Sparse Wavelet Networks
Amir Reza Sadri, Mehmed Emre Celebi, Senior Member, IEEE, Nazanin Rahnavard, Senior Member, IEEE, and Satish E. Viswanath

Abstract—A wavelet network (WN) is a feed-forward neural network that uses wavelets as activation functions for the neurons in its hidden layer. By predetermining the wavelet positions and dilations, the WN can turn into a linear regression model. The common approach for the construction of these WN families is to use least-squares type algorithms. In this letter, we propose a novel approach by formulating a WN as a sparse linear regression problem, which we call a sparse wavelet network (SWN). In this WN, the problem of calculating the unknown inner parameters of the network becomes that of finding the sparse solution of an under-determined system of linear equations. Our sparse solution algorithm is a non-convex sparse relaxation approach inspired by smoothed L0 (SL0), a distinguished sparse recovery algorithm. The proposed SWN can be applied as a tool for the prediction and identification of dynamical systems.

Index Terms—Wavelet network, sparse representation, non-convex regularization, system identification.

I. INTRODUCTION

PARSE modeling is a flourishing interdisciplinary field of research that bridges signal processing, machine learning, and statistics. It is particularly advantageous in selecting or constructing a small set of predictive variables in cases where the aim is to find the input and output of a system relationship [1]. Building on sparse modeling, in this letter, we propose a novel wavelet network (WN) that has the potential to be used in various areas, for example, in engineering disciplines [2]–[5].

The inherent time-frequency localization property of the wavelet basis makes them more effective than other basis functions. This insight inspired the concept of WNs by using wavelets as the basic components of a traditional neural network [6]–[9]. Depending on the types of wavelets and network training scheme used, there are different categories of WNs [10]. The adaptive wavelet network (AWN) is a primitive type of WN that takes advantage of the continuous wavelet transform for the formation of the network building blocks and a gradient type algorithm for model training [11]. Model initialization and training complications often limit AWNs to low dimensional applications [6].

A WN is called fixed grid wavelet network (FGWN) if it originates from the discrete wavelet transform with predefined network inner parameters (the wavelet shifts and scales) [11]. The FGWNs basically act by using various model structures or different parameter estimation algorithms [12]. Substantial techniques for modifying and improving the efficiency of FGWNs have been created in the literature. For example, in [6] multiscale wavelet decomposition was applied as the model construction and the orthogonal least-squares algorithm was applied for computing the network outer parameters. Li et al. extended the model structure based on multi-wavelet basis functions and refined the associated regression using a block least mean squares method [8] and an ultra-orthogonal forward regression algorithm aided by mutual information [12].

As a linear regression model, the output vector of an FGWN can be represented as the multiplication of a wavelet matrix and the coefficient vector. The common approach for finding the coefficient vector is based on greedy strategies such as forward selection which are highly suboptimal [13]. Since the construction of the wavelet matrix is based on the positions and dilations of wavelet coefficients, in order to simplify computational complexity, the FGWN regression problem may be considered as an optimization equation and equivalently as an under-determined systems of linear equations (USLE). Considering the sparse solution of a USLE taken from the corresponding FGWN, which is equivalent to the hidden layer weights, a network with low inner dimension is achieved. This procedure might be useful for high dimensional problems. In the current study, we take an FGWN as a sparse linear regression problem, which we refer to as a sparse wavelet network (SWN). Our proposed algorithm for finding the sparse solution is based on the graduated non-convexity (GNC) method and in particular, the smoothed $\ell_1$ norm (SL0 algorithm) which is an effective and fast approach [14]. This letter is a major contribution to the literature on WN for at least two reasons: (i) sparse modeling of the WN which brings about a network with simple internal structure one that is easy to implement; (ii) analyzing the convergence of the SL0 method using $\ell_0$ norm approximation with a non-convex but gradient-Lipschitz function.

II. STRUCTURE OF SWN

Assume that the observations of input-output data pairs are as \{$(x^{(p)}, y^{(p)}) : x^{(p)} \in \mathbb{R}^n, y^{(p)} \in \mathbb{R}, p = 1, \ldots, P$\}. The $p$th output sample of the WN is given by [15]:

$$y^{(p)} = \sum_{i=1}^{m} \theta_i \psi(D_i x^{(p)} - B t_i) = \sum_{i=1}^{m} \theta_i \psi^{(p)}_i$$

where $m$ is the number of wavelets (wavelet neurons) in the hidden layer, $\theta_i$ are the weights between the hidden layer and output, $\psi \in L^2(\mathbb{R}^n)$ is the mother wavelet function, $D_i = \text{diag}(d_i)$, $d_i \in \mathbb{R}^n$ is the scale parameter vector of the wavelets, $t_i \in \mathbb{R}^n$ is the shift parameter vector of the wavelets, and $B = \text{diag}(b)$. 

Manuscript received September 26, 2019; revised December 7, 2019; accepted December 8, 2019. Date of publication December 11, 2019; date of current version January 24, 2020. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Demetrio Labate.

A. R. Sadri and S. E. Viswanath are with the Department of Biomedical Engineering, Case Western Reserve University, Cleveland, OH 44106 USA (e-mail: ars329@case.edu; sev21@case.edu).

M. E. Celebi is with the School of Computer Science, University of Central Arkansas, Conway, AR 72035 USA (e-mail: ecelebi@uca.edu).

N. Rahnavard is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816 USA (e-mail: nazanin@eecs.ucf.edu).

Digital Object Identifier 10.1109/LSP.2019.2959219

Authorized licensed use limited to: University of Central Florida. Downloaded on July 15,2020 at 01:11:18 UTC from IEEE Xplore. Restrictions apply.
\( b \in \mathbb{R}^n \) is the discretization factor. Considering the total number of samples, the network output vector \( y \in \mathbb{R}^b \) given in matrix form as:

\[
y = \sum_{i=1}^{m} \theta_i \psi_i = \mathbf{W} \theta
\]

where \( \mathbf{W} = [\psi_1 \ldots \psi_m] \) is called the wavelet matrix (dictionary). The vectors \( \psi_i = [\psi_i^{(1)} \ldots \psi_i^{(P)}]^T \) are regressors (atoms) and \( \theta = [\theta_1 \ldots \theta_m]^T \) is the coefficient vector.

### A. The Wavelet Matrix

According to the multiscaling wavelet frame theorem [16], the wavelet matrix constitutes a multidimensional frame and has significant characteristics by the following elementary lemma and corollary.

**Lemma 1**: If the columns of \( \mathbf{W} = [\psi_1 \ldots \psi_m] \) are frames, with frame bounds \( A > 0, B < \infty \), then inequalities \( AI \leq \mathbf{W} \mathbf{W}^T \leq BI \) hold. For a tight frame \( A = B \) and thus \( \mathbf{W} \mathbf{W}^T = AI \).

**Proof**: See, for example, [17].

**Corollary 1**: The wavelet matrix \( \mathbf{W} \) is full row rank.

**Proof**: The matrix \( \mathbf{W} \) is full row rank if and only if \( \{\forall f \in \mathbb{R}^P. \mathbf{W}f = 0 \implies f = 0\} \). The condition \( \mathbf{W}f = 0 \) implies \( \mathbf{W} \mathbf{W}^T = 0 \), which in turn implies \( f = 0 \) because the \( \mathbf{W} \mathbf{W}^T \) is invertible according to Lemma 1.

### B. The Coefficient Vector

Since the wavelet matrix is full row rank, the USLE extracted from the FGWN has infinitely many solutions. We are interested in seeking its sparsest solution of the coefficient vector. The sparsity of the coefficient vector affects the internal structure of the WN. The sparser the coefficient vector, the less the network’s computational complexity. A WN with too many hidden layer nodes is slower, may cause training to diverge, or lead to overfitting, which would reduce the network generalizability [18]. On the other hand, having too few hidden units, results in large training and generalization errors due to underfitting and high statistical bias [6]. Therefore, we should look for a coefficient vector that has an acceptable error and as much sparsity as possible.

Finding the proper solution of vector \( \theta \) can be cast as a constrained optimization problem as follows:

\[
\min_{\theta} \|\theta\|_0 \quad \text{subject to} \quad \|y - \mathbf{W} \theta\|_2 \leq \epsilon
\]

where \( \epsilon \) is a predefined error tolerance.

Our strategy for solving (3) is based on the non-convex sparse regularization technique. These methods are part of the GNC family and are often significantly slower than greedy algorithms [19]. A fast GNC technique which is based on smoothed \( \ell_0 \) norm (SLO) with reasonable computing time is proposed in [14]. Inspired by the SL0 method, we propose a mathematical framework for finding the sparsest solution of the USLE (2) as the coefficient vector of the SWN.

### III. FINDING A SPARSE SOLUTION

#### A. Non-Convex Regularization

The strategy of the SL0 algorithm is based on the definition of a smoothing parameter \( \sigma \geq 0 \) and approximates the smoothed \( \ell_0 \) norm with a non-convex function \( \|y\|_{\sigma} \) as \( \|\theta\|_0 = \lim_{\sigma \to 0} \|\theta\|_\sigma \). In this way, the sparsity is induced gradually by decreasing the smoothing parameter, so the nonconvexity of the smooth function increases without getting trapped in local minima [20].

The function \( \|\theta\|_{\sigma} : \mathbb{R}^n \to \mathbb{R} \) parameterized by \( \sigma \geq 0 \) is defined as

\[
f(\theta) = \|\theta\|_{\sigma} = \sum_{i=1}^{m} (1 - f_\sigma(\theta_i))
\]

where the one variable function \( f_\sigma(\cdot) \) has the following properties:

- **P1** \( \lim_{\sigma \to 0} f_\sigma(\theta_i) = f(\theta_i) = \{1 \quad \text{if} \quad \theta_i = 0; 0 \quad \text{if} \quad \theta_i \neq 0\} \)
- **P2** \( f_\sigma'(\theta_i) \) is gradient-Lipschitz with constant \( M/\sigma^2 \), where \( M \) is a positive constant. Hence the second derivative of \( f_\sigma(\theta_i) \) is bounded (i.e. \( \forall \theta_i \in \mathbb{R} : |f''(\theta_i)| \leq M/\sigma^2 \)).

With the definition of \( \|\theta\|_{\sigma} \), the sparsest solution of (3) is in the form

\[
\theta^* = \arg\min_{\theta \in C_\epsilon} \left\{ f(\theta) = \|\theta\|_{\sigma} \right\}
\]

where \( C_\epsilon = \{\theta : \|y - \mathbf{W} \theta\|_2 \leq \epsilon\} \).

#### B. The Final Algorithm

The function \( f \) in the form of (4) is gradient-Lipschitz through the following lemma.

**Lemma 2**: If \( f_\sigma(\theta_i) \) is gradient-Lipschitz with constant \( L \) then \( \|\theta\|_{\sigma} \) in the form of (4) is gradient-Lipschitz with constant \( L \).

**Proof**: See [20].

A gradient-Lipschitz function has an elementary but important property which is expressed through the descent lemma as follows:

**Lemma 3 (descent lemma [21])**: Assume that \( f : \text{dom} f \to \mathbb{R} \) is gradient-Lipschitz function with constant \( L > 0 \). Then for any two vectors \( \theta, \theta_k \in \text{dom} f \)

\[
f(\theta) \leq f(\theta_k) + \nabla^T f(\theta_k)(\theta - \theta_k) + \frac{1}{2\gamma} \|\theta - \theta_k\|^2_2
\]

where \( \gamma \in (0, 1/L] \) and \( \text{dom} f \) express the domain of the function \( f \). The right hand side of (6) is called the upper-bound of \( f(\theta) \) at the point \( \theta_k \) and it is shown by \( f(\theta, \theta_k) \). The minimum upper-bound is attained when \( \gamma = 1/L \).

**Proof**: See, for example, [21].

As it stands, \( f(\theta_k, \theta_k) = f(\theta_k) \). Therefore, instead of minimizing \( f \), we can minimize its upper-bound. Thus, the iterative solution algorithm for (5) is \( \theta_{k+1} = \arg\min_{\theta \in C_\epsilon} \{ f(\theta, \theta_k) \} \).

Considering (4), we have

\[
\theta_{k+1} = \arg\min_{\theta \in C_\epsilon} \left\{ \|\theta_k\|_{\sigma} + \nabla^T \|\theta_k\|_{\sigma}(\theta - \theta_k) + \frac{1}{2\gamma} \|\theta - \theta_k\|^2_2 \right\}
\]

equivalently

\[
\theta_{k+1} = \arg\min_{\theta \in C_\epsilon} \frac{1}{2\gamma} \|\theta - \theta_k\|^2_2
\]

where \( \theta_k = \theta - \gamma \nabla \|\theta_k\|_{\sigma} \). So, the final solution to find the sparse solution of USLE (2), which is summarized in Algorithm 1, can be obtained.
Algorithm 1: The Sparse Solution of Coefficient Vector.

Input: $y$, $W$, $M$, $\sigma_0$, $\sigma_{\min}$, $0 < c < 1$, $K$, $\gamma$, $\epsilon$.
Initialization: $\theta = 0$, $\sigma = \sigma_0$.

1: while $\sigma > \sigma_{\min}$ do
2: for $k = 1, 2, \ldots, K$ do
3: $\theta = \theta - \gamma \nabla \|\theta\|_\sigma$
4: $\theta = \arg \min_{\theta \in C_\gamma} \frac{1}{2} \|\theta - \bar{\theta}\|_2^2$
5: end for
6: $\sigma = c \sigma$
7: $\gamma = (\sigma^2 / M) \gamma$
8: end while

Output: $\theta$.

Remark 1: In Algorithm 1, $\sigma_0$, $\sigma_{\min}$, and $c$ are the initial value, the final value, and the reduction factor for $\sigma$, respectively, $K$ is the number of inner-loop iterations, and $\gamma$ is the learning rate.

Remark 2: Since $\bar{f} : C_\gamma \times C_\gamma \rightarrow \mathbb{R}$ satisfies $\bar{f}(\theta, \theta_k) \geq f(\theta)$, $f(\theta, \theta_k) = f(\theta_k)$ for $\theta, \theta_k \in C_\gamma$, $f(\theta, \theta_k)$ is so-called majorization function of $f(\theta)$ [22]. Therefore, our algorithm is a type of majorization-minimization algorithms [21].

Remark 3: According to the proximal operator definition, (8) can be rewritten as $\theta_{k+1} = \text{prox}_{\gamma \|.|_\sigma}(\bar{\theta}_k)$, where $g$ is an indicator function. So, our method can be considered as a proximal method for non-convex optimization [20].

C. Convergence Analysis

We will now assess the bound of the parameter $\gamma$ to guarantee convergence of the iterations in (7) through the following theorem.

Theorem 1: Let $f(\theta) = \|\theta\|_\sigma$. Then, the sequence $\{\theta_k\}$ in (8) converges to a stationary point of $f$. To guarantee convergence, parameter $\gamma$ should satisfy

$$0 < \gamma \leq \frac{2}{M}. \quad (9)$$

Proof: According to (7), the iterations $\theta_{k+1}$ can be written as the associated algorithm

$$\theta_{k+1} = \arg \min_{\theta} \left\{ \nabla^T \|\theta\|_\sigma (\theta - \theta_k) + \frac{1}{2\gamma} \|\theta - \theta_k\|_2^2 \right\}. \quad (10)$$

Since $\theta_{k+1}$ is the minimizer of (10)

$$\nabla^T \|\theta\|_\sigma (\theta_{k+1} - \theta_k) + \frac{1}{2\gamma} \|\theta_{k+1} - \theta_k\|_2^2 \leq 0. \quad (11)$$

On the other hand, by (6) for minimum upper-bound of $f(\theta)$ at the point $\theta_k$, we have

$$\|\theta_{k+1}\|_\sigma \leq \|\theta_k\|_\sigma + \nabla^T \|\theta\|_\sigma (\theta_{k+1} - \theta_k) + \frac{L}{2} \|\theta_{k+1} - \theta_k\|_2^2 \quad (12)$$

where $L$ is the Lipschitz constant of the $\nabla \|\theta\|_\sigma$ and according to Lemma 2, $L = M/\sigma^2$.

Adding (11) and (12) results in

$$f(\theta_{k+1}) \leq f(\theta_k) - \left(1 - \frac{1}{2\gamma} \frac{M}{2\sigma^2}\right) \|\theta_{k+1} - \theta_k\|_2^2 \quad (13)$$

which implies that the sequence $\{f(\theta_k)\}_{k=0}^\infty$ is decreasing if $0 < \gamma \leq \sigma^2 / M$. Since $f$ is bounded from below $(\|\theta\|_\sigma \leq \|\theta_0\|_\sigma)$ and decreasing, we conclude that $\{f(\theta_k)\}_{k=0}^\infty$ converges.

Summing (13) over all $k \geq 0$ leads to

$$\sum_{k=0}^\infty \left\{ \frac{1}{2\gamma} - \frac{M}{2\sigma^2} \right\} \|\theta_{k+1} - \theta_k\|_2^2 \leq f(\theta_0) - f(\theta_\infty). \quad (14)$$

It is clear that right-hand side of (14) is finite and non-negative.

Necessarily, $\theta_{k+1} \rightarrow \theta_k$ and therefore, $\{\theta_k\}_{k=0}^\infty$ converges.

Furthermore, since $\theta_{k+1}$ is the minimizer of (10), we have

$$\nabla \|\theta_k\|_\sigma + \frac{1}{\gamma} (\theta_{k+1} - \theta_k) = 0. \quad (15)$$

Since $\theta_{k+1} \rightarrow \theta_k$, so $\nabla \|\theta_k\|_\sigma \rightarrow 0$. This means that as $k \rightarrow \infty$, $\theta_k \rightarrow \theta^*$ where $\theta^*$ is a stationary point of $f$.

D. Tight Wavelet Frame

At each iteration of the algorithm, the following constrained minimization problem needs to be solved:

$$\min_{\theta} \frac{1}{2\gamma} \|\theta - \theta_k\|_2^2 \quad \text{subject to} \quad \|y - W\theta\|_2 \leq \epsilon \quad (16)$$

where $\bar{\theta} = \gamma \nabla \|\theta\|_\sigma$ and $\epsilon$ denotes the error tolerance. To solve (16), we derive the Lagrangian with multiplier $\lambda$ in the form

$$L(\theta, \lambda) = \frac{1}{2\gamma} \|\theta - \theta_k\|_2^2 + \lambda (\|y - W\theta\|_2^2 - \epsilon^2). \quad (17)$$

Karush-Kuhn-Tucker conditions imply the following optimality conditions:

$$\begin{cases} \theta^* = (I + 2\lambda^* W^T W)^{-1}(\bar{\theta} + 2\lambda^* W^T y) \\ \|y - W\theta^*\|_2 = \epsilon^2 \end{cases} \quad (18)$$

and after substitutions, we obtain the following equation

$$\|y - W(I + 2\lambda^* W^T W)^{-1}(\bar{\theta} + 2\lambda^* W^T y)\|_2^2 = \epsilon^2. \quad (19)$$

Generally there is no closed-form solution for this nonlinear equation, unless $W$ is a tight frame.

According to Lemma 1, if a wavelet matrix is a tight frame then $W W^T = A I$. By applying the matrix inversion lemma, we obtain: $(I + 2\lambda^* W^T W)^{-1} = I - (2\lambda^*/(1 + 2\lambda^*) A) W^T W$, which by combining (19) and (18), leads to

$$\begin{cases} \lambda^* = \frac{1}{2} \max_{\epsilon \in [0, 1]} \left\{ \left(\frac{\|y - W\theta|_2^2}{\epsilon^2} - 2\right) \right\} \\ \theta^* = \bar{\theta} + 2\lambda^* I + 2\lambda^* A W^T (y - W\theta) \end{cases}. \quad (20)$$

Since this approach simplifies the computations, in this study, we consider only tight wavelet frames for the construction of the WN wavelet matrix. It is worth mentioning that, if $\epsilon = 0$ the final solution of the algorithm is given by $\theta^* = \theta_k - \gamma \nabla \|\theta\|_\sigma$. In this situation, the algorithm is indeed a gradient descent with step-size $\gamma$ (similar to SL0).
IV. SIMULATION RESULTS

In WN equation (1), \(d_1 = [a^1, \ldots, a^n]^T, b = [b, \ldots, b]^T\) with \(a > 1\) and \(b > 0\) are considered [16]. One dimensional mother wavelet admissibility theorem for tight wavelet frame [17] states that the frame bound is equal to \(A = 2\pi \int_{0}^{\infty} |\hat{\psi}(\omega)|^2 d\omega\), where \(\hat{\psi}(\omega)\) is the Fourier transform of \(\psi(x)\). Since in the multidimensional case, the wavelet function is the tensor product of one-dimensional mother wavelets [16], the tight frame bound is \(nA\), where \(n\) is the WN input dimensionality. As is customary in the WN literature, we use the Mexican hat as the mother wavelet function in the construction of SWN.

To take advantage of the Mexican hat tight wavelet frame, we are confined to choose \(\{1 < a \leq 2^{0.25}\}\) [17, Ch. 3, p. 71] or \(\{a = 2, 0 < b \leq 0.75\}\) [17, Ch. 3, p. 76]. In our experiments, we used MATLAB 9.4 on a PC with Intel(R) Core(TM) i7 CPU 930 i7 CPU 930 and the frame bound is 7850. 7850 ≤ \(\pi p\) + 0 points are used for training and 1000 ≤ \(\pi p\) + 0 points are used for testing.

As an example, suppose the nonlinear two inputs, two outputs system is given by

\[
\begin{align*}
g_1(p) &= \frac{1}{1 + (y_1(p-1))^2} \left(0.1y_1(p-1) + 0.9u_1(p-2) + 0.1u_2(p-3)\right) \\
g_2(p) &= \frac{1}{1 + (y_2(p-1))^2} \left(0.5y_2(p-1) + 0.3u_1(p) + u_2(p-1)\right)
\end{align*}
\]

(21)

where, the pairs \((u_1(p), y_1(p))\) and \((y_1(p), y_2(p))\) are the input and output samples, respectively. An additive independent and identically distributed (i.i.d.) noise is also considered for both system outputs where the noise term is uniformly distributed in \([-\epsilon, \epsilon]\).

For identifying this system, two SWN with \(n = 3\) inputs and one output is formed. The inputs of the first SWN are \(x_1(p) = [y_1(p-1), y_1(p-2), u_1(p-3)]^T\) and the inputs of the second SWN are \(x_2(p) = [y_2(p-1), u_1(p), u_2(p-1)]^T\). 900 points are used for training the network. Half of them are uniformly distributed on \([-1, 1]\) and the remaining are sinusoids of the form \(1.05 \sin(\pi k/45)\), \(\sigma_0 = 0.5, \sigma_{\text{min}} = 0.05, c = 0.8, \gamma = 0.1\). The algorithm is terminated when it reaches \(K = 15\) or when a given noise level threshold \((\epsilon = 0.1, 0.25)\) is met.

After SWN construction and determination of the coefficient vector, (22) and (23) test signals are used for testing the performance of the SWN models. The performance of the SWN outputs for the test signals are presented in Fig. 1. The SWN performance was evaluated through simulations and compared against several WN models using the same training and testing procedure. The results in terms of the number of network waveforms and root mean square error (RMSE) between the actual and predicted output signals for two different noise levels are given in Table I.

The AWN [15] is trained using the backpropagation algorithm in the publicly available weavenet MATLAB Toolbox [23]. In the FGWNs, instead of using the proposed algorithm, the orthogonal least-squares method [11] and [18] or the D-optimality orthogonal matching pursuit algorithm [24] is applied. It can be seen that the number of the proposed SWN waveforms is much lower than the other methods, while at the same time our models result in considerably smaller RMSE for both system outputs.

Here the sparsity concept is directly fed to the model construction, which provides aparsimonious model with good generalization performance.

V. CONCLUSION

In this letter, we made a novel contribution by looking at wavelet networks from a sparse linear regression point of view and proposed a sparse wavelet network (SWN). In an SWN, the sparsity concept is equivalent to the number of hidden layer neurons which are specified from the sparse solution of a linear regression model. Our sparse solution algorithm is based on \(l_0\) norm approximation with a non-convex gradient-Lipschitz function. The function non-convexity can be controlled by varying the smoothing parameter in each algorithm iteration [14].

The proposed SWN has a solid mathematical foundation with low complexity which can be utilized in practical implementations.

![Fig. 1. Test results of the proposed SWN for the actual and approximated signals: (a) the output \(y_1\) and (b) the output \(y_2\).](image-url)
REFERENCES


