of problems.

It is well known that a large feed-forward neural network trained on a small training set will typically perform poorly on held-out test data. To tackle this problem, the authors in [69] propose a method called "dropout" to prevent co-adaptation of feature detectors. In their method, The key idea is to randomly drop units with a pre-defined probability (along with their connections) from a neural network during training. This prevented the units from co-adapting too much. Dropping units created thinned networks during training. Dropout can be seen as an extreme form of bagging in which each model is trained on a single case and each parameter of the model is very strongly regularized by sharing it with the corresponding parameter in all the other models. During testing, all possible thinned networks were combined using an approximate model averaging procedure. The idea of dropout is not limited to feed-forward neural networks. It can be more generally applied to graphical models such as Boltzmann Machines. Random dropout yielded big improvements on many benchmark tasks and sets new record accuracies in speech and object recognition tasks [69].

In [73], DropConnect network was proposed for regularizing large fully-connected layers within neural networks which can be regarded as a generalization of drop out. DropConnect can be regarded as a larger ensemble of deep neural network than dropout. When training with Dropout, a randomly selected subset of activations are set to zero within each layer. DropConnect instead sets a randomly selected subset of weights within the network to zero. In the testing time, DropConnect network uses a sampling based inference which was shown to be better than the mean-inference used by dropout. The authors also give some theoretical insights on why DropConnect regularizes the network [73]. They proved that the Rademacher complexity [6] of the DropConnect is less than the standard models.

Multilayer bootstrap network (MBN) [78] is another deep neural network model where each layer of the network is a group of mutually independent *k*- clusters of which centers are randomly sampled data points. In [78], the author shows the relationship between MBN and product of experts, contrastive divergence learning and sparse coding.

7. Other Randomized Algorithms

Randomization based methods can also be employed in other network configurations. In [27], a special kind of random neural network is proposed. For other neural networks, the activation of neurons is either binary or continuous variables. For random neural network, each neuron is represented by its potential (which is a non-negative integer) and a neuron is considered to be in its firing state if its potential is positive. Readers are referred to [28] for a comprehensive review about training this family of randomized feedforward neural network.

Kernel machines such as the Support Vector Machine are attractive because of their excellent generalization ability. However, one drawback of such method is that the kernel matrix (Gram matrix) scales poorly with the size of the training data. In [61], the authors proposed to map the input data to a randomized low-dimensional feature space where the inner products uniformly approximate many popular kernels. The proposed random features are demonstrated to be powerful and economical for large scale learning. Another research in [20] scaled up the kernel methods by a novel concept called “doubly stochastic functional gradient”. It is well known that many kernel methods can be solved by convex optimization algorithms. Dai et al [20] solved the optimization problems by making two unbiased stochastic approximations to the functional gradient. One used random training points and another used random features associated with the kernel. Fast convergence rate and tight bound were reported in their method.

In [62], motivated by the fact that randomization is computationally cheaper than optimization based learning, the authors proposed to replace the minimization with randomization in a classifier similar to kernel machines, which computes a weighted sum of their inputs after passing them through a pool of arbitrarily randomized nonlinearities. The error bound can be roughly decomposed into 2 parts. The first part of the error bound indicates that the lowest true risk attainable by a function in the family of the randomized classifiers is close to the lowest true risk attainable. The second part of the error bound shows that the true risk of every function in the family of the randomized classifiers is close to its empirical risk.

In [5], the author showed that the conventional back propagation would not work on neural networks whose neurons have hard-limiting input-output characteristics. In this case, the derivatives of the error function are not available. However, the authors showed if the network weights are random variables with smooth distribution functions, the probability of a hard-limiting unit taking one of its two possible values is a continuously differentiable function.

It is not difficult to find many attempts to train neural networks by using evolutionary algorithms. Evolutionary algorithm is a special randomization method inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Population-based algorithms, such as genetic algorithm [34, 33], particle swarm optimization [41, 51], immune approaches [74, 21], multi-objective optimization [75, 18], are known to improve the training of neural network. Other approaches such as simulated annealing and metropolis have also been successfully applied for neural network training [24, 11].

8. Future directions

Though a large body of literature has been published to exploit randomization for training neural networks, there are still several research gaps in this field. One can investigate the effects of a specific randomization scheme (sparse or non-sparse, uniform or gaussian, etc.) on different neural network classes by using the compressive sensing theory [13, 45, 14, 4]. Another major gap is the lack of extensive experimental comparisons between different randomization based networks [68, 54, 26, 50, 39, 60]. Randomized neural networks for ensemble learning are also under-researched. It is widely accepted that ensemble methods can benefit from low bias and high variance learner [63]. Thus, it worth investigating how to integrate randomized neural networks by using ensemble learning frameworks.

In [32], the authors investigated the approximation ability of standard multilayer feedforward networks with as few as a single hidden layer. They showed when the activation function is bounded and nonconstant, the standard multilayer feedforward networks are universal approximators. In [37], the authors showed when the parameters of the hidden layers are randomly sampled from a uniform distribution within a proper range, the resulting neural network, RVFL, is a universal approximator for a continuous function on a bounded finite dimensional set with efficient convergence rate. However, how to optimize the range for the random parameters remains untouched in the literature. Hence, this is another area where further research is required. Also, we should investigate the random parameter settings in other models such as polynomial, wavelet and so on.

"Big data" is a hot topic recently leading to an upsurge of research. Deep learning [31] has been gaining its popularity in machine learning and computer vision community. It has been shown that high-level abstraction which comes from deep structure is a key factor leading to the success of many state-of-the-art systems in vision, language, and other AI-level tasks. Deep networks are much more difficult to train than shallow ones because they need a relatively large training data set to tune a large number of parameters. Despite the surge in interest in deep networks as more large scale data becomes available [22, 35], the theoretical aspects have remained under-researched. In [42], the authors showed that deep but narrow networks trained with a greedy layer-wise unsupervised learning algorithm do not require more parameters than shallow ones to achieve universal approximation. However, when it comes to deep random neural network, it remains unclear regarding how to set the random parameters. Most importantly, the performance gap between deep random neural network and deep neural network trained with back propagation is wide in favor of back propagation. Moreover, when it comes to bigdata or large-scale learning, high dimensional feature space is usually preferred [17]. In this case, commonly used approach for randomized neural network in prime space such as Eq. (3) whose complexity is between quadratic and cubic with respect to the data samples may become computationally intractable. Thus, one may alternatively turn to other approaches such as conjugate gradient [49]. On the other hand, solutions in the dual space [66] has a time complexity of $O(n^2p)$ (p is the number of features and n is the number of data samples), which can also be very slow. In this case, randomized approximation methods such as [48] and [61] can be more efficient and reliable.

9. Conclusion

In this article, we presented an extensive survey on randomized methods for training neural networks, the use of randomization in kernel machines and related topics. We divided this family into several methods based on the network configuration. We believe that, this article, the first survey on randomized methods for training neural network, offers valuable insights into this important research topic. We also offered several potential future research directions. We trust that this article will encourage further advancements in this field.

Acknowledgement

Authors wish to thank Guest Editor Associate Professor Dianhui Wang and reviewers for providing us with their valuable comments.

References

[1] Albers, D. J. (1996). Dynamical behavior of artificial neural networks with random weights. In *Intelligent Engineering Systems Through Artificial Neural Networks*, pages 17–22.

[2] Alhamdoosh, M. and Wang, D. (2014). Fast decorrelated neural network ensembles with random weights. *Information Sciences*, 264:104–117.

[3] Bakırcıoğlu, H. and Koçak, T. (2000). Survey of random neural network applications. *European Journal of Operational Research*, 126(2):319–330.

[4] Baraniuk, R. (2007). Compressive sensing. *IEEE Signal Processing Magazine*, 24(4).

[5] Barlett, P. L. and Downs, T. (1992). Using random weights to train multilayer networks of hard-limiting units. *IEEE Transactions on Neural Networks*, 3(2):202–210.

[6] Bartlett, P. L. and Mendelson, S. (2003). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482.

[7] Bengio, Y., LeCun, Y., et al. (2007). Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, 34(5).

[8] Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(1):281–305.

[9] Berry, H. and Quoy, M. (2006). Structure and dynamics of random recurrent neural networks. *Adaptive Behavior*, 14(2):129–137.

[10] Block, H. D. (1962). The perceptron: A model for brain functioning. i. *Reviews of Modern Physics*, 34:123–135.

[11] Boese, K. D. and Kahng, A. B. (1993). Simulated annealing of neural networks: thecooling'strategy reconsidered. In *1993 IEEE International Symposium on Circuits and Systems*, pages 2572–2575. IEEE.

[12] Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document.

[13] Candes, E. J. and Tao, T. (2005). Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215.

[14] Candes, E. J. and Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425.

[15] Chen, C. P. (1996). A rapid supervised learning neural network for function interpolation and approximation. *IEEE Transactions on Neural Networks*, 7(5):1220–1230.

[16] Chen, C. P. and Wan, J. Z. (1999). A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(1):62–72.

[17] Chen, D., Cao, X., Wen, F., and Sun, J. (2013). Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *2013 IEEE Conference on Computer Vision and Pattern Recognition ,*, pages 3025–3032. IEEE.

[18] Chiam, S. C., Tan, K. C., and Al Mamun, A. (2007). Multiobjective evolutionary neural networks for time series forecasting. In *Evolutionary Multi-Criterion Optimization*, pages 346–360. Springer.

[19] Coates, A. and Ng, A. Y. (2011). Selecting receptive fields in deep networks. In *Advances in Neural Information Processing Systems*, pages 2528–2536.

[20] Dai, B., Xie, B., He, N., Liang, Y., Raj, A., Balcan, M.-F. F., and Song, L. (2014). Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049.

[21] de Castro, L. N. and Von Zuben, F. J. (2001). Immune and neural network models: theoretical and empirical comparisons. *International Journal of Computational Intelligence and Applications*, 1(03):239–257.

[22] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 248–255. IEEE.

[23] Elisseeff, A. and Paugam-Moisy, H. (1999). JNN, a randomized algorithm for training multilayer networks in polynomial time. *Neurocomputing*, 29(1):3–24.

[24] Engel, J. (1988). Teaching feed-forward neural networks by simulated annealing. *Complex Systems*, 2(6):641–648.

[25] Fan, J., Xu, W., Wu, Y., and Gong, Y. (2010). Human tracking using convolutional neural networks. *IEEE Transactions on Neural Networks*, 21(10):1610–1623.

[26] Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems? *Journal of Machine Learning Research*, 15(1):3133–3181.

[27] Gelenbe, E. (1989). Random neural networks with negative and positive signals and product form solution. *Neural Computation*, 1(4):502–510.

[28] Georgiopoulos, M., Li, C., and Kocak, T. (2011). Learning in the feed-forward random neural network: A critical review. *Performance Evaluation*, 68(4):361–384.

[29] Gori, M. and Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86.

[30] Haykin, S. and Network, N. (2004). Neural Networks: A comprehensive foundation. *Neural Networks*, 2(2004).

[31] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[32] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

[33] Hu, Y.-C. (2009). Functional-link nets with genetic-algorithm-based learning for robust nonlinear interval regression analysis. *Neurocomputing*, 72(7):1808–1816.

[34] Hu, Y.-C. and Tseng, F.-M. (2007). Functional-link net with fuzzy integral for bankruptcy prediction. *Neurocomputing*, 70(16):2959–2968.

[35] Huang, G., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, MA.

[36] Husmeier, D. and Taylor, J. G. (1998). Neural networks for predicting conditional probability densities: Improved training scheme combining EM and RVFL. *Neural Networks*, 11(1):89–116.

[37] Igelnik, B. and Pao, Y.-H. (1995). Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE Trans on Neural Networks*, 6(6):1320–1329.

[38] Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307.

[39] Jaeger, H. (2002). Adaptive nonlinear system identification with echo state networks. In *Advances in Neural Information Processing Systems*, pages 593–600.

[40] Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153. IEEE.

[41] Kennedy, J. (2010). Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer.

[42] Le Roux, N. and Bengio, Y. (2010). Deep belief networks are compact universal approximators. *Neural Computation*, 22(8):2192–2207.

[43] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[44] Li, H., Chen, C. P., and Huang, H.-P. (2010). *Fuzzy neural intelligent systems: mathematical foundation and the applications in engineering*. CRC Press.

[45] Li, P., Hastie, T. J., and Church, K. W. (2006). Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 287–296. ACM.

[46] Li, W., Wang, D., and Chai, T. (2015). Multisource data ensemble modeling for clinker free lime content estimate in rotary kiln sintering processes. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(2):303–314.

[47] Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2014). Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*.

[48] Lu, Y., Dhillon, P., Foster, D. P., and Ungar, L. (2013). Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in Neural Information Processing Systems*, pages 369–377.

[49] Luenberger, D. G. (1973). *Introduction to linear and nonlinear programming*, volume 28. Addison-Wesley Reading, MA.

[50] Maass, W., Natschläger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.

[51] Mendes, R., Cortez, P., Rocha, M., and Neves, J. (2002). Particle swarms for feedforward neural network training. *Learning*, 6(1).

[52] Moody, J. and Darken, C. (1988). *Learning with localized receptive fields*. Yale Univ., Department of Computer Science.

[53] Pao, Y.-H. (1989). *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[54] Pao, Y.-H., Park, G.-H., and Sobajic, D. J. (1994). Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180.

[55] Pao, Y.-H. and Phillips, S. M. (1995). The functional link net and learning optimal control. *Neurocomputing*, 9(2):149–164.

[56] Pao, Y.-H. and Takefuji, Y. (1992). Functional-link net computing. *IEEE Computer*, 25(5):76–79.

[57] Pinto, N., Doukhan, D., DiCarlo, J. J., and Cox, D. D. (2009). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput Biol*, 5(11):e1000579.

[58] Poggio, T. and Girosi, F. (1989). A theory of networks for approximation and learning. Technical report, DTIC Document.

[59] Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

[60] Principe, J. C. and Chen, B. (2015). Universal approximation with convex optimization: Gimmick or reality?[discussion forum]. *IEEE Computational Intelligence Magazine*, 10(2):68–77.

[61] Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184.

[62] Rahimi, A. and Recht, B. (2009). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems*, pages 1313–1320.

[63] Ren, Y., Zhang, L., and Suganthan, P. N. Ensemble classification and regression – recent developments, applications and future directions. *IEEE Computational Intelligence Magazine, doi: 10.1109/MCI.2015.2471235*.

[64] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386.

[65] Rosenblatt, F. (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document.

[66] Saunders, C., Gammerman, A., and Vovk, V. (1998). Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning*, pages 515–521. Morgan Kaufmann.

[67] Saxe, A., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. Y. (2011). On random weights and unsupervised feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1089–1096.

[68] Schmidt, W. F., Kraaijveld, M. A., and Duin, R. P. (1992). Feedforward neural networks with random weights. In *11th IAPR International Conference on Pattern Recognition,*, pages 1–4. IEEE.

[69] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[70] Tang, Y. and Salakhutdinov, R. R. (2013). Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems*, pages 530–538.

[71] Timotheou, S. (2010). The random neural network: a survey. *The Computer Journal*, 53(3):251–267.

[72] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408.

[73] Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066.

[74] Wang, L. and Courant, M. (2002). A novel neural network based on immunity. In *IC-AI*, pages 147–153. Citeseer.

[75] Yusiong, J. P. T. and Naval Jr, P. C. (2006). Training neural networks using multiobjective particle swarm optimization. In *Advances in Natural Computation*, pages 879–888. Springer.

[76] Zeiler, M. D. and Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. In *2013 International Conference on Learning Representations*.

[77] Zhang, L. and Suganthan, P. N. A comprehensive evaluation of random vector functional link networks. *Information Sciences, doi: http://dx.doi.org/10.1016/j.ins.2015.09.025*.

[78] Zhang, X. (2014). Nonlinear dimensionality reduction of data by deep distributed random samplings. In *2014 Proceedings of the Sixth Asian Conference on Machine Learning*.