FEEDFORWARD NEURAL NETWOTKS WITHOUT ORTHONORMALIZATION *

Lei Chen, Hung Keng Pung and Fei Long
Network Systems and Service Lab., Department of Computer Science, National University of Singapore, Singapore

Keywords: Feedforward neural networks, kernel function, orthonormal transformation, Extreme Learning Machine(ELM), approximation, generalization performance.

Abstract: Feedforward neural networks have attracted considerable attention in many fields mainly due to their approximation capability. Recently, an effective noniterative technique has been proposed by Kaminski and Strumillo(Kaminski and Strumillo, 1997), where kernel hidden neurons are transformed into an orthonormal set of neurons by using Gram-Schmidt orthonormalization. After this transformation, neural networks do not need recomputation of network weights already calculated, therefore the orthonormal neural networks can reduce computing time. In this paper, we will show that it is equivalent between neural networks without orthonormal transformation and the orthonormal neural networks, thus we can naturally conclude that such orthonormalization transformation is not necessary for neural networks. Moreover, we will extend such orthonormal neural networks into additive neurons. The experimental results based on some benchmark regression applications further verify our conclusion.

1 INTRODUCTION

Feedforward neural networks(FNNs) have been successfully applied in many nonlinear approximation and estimation fields due to their approximation capability, which ensures that single-hidden-layer feedforward neural networks(SLFNs) can accurately prescribe target functions with a finite number of neurons. The output of an SLFN with L hidden neurons can be represented by $f_L = \sum_{i=1}^{L} \beta_i g(\mathbf{a}_i, b_i, \mathbf{x})$, where $\mathbf{a}_i$ and b_i are the learning parameters of hidden neurons and β_i is the weight connecting the i-th hidden neuron to the output neuron. Based on different parameter combinations, there are two main SLFN network architectures: additive neurons and kernel neurons. For the additive neurons, the activation function $g(x) : R \rightarrow R$ takes the form $g(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i)$, where $\mathbf{a}_i \in R^n$ is the weight vector connecting the input layer to the i-th hidden neuron and $b_i \in R$ is the bias of the i-th hidden neuron. $\mathbf{a}_i \cdot \mathbf{x}$ denotes the inner product of vectors $\mathbf{a}_i$ and $\mathbf{x}$ in R^n. For the kernel neurons, the activation function $g(x) : R \rightarrow R$ takes the form $g(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|)$, where $\mathbf{a}_i \in R^n$ is the center of the i-th RBF neuron and $b_i \in R^+$ is the impact of the i-th RBF neuron. R^+ indicates the set of all positive real value.

Recently, an effective noniterative technique has been proposed by Kaminski and Strumillo(Kaminski and Strumillo, 1997), where kernel hidden neurons are transformed into an orthonormal set of neurons by using Gram-Schmidt orthonormalization. After this transformation, FNNs do not require recomputation of network weights already calculated, which can remarkably reduce computing time.

Through in-depth analysis, we have found that neural networks without orthonormal transformation (also named as ELM(Huang et al., 2006)) is equivalent to the orthonormal neural networks, therefore such orthonormal transformation is not necessary for feedforward neural networks. Moreover, the original orthonormal neural networks are only suitable for kernel neurons. In this paper, we will extend such orthonormal neural networks into additive func-

*The work is funded by SERC of A*Star Singapore through the National Grid Office (NGO) under the research grant 0520150024 for two years.

Chen L., Keng Pung H. and Long F. (2007).
FEEDFORWARD NEURAL NETWOTKS WITHOUT ORTHONORMALIZATION.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - AIDSS*, pages 420-423
DOI: 10.5220/0002374704200423

tion neural networks. Experimental results based on some benchmark regression applications also verify our conclusion: neural networks without orthonormalization transformation may achieve faster training speed for the same generalization performance.

2 SINGLE-HIDDEN-LAYER FEEDFORWARD NETWORKS

Before we show our main results, we need to introduce some symbols for a standard SLFN. For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, \cdots, N$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^T \in \mathbf{R}^n$ is an input vector and $\mathbf{t}_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^T \in \mathbf{R}^m$ is a target vector. A standard SLFN with L hidden neurons with activation function $g(x)$ can be expressed as

$$\sum_{i=1}^{L} \beta_i g(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{o}_j, \ \ j = 1, \cdots, N,$$

where $\mathbf{o}_j$ is the actual output of SLFN. As mentioned in introduction section, $g(\mathbf{a}_i, b_i, \mathbf{x}_j)$ may be additive model or RBF model.

Definition 2.1 *A standard SLFN with L hidden neurons can learn N arbitrary distinct samples* $(\mathbf{x}_i, \mathbf{t}_i)$, $i = 1, \cdots, N$, *with* **zero error** *means that there exist parameters* $\mathbf{a}_i$ *and* b_i, *for* $i = 1, \cdots, L$, *such that*

$$\sum_{i=1}^{N} \|\mathbf{o}_i - \mathbf{t}_i\| = 0.$$

According to Definition 2.1, our ideal objective is to find proper parameters $\mathbf{a}_i$ and b_i such that

$$\sum_{i=1}^{L} \beta_i g(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \ \ j = 1, \cdots, N,$$

The above N equations can be expressed as

$$\mathbf{H}\beta = \mathbf{T} \tag{1}$$

where $\beta = [\beta_1, \cdots, \beta_L]^T$, $\mathbf{T} = [t_1, \cdots, t_N]^T$ and the matrix $\mathbf{H}$ is called as the hidden layer matrix of the SLFN.

In (Kaminski and Strumillo, 1997), they showed that by randomly choosing centers of kernel neurons, the column vectors of matrix $\mathbf{H}$ are linearly independent. In order to extend the corresponding result into additive neurons, we need to introduce one lemma:

Lemma 2.1 P.491 of (Huang et al., 2006) *Given a standard SLFN with N hidden nodes and activation function* $g : R \rightarrow R$ *which is infinitely differentiable in any interval, for N arbitrary distinct samples* $(\mathbf{x}_i, \mathbf{t}_i)$, *where* $\mathbf{x}_i \in \mathbf{R}^n$ *and* $\mathbf{t}_i \in \mathbf{R}^m$, *for any* $\mathbf{a}_i$ *and* b_i *chosen from any intervals of* $\mathbf{R}^n$ *and* $\mathbf{R}$, *respectively, according to any continuous probability distribution, then with probability one, the hidden layer output matrix* $\mathbf{H}$ *of the SLFN is invertible and* $\|\mathbf{H}\beta - \mathbf{T}\| = 0$.

Lemma 2.1 illustrates that when the number of neurons L is equal to the number of samples N, the corresponding hidden layer matrix $\mathbf{H}$ is nonsingular such that SLFN can express those samples with zero error. The value of β could be calculated by $\mathbf{H}^{-1}\mathbf{T}$. In another word, it means that the column vectors of $\mathbf{H}$ are linearly independent each other for any infinitely differentiable function with almost all the random parameters, which is consistent with the conclusion of Kaminski and Strumillo(Kaminski and Strumillo, 1997)(p. 1179). However, we should note that Kaminski and Strumillo's conclusion is only suitable for kernel functions.

According to Lemma 2.1, for SLFNs with any infinitely differentiable additive neuron $g(x)$, the hidden neuron parameters $\mathbf{a}_i$ and b_i may be assigned with random values such that SLFN learn training samples with zero error. In fact, full rank $\mathbf{H}$, i.e., $L = N$, is not necessary. The number of neurons L will be far less than N in most cases. In this case $(L < N)$, the linearly independent property is still ensured, then SLFN can approach a nonzero training error ε by using the Moore-Penrose generalized inverse of matrix $\mathbf{H}$, i.e., $\beta = \mathbf{H}^{\dagger}\mathbf{T}$, where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix.

3 NO NEED FOR ORTHONORMALIZATION

In this section, we will demonstrate the equivalence between neural networks without orthonormal transformation and the orthonormal neural networks. First, we introduce Gram-Schmidt orthonormalization in brief. For the simplicity, we denote $g_j(\mathbf{x}) = g(\mathbf{a}_j, b_j, \mathbf{x})$. Our aim is to find proper parameters such that

$$\beta_1 g_1(\mathbf{x}_i) + \cdots + \beta_L g_L(\mathbf{x}_i) = t_i, i = 1, \cdots, N \tag{2}$$

where $t_i = f(\mathbf{x}_i)$.

Multiplying equation (2) by $g_j(\mathbf{x}_i)$ and adding the corresponding L equations for $i = 1, \cdots, N$, we have

$$\beta_1 \sum_{i=1}^{N} g_1(\mathbf{x}_i) g_j(\mathbf{x}_i) + \cdots + \beta_L \sum_{i=1}^{N} g_L(\mathbf{x}_i) g_j(\mathbf{x}_i) = \sum_{i=1}^{N} y_i g_j(\mathbf{x}_i), j = 1, \cdots, L \tag{3}$$

Similar to (Kaminski and Strumillo, 1997) (p. 1179), the inner product of two functions is defined

as

$$\langle u(\mathbf{x}), v(\mathbf{x})\rangle = \sum_{i=1}^{N} u(\mathbf{x}_i)v(\mathbf{x}_i) \tag{4}$$

where N is the number of training samples, then equations (3) can be written as

$$\begin{aligned}&\beta_1 \langle g_1(\mathbf{x}), g_j(\mathbf{x})\rangle + \cdots + \beta_L \langle g_L(\mathbf{x}), g_j(\mathbf{x})\rangle \\ &= \langle f(\mathbf{x}), g_j(\mathbf{x})\rangle, j = 1, \cdots, L\end{aligned} \tag{5}$$

The above L equations can be rewritten as

$$\widetilde{\mathbf{H}}\beta = \widetilde{\mathbf{T}}$$

where

$$\widetilde{\mathbf{T}} = \begin{pmatrix} \langle f(\mathbf{x}), g_1(\mathbf{x})\rangle \\ \vdots \\ \langle f(\mathbf{x}), g_L(\mathbf{x})\rangle \end{pmatrix}$$

and

$$\widetilde{\mathbf{H}} = \begin{bmatrix} \langle g_1(\mathbf{x}), g_1(\mathbf{x})\rangle & \cdots & \langle g_L(\mathbf{x}), g_1(\mathbf{x})\rangle \\ \vdots & \ddots & \vdots \\ \langle g_1(\mathbf{x}), g_L(\mathbf{x})\rangle & \cdots & \langle g_L(\mathbf{x}), g_L(\mathbf{x})\rangle \end{bmatrix}_{L\times L}$$

We call $\widetilde{\mathbf{H}}$ as inner product hidden layer matrix. If $\{g_k(\mathbf{x})\}_{k=1}^{L}$ are linearly independent each other, then the solution of the linear system (2) is unique. In another word, the hidden-to-output weights calculated by the inverse of hidden matrix $\widetilde{\mathbf{H}}$ are the same as the hidden-to-output weights calculated by the inverse of hidden matrix $\mathbf{H}$, i.e.,

$$\mathbf{H}^{\dagger}\mathbf{T} = \widetilde{\mathbf{H}}^{-1}\widetilde{\mathbf{T}} \tag{6}$$

If $\{g_k(\mathbf{x})\}_{k=1}^{L}$ are orthonormal each other, the diagonal parts of $\widetilde{\mathbf{H}}$ are one and others parts are zero, then the hidden-to-output weights β can be expressed as

$$\beta_k = \langle f(\mathbf{x}), g_k(\mathbf{x})\rangle = \sum_{i=1}^{N} y_i g_k(\mathbf{x}_i) \tag{7}$$

However, as Kaminski and Strumillo said, the set of functions $\{g_k(\mathbf{x})\}_{k=1}^{L}$ is not orthonormal each other, so our purpose is to use orthonormal transformations to construct orthonormal basis. In Kaminski and Strumillo's paper(Kaminski and Strumillo, 1997), they introduce Gram-Schmidt Orthonormalization process in detail.

By applying the standard Gram-Schmidt orthonormalization algorithm, the sequence $\{g_1(\mathbf{x}), g_2(\mathbf{x}), \cdots, g_L(\mathbf{x})\}$ are transformed as an orthonormal set of basis functions $\{u_1(\mathbf{x}), u_2(\mathbf{x}), \cdots, u_L(\mathbf{x})\}$, i.e.,

$$[u_1(\mathbf{x}), u_2(\mathbf{x}), \cdots, u_L(\mathbf{x})] = [g_1(\mathbf{x}), g_2(\mathbf{x}), \cdots, g_L(\mathbf{x})]\cdot\mathbf{V} \tag{8}$$

where $\mathbf{V}$ is an upper triangular matrix. Its detail expression can be refereed to equations({P.1182 of (Kaminski and Strumillo, 1997)}). After Gram-Schmidt transformation, the new hidde-to-output weights $\{\alpha_i\}_{i=1}^{L}$ are expressed as

$$\alpha_i = \langle f(\mathbf{x}), u_i(\mathbf{x})\rangle, i = 1, \cdots, L \tag{9}$$

We set $\alpha = [\alpha_1, \cdots, \alpha_L]^T$, and let β denote the hidden-to-output weights by the generalized inverse of the hidden matrix $\mathbf{H}$, i.e., $\beta = \mathbf{H}^{\dagger}\mathbf{T}$. According to equation (6), we have

$$\begin{aligned}\alpha &= \mathbf{U}^{\dagger}\mathbf{T} \\ &= \mathbf{U}^{\dagger}\mathbf{H}\beta\end{aligned} \tag{10}$$

Equation (10) shows the corresponding equivalent relation between neural networks with Gram-Schmidt transformation and without Gram-Schmidt transformation. It also illustrates that such orthonormal transformation is no need for neural networks.

In order to ensure the validity of equation (10), we need the following precondition: $\{g_k(\mathbf{x})\}_{k=1}^{L}$ are linearly independent each other. According to Lemma 2.1 and the statements of the paper({P.1179 of (Kaminski and Strumillo, 1997)}), we have the following conclusion: given a standard SLFN with any infinitely differentiable additive neuron or any kernel neuron, for any $\mathbf{a}_i$ and b_i chosen from any intervals of $\mathbf{R}^n$ and $\mathbf{R}$, respectively, according to any continuous probability distribution, we can obtain the equivalent relation(10), i.e., orthonormal neural networks is no longer needed.

4 PERFORMANCE EVALUATION

In order to verify our conclusion, we will compare the simulation results between ELM(without orthonormal transformation) and such orthonormalization neural networks based on some benchmark regression applications. Neural networks without orthonormalization transformation, i.e., ELM, may achieve faster training speed under the same generalization performance. Although the scope of additive neurons can be extended, we only choose Gaussian kernel activation function for all the simulations in this section.

For simplicity, all the inputs data are normalized into the range $[-1, 1]$ in our experiments. Neural networks with ELM and with Gram-Schmidt Orthonormalization both are assigned the same of number of hidden neurons, i.e. 30 neurons. All the simulations are running in MATLAB 6.5 environment and the same PC with Pentium 4 3.0 GHZ CPU and 1G RAM. The kernel function used in the simulations is the Gaussian function $\phi(\mathbf{x}) = \exp(-\gamma\|\mathbf{x} - \mu\|^2)$, where the centers μ_i are randomly chosen from the

range $[-1,1]$ whereas the impact factor γ is chosen from the range $(0,0.5)$.

Based on 5 real world benchmark regression datasets, the performance of neural networks with ELM and transformed by Gram-Schmidt Orthonormalization will be given out. Table 1 gives the characteristics of these regression datasets.

Table 1: Specification of 5 Benchmark Regression Datasets.

Name	No. of Observations		Attributes
	Training	Testing	
Abalone	2000	2177	8
Airplane	450	500	9
Boston	250	256	13
Census	10000	12784	8
Elevators	4000	5517	6

Table 2: Comparison of Average Testing Root Mean Square Error.

Name	ELM	Gram-Schmidt
Abalone	0.0784	0.0772
Airplane	0.0481	0.0491
Boston	0.1095	0.1117
Census	0.0758	0.0760
Elevators	0.0604	0.0606

For each problem, 50 trials have been done. Table 2 gives out the testing root mean square error(RMSE) results of ELM and Gram-Schmidt orthonormalization neural networks with the 30th neuron. Seen from Table 2, the two neural networks both achieve good generalization performance with almost the same error level, which also verify our conclusion: The hidden-to-output weights directly determined by hidden layer matrix is the same solution as the hidden-to-out weights calculated by orthonormal inner product hidden layer matrix.

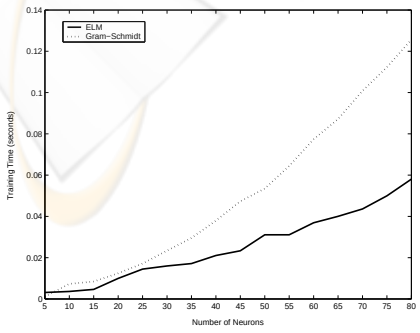

Figure 1: Comparison of training time curves with Gaussian kernel neurons between ELM and Gram-Schmidt Orthonormalization for Airplane case.

Table 3: Comparison of Average Mean Training Time (seconds).

Name	ELM	Gram-Schmidt
	Training Time	Training Time
Abalone	**0.0717**	0.0942
Airplane	**0.0159**	0.0234
Boston	**0.0087**	0.0187
Census	**0.2853**	0.4280
Elevators	**0.0927**	0.1449

The mean training time of ELM and Gram-Schmidt orthonormalization neural networks with the 30th neuron are illustrated in Table 3. From Table 3, we can know that neural networks without orthonormalization transformation take less training time than neural networks with orthonormalization transformation. In the absence of orthonormalization transformation, neural networks can have faster training speed. Fig 1 records training time of Airplane case from 5th neuron to 80th neuron every 5 steps. Seen from Fig 1, with the growth of neuron number, the difference in training time between ELM and Gram-Schmidt orthonormalization neural networks becomes greater, which further verifies the correctness of our conclusion.

5 CONCLUSION

An orthonormal kernel neural networks have been proved to be a fast learning mechanism in (Kaminski and Strumillo, 1997). However, in this paper, we have rigorously proved and demonstrated that such orthonormalization transformation is not necessary for neural networks. Therefore neural networks without orthonormalization transformation can run faster than neural networks with orthonormalization transformation while achieving the same generalization performance. We have also extended in this paper the applied scope of activation functions into additive neurons. Some benchmark regression problems further verify our conclusion.

REFERENCES

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K. (2006). Extreme learning machine: Theorey and applications. *Neurocomputing*, 70:489–501.

Kaminski, W. and Strumillo, P. (1997). Kernel orthonormalization in radial basis function neural networks. *IEEE Transactions On Neural Networks*, 8(5):1177–1183.