2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Neural Decomposition of Time-Series Data

Luke B. Godfrey
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701

Michael S. Gashler
Department of Computer Science and Computer Engineering
University of Arkansas
Fayetteville, AR 72701

***Abstract*—We present a neural network technique for the analysis and extrapolation of time-series data called Neural Decomposition (ND). Units with a sinusoidal activation function are used to perform a Fourier-like decomposition of training samples into a sum of sinusoids, augmented by units with nonperiodic activation functions to capture linear trends and other nonperiodic components. We show how careful weight initialization can be combined with regularization to form a simple model that generalizes well. Our method generalizes effectively on the Mackey-Glass series, a dataset of unemployment rates as reported by the U.S. Department of Labor Statistics, a time-series of monthly international airline passengers, and an unevenly sampled time-series of oxygen isotope measurements from a cave in north India. We find that ND outperforms popular time-series forecasting techniques including LSTM, echo state networks, (S)ARIMA, and SVR with a radial basis function.**

I. Introduction

The analysis and forecasting of time-series is a challenging problem that continues to be an active area of research. Predictive techniques have been presented for an array of problems, including weather [1], traffic flow [2], seizures [3], sales [4], and others [5], [6], [7]. Because research in this area can be so widely applied, there is great interest in discovering more accurate methods for time-series forecasting.

One approach for analyzing time-series data is to interpret it as a signal and apply the Fourier transform to decompose the data into a sum of sinusoids [8]. Unfortunately, despite the well-established utility of the Fourier transform, it cannot be applied directly to time-series forecasting. Although the signal produced by the Fourier transform exactly reproduces the training samples, it also predicts that the same pattern of samples will repeat indefinitely. Another limitation of the Fourier transform is that it only uses periodic components, and thus cannot accurately model the nonperiodic aspects of a signal, such as a linear trend or nonlinear abnormality.

Another approach to forecasting time-series data is to use a model such as a neural network. Both feedfoward neural networks and recurrent neural networks have been applied to time-series analysis. Feedforward networks, such as Fourier neural networks that are initialized to compute the Fourier transform, have yielded promising results, but have also proven difficult to train [1]. Recurrent networks have tended to perform better at this task, with LSTM networks being among the most popular approaches [9]. Other models perform sinusoidal regression [10], borrowing insights from harmonic analysis methods like the discrete Fourier transform. Regression and extrapolation-based approaches, however, have been largely abandoned in favor of the more successful recurrent models.

We claim that effective generalization can be achieved by regression and extrapolation using a model with two essential properties: (1) it must combine both periodic and nonperiodic components, and (2) it must be able to tune its components as well as the weights used to combine them. We present a neural network technique called Neural Decomposition (ND) that demonstrates this claim. Like the Fourier transform, it decomposes a signal into a sum of constituent parts. Unlike the Fourier transform, however, ND is able to reconstruct a signal that is useful for extrapolating beyond the training samples. ND trains the components into which it decomposes the signal represented by training samples. This enables it to find a simpler set of constituent signals. In contrast to the fast Fourier transform, ND does not require the number of samples to be a power of two, nor does it require that samples be measured at regular intervals. Additionally, ND facilitates the inclusion of nonperiodic components, such as linear or sigmoidal components, to account for trends and nonlinear irregularities in a signal.

In Section IV, we demonstrate that the simple innovations of ND work together to produce significantly improved generalizing accuracy with several problems. We tested with the chaotic Mackey-Glass series, a dataset of unemployment rates as reported by the U.S. Department of Labor Statistics, a time-series of monthly international airline passengers, and an unevenly sampled time-series of oxygen isotope measurements from a cave in north India. We compared against long short-term memory networks (LSTM), echo state networks, autoregressive integrated moving average (ARIMA) models, seasonal ARIMA (SARIMA) models, and support vector regression with a radial basis function (SVR). In all but one case, ND made better predictions than each of the other prediction techniques evaluated; in the excepted case, LSTM and echo state networks performed slightly better than ND.

II. Related Work

A. Models for Time-Series Prediction

Many works have diligently surveyed the existing literature regarding forecasting techniques [11], [12], [13], [14], [15]. Among the most popular statistical approaches are ARIMA models [16]. Seasonal ARIMA (SARIMA) is considered to be the state of the art "classical" time-series approach [2].

978-1-5386-1645-1/17/$31.00 ©2017 IEEE

In the field of machine learning, three high-level classes of techniques are commonly used to forecast time-series data [1]. The first method, perhaps the most common approach, is to train a model to directly forecast future samples based on a sliding window of recently collected samples [14]. The second method is is to train a recurrent neural network [17]. Recurrent models, such as LSTM networks [9], have reported very good results for forecasting time-series. Our model falls into the third category of machine learning techniques: regression-based extrapolation. Our model is more closely related to a subclass of methods in this category, called Fourier neural networks (see Section II-C). Models in the first two categories have already been well-studied, whereas extrapolation with sinusoidal neural networks remains relatively unexplored.

B. Harmonic Analysis

The harmonic analysis of a signal transforms a set of samples from the time domain to the frequency domain. This is useful in time-series prediction because the resulting frequencies can be used to reconstruct the original signal (interpolation) and to forecast values beyond the sampled time window (extrapolation). Harmonic analysis, also known as spectral analysis or spectral density estimation, has been well-studied for decades [18], [19].

Perhaps the most popular method of harmonic analysis is the discrete Fourier transform (DFT). The DFT maps a series of N complex numbers in the time domain to the frequency domain, and its inverse can be applied to map frequency domain values back to the time domain. For real-valued input, the inverse DFT can be sufficiently described as

$$x(t) = \sum_{k=0}^{N/2} R_k \cdot cos(\frac{2\pi k}{N} t) - I_k \cdot sin(\frac{2\pi k}{N} t). \quad (1)$$

Equation 1 is useful as a continuous representation of the real-valued discrete input. Because it perfectly passes through the input samples, one might naively expect this function to be a good basis for generalization. However, the DFT cannot effectively mode nonperiodic components of a signal, nor can it form a simple model for series that are not periodic at N.

Because of these limitations, other approaches to the harmonic analysis of time-series have been proposed. Some of these other approaches perform sinusoidal regression to determine frequencies that better represent the periodicity of the sampled signal [10]. However, these approaches have largely been abandoned in favor of the fast Fourier transform, which allows the DFT to be calculated in $log(n)$ time. Like these less popular approaches, our approach uses regression to find better frequencies. Our results show that this antiquiated regression-based approach, combined with nonlinear components, is able to outperform state of the art methods for time-series prediction.

C. Fourier Neural Networks

Use of the Fourier transform in neural networks has already been explored in various contexts [20], [21]. The term *Fourier neural network* has been used to refer to neural networks that use a Fourier-like neuron [22], that use the Fourier transform of some data as input [23], or that use the Fourier transform of some data as weights [1]. Our work is not technically a Fourier neural network, but of these three types, our approach most closely resembles the third.

Silvescu provided a model for a Fourier-like activation function for neurons in neural networks [22]. His model utilizes every unit to form DFT-like output for its inputs. He notes that by using gradient descent to train sinusoid frequencies, the network is able to learn "exact frequency information" as opposed to the "statistical information" provided by the DFT. Our approach also trains the frequencies of neurons with a sinusoidal activation function.

Gashler and Ashmore presented a technique that used the fast Fourier transform (FFT) to approximate the DFT, then used the obtained values to initialize the sinusoid weights of a neural network that mixed sinusoidal, linear, and softplus activation functions [1]. Because this initialization used sinusoid units to model nonperiodic components of the data, their model was designed to heavily regularize sinusoid weights so that as the network was trained, it gave preference to weights associated with nonperiodic units and shifted the weights from the sinusoid units to the linear and softplus units. Use of the FFT required their input size to be a power of two, and their trained models were slightly out of phase with their validation data. However, they were able to generalize well for certain problems. Our approach is similar, except that we do not use the Fourier transform to initialize any weights.

III. Approach

In this section, we describe Neural Decomposition (ND), a neural network technique for the analysis and extrapolation of time-series data.

A. High-Level Description

We use a DFT-like model with two simple but important innovations. First, we allow sinusoid frequencies to be trained. Second, we augment the sinusoids with a nonperiodic function. The use of sinusoids allows our model to fit to periodic data, the ability to train the frequencies allows our model to learn the true period of a signal, and the augmentation function enables our model to forecast time-series that are made up of both periodic and nonperiodic components.

Our model is defined as follows. Let each a_k represent an amplitude, each w_k represent a frequency, and each ϕ_k represent a phase shift. Let t refer to time, and let $g(t)$ be an augmentation function that represents the nonperiodic components of the signal. Our model is defined as

$$x(t) = \sum_{k=1}^{N} \big(a_k \cdot sin(w_k t + \phi_k)\big) + g(t). \quad (2)$$

Note that the lower index of the sum has changed from $k = 0$ in the DFT to $k = 1$ in our model. This is because ND can account for bias in the augmentation function $g(t)$. Therefore, only N sinusoids are required rather than $N + 2$.

B. Topology

We use a feedforward artificial neural network as the basis of our model. For an input of size N, the neural network is initialized with two layers: $1 \rightarrow m$ and $m \rightarrow 1$, where $m = N + |g(t)|$ and $|g(t)|$ denotes the number of nodes required by $g(t)$. The first N nodes in the hidden layer have the sinusoid activation function, $sin(t)$, and the rest of the nodes in the hidden layer have other activation functions to compute $g(t)$.

The augmentation function $g(t)$ can be made up of any number of nodes with one or more activation functions. For example, it could be made up of linear units for learning trends and sigmoidal units to fit nonperiodic, nonlinear irregularities. Gashler and Ashmore have suggested that softplus units may yield better generalizing predictions compared to standard sigmoidal units [1]. In our experiments, we used a combination of linear, softplus, and sigmoidal nodes for $g(t)$. The network tended to only use a single linear node, which may suggest that the primary benefit of the augmentation function is that it can model linear trends in the data. Softplus and sigmoidal units tended to be used very little or not at all by the network in the problems we tested, but intuitively it seems that nonlinear activation functions could be useful in some cases.

C. Weight Initialization

The weights of the neural network are initialized as follows. Let each of the N sinusoid nodes in the hidden layer, indexed as k for $0 \leq k < N/2$, have a weight w_k and bias ϕ_k. Each w_k represents a frequency and is initialized to $2\pi\lfloor k/2 \rfloor$. Each ϕ_k represents a phase shift. For each even value of k, ϕ_k is set to $\pi/2$ to transform $sin(t + \phi_k)$ to $cos(t)$. For each odd value of k, ϕ_k is set to π to transform $sin(t + \phi_k)$ to $-sin(t)$. A careful comparison of these initialized weights with Equation 1 shows that these are identical to the frequencies and phase shifts used by the DFT, except for a missing $1/N$ term in each frequency, which is absorbed in the input preprocessing step (see Section III-D).

All weights feeding into the output unit are set to small random values. At the beginning of training, therefore, the model will predict something like a flat line centered at zero. As training progresses, the neural network will learn how to combine the hidden layer units to fit the training data.

Weights in the hidden layer associated with the augmentation function are initialized to approximate the identity function. For example, in $g(t) = wt + b$, w is randomly perturbed from 1 and b is randomly perturbed near 0. Because the output layer will learn how to use each unit in the hidden layer, it is important that each unit be initialized in this way.

D. Input Preprocessing

Before training begins, we preprocess the input data to facilitate learning and prevent the model from falling into a local optimum. First, we normalize the time associated with each sample so that the training data lies between 0 (inclusive) and 1 (exclusive) on the time axis. If there is no explicit time, equally spaced values between 0 and 1 are assigned to each sample in order. Predicted data points will have a time value greater than or equal to 1 by this new scale. Second, we normalize the values of each input sample so that all training data is between 0 and 10 on the y axis.

This preprocessing step serves two purposes. First, it absorbs the $1/N$ term in the frequencies by transforming t into t/N, which is why we were able to omit the $1/N$ term from our frequencies in the weight initialization step. Second, it ensures that the data is appropriately scaled so that the neural network can learn efficiently. If the data is scaled too large, training will be slow and susceptible to local optima. If the data is scaled too small, on the other hand, the learning rate of the machine will cause training to diverge and only use linear units and low frequency sinusoids.

In some cases, it is also appropriate to pass the input data through a filter. For example, financial time-series data is commonly passed through a logarithmic filter before being presented for training, and outputs from the model are then exponentiated to obtain predictions. We use this filtering method in two of our experiments in Section IV.

E. Regularization

Prior to each sample presentation, we apply regularization on the output layer of the neural network. Even though we do not initialize sinusoid amplitudes using the DFT, the network is quickly able to learn how to use the initialized frequencies to closely fit the input samples. Without regularizing the output layer, training halts as soon as the model fits the input samples, because the measurable error is near zero. By relaxing the learned weights, regularization allows our model to redistribute its weights over time. We find that regularization amount is especially important; too much prevented our model from learning, but too little caused our model to fall into local optima. In our experiments, setting the regularization term to 10^{-2} avoided both of these potential pitfalls.

Another important function of regularization in ND is to promote sparsity in the network, so that the redistribution of weights by stochastic gradient descent produces as simple a model as the input samples allow. We use L^1 regularization for this reason. Usually, the trained model does not require all N sinusoid nodes in order to generalize well, and this type of regularization enables the network to automatically discard unnecessary nodes by driving their amplitudes to zero.

It is worth noting that we only apply regularization to the output layer of the neural network. Any regularization that might occur in the hidden layer would adjust sinusoid frequencies before the output layer could learn sinusoid amplitudes. By allowing weights in the hidden layer to change without regularization, the network has the capacity to adjust frequencies but is not required to do so.

IV. Validation

In this section, we report results that validate the effectiveness of Neural Decomposition. In each of these experiments, we used an ND model with an augmentation function made up of ten linear units, ten softplus units, and ten sigmoidal units. It is worth noting that $g(t)$ is under no constraint to consist only

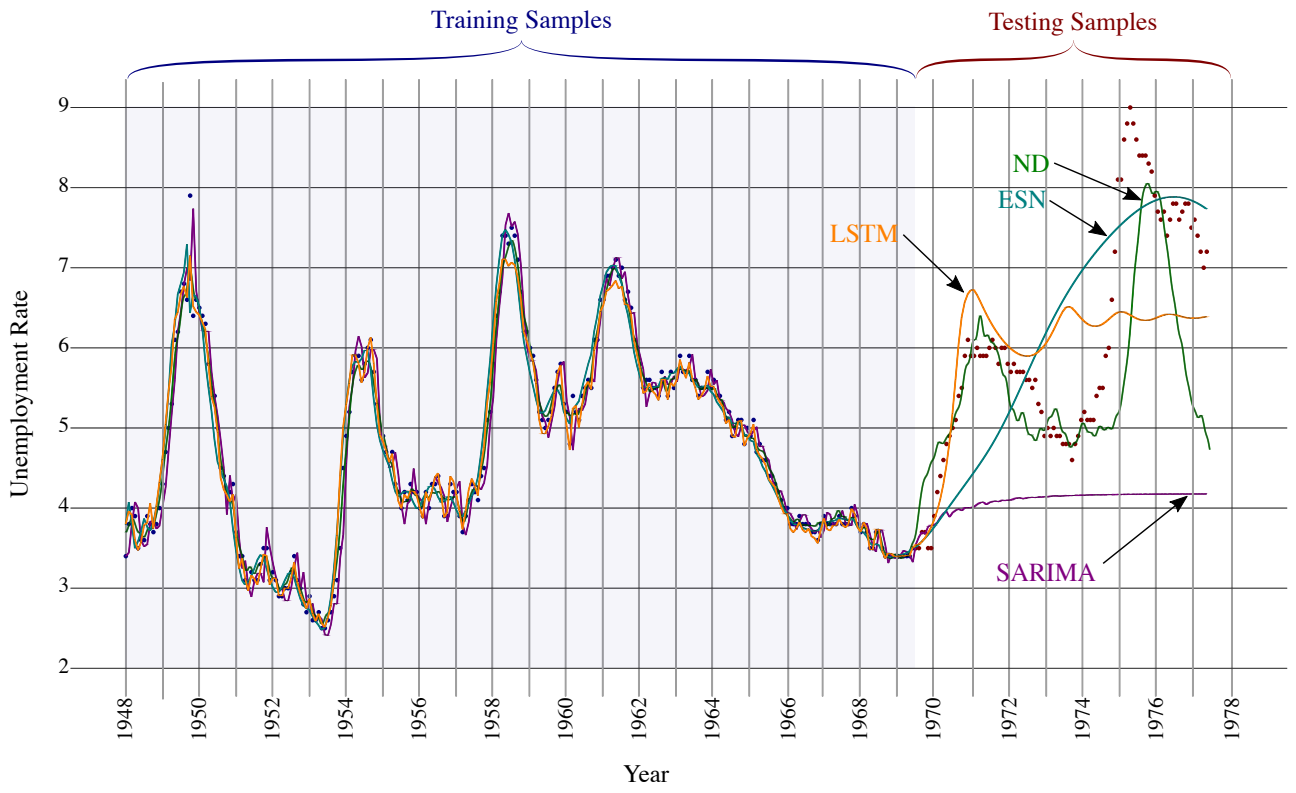

Fig. 1. A comparison of the four best predictive models on the monthly unemployment rate in the US. Blue points represent training samples from January 1948 to June 1969 and red points represent testing samples from July 1969 to December 1977. ND, shown in green, successfully predicts both surges in unemployment that followed the training samples.

TABLE I
ROOT MEAN SQUARE ERROR (RMSE) ON THE VALIDATION PROBLEMS FOR ARIMA, SARIMA, SVR, ESN, LSTM, AND ND. BEST RESULT (SMALLEST ERROR) FOR EACH PROBLEM IS SHOWN IN **BOLD**.

Model	Labor	Airline	Speleothem
ARIMA	2.97	75.32	N/A
SARIMA	2.41	67.54	N/A
SVR	2.18	209.57	1.078
ESN	**1.09**	63.50	N/A
LSTM	1.14	93.61	N/A
ND	**1.09**	**45.03**	**0.214**

of these units; it could include other activation functions or only contain a single linear node to capture trend information. We use a regularization term of 10^{-2} and a learning rate of 10^{-3} in every experiment to demonstrate the robustness of our approach; we did not tune these meta-parameters for each experiment.

In our experiments, we compare ND with LSTM, ESN, ARIMA, SARIMA, and SVR. We used PyBrain's implementation of LSTM networks [25] with one input neuron, one output neuron, and one hidden layer. We implemented a grid-search to find the best hidden layer size for the LSTM network for each problem and used PyBrain's RPROP- algorithm to train the network. We used Lukoševičius' implementation of ESN [26] and implemented a grid-search to find the best parameters for each problem. We used the R language implementation for ARIMA, SARIMA, and SVR [27]. For the ARIMA models, we used a variation of the `auto.arima` method that performs a grid-search to find the best parameters for each problem. For SVR, we used the `tune.svm` method, which also performs a grid-search for each problem. Although these methods select the best models based on the amount of error calculated using the training samples, the grid-search is a very slow process. With ND, no problem-specific parameter tuning was performed.

In each figure, the blue points in the shaded region represent training samples and the red points represent withheld testing samples. The curves on the graph represent the predictions made by the four models that made the most accurate predictions (only two models are shown in the fourth experiment because only two models could be applied to an irregularly sampled time-series). The actual error for each model's prediction is reported for all experiments and all models in Table I.

The LSTM network tended to fall into local optima, and was thus extremely sensitive to the random seed. Running the same experiment with LSTM using a different random seed yielded very different results. In each experiment, therefore, we tried the LSTM model 100 times for each different topology tested in our grid-search and selected the result with the highest accuracy to present for comparison with ND. Conversely, ND

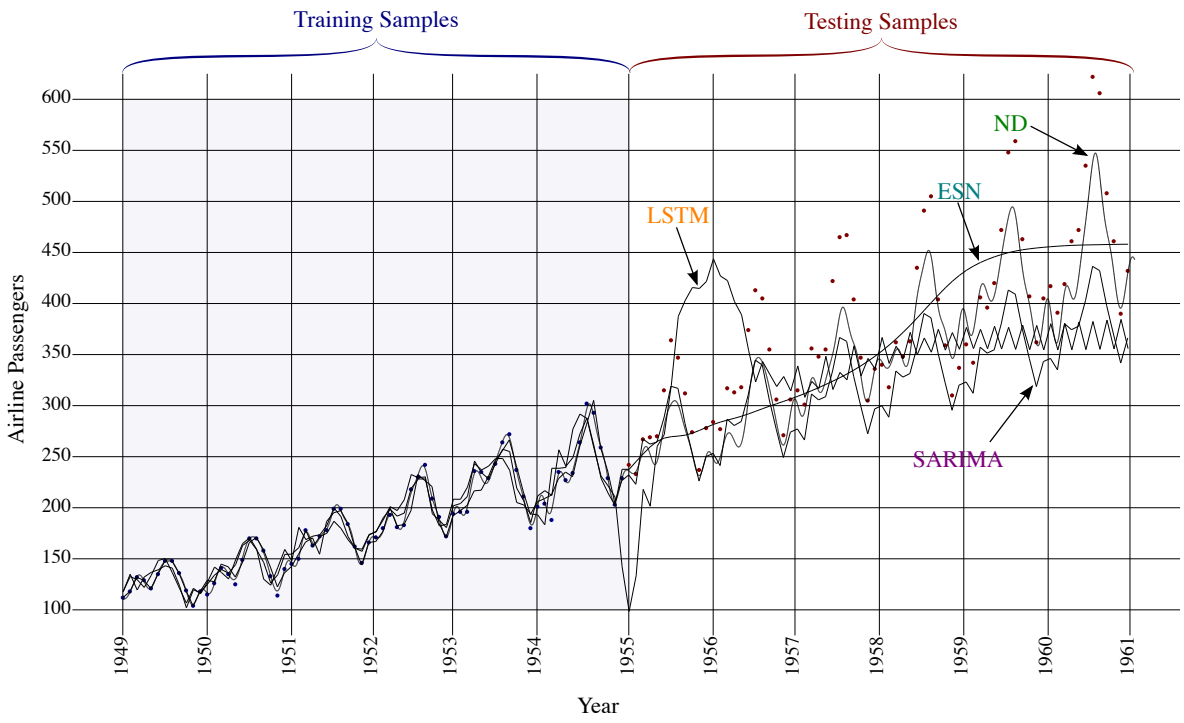

Fig. 2. A comparison of the four best predictive models on monthly totals of international airline passengers from January 1949 to December 1960 [24]. Blue points represent the 72 training samples from January 1949 to December 1954 and red points represent the 72 testing samples from January 1955 to December 1960. ND, shown in green, learns the trend, shape, and growth better than the other compared models.

consistently made approximately identical predictions when run multiple times, regardless of the random seed.

In our first experiment, we demonstrated the effectiveness of ND on real-world data compared to widely used techniques in time-series analysis and forecasting. We trained our model on the unemployment rate from 1948 to 1969 as reported by the U.S. Bureau of Labor Statistics, and predicted the unemployment rate from 1969 to 1977. These results are shown in Figure 1. Blue points on the left represent the 258 training samples from January 1948 to June 1969, and red points on the right represent the 96 testing samples from July 1969 to December 1977. The four curves represent predictions made by ND (green), LSTM (orange), ESN (cyan), and SARIMA (magenta); ARIMA and SVR yielded poorer predictions and are therefore omitted from the figure. Grid-search found ARIMA(3,1,2) and ARIMA(1,1,2)(1,0,1)[12] for the ARIMA and SARIMA models, respectively. ARIMA, not shown, did not predict the significant rise in unemployment. SARIMA, shown in magenta, did correctly predict a rise in unemployment, but underestimated its magnitude, and did not predict the shape of the data well. SVR, not shown, correctly predicted that unemployment would rise, then fall again. However, it also underestimated the magnitude. ESN, shown in cyan, predicted a reasonable mean value for the general increase in unemployment, but failed to capture the dynamics of the actual data. The best LSTM network topology, found by grid-search, had a hidden layer with 16 neurons. LSTM, shown in orange, predicted the first peak in the data, but leveled off to predict only the mean. ND, shown in green, successfully predicted both the depth and approximate shape of the surge in unemployment. Furthermore, it correctly anticipated another surge in unemployment that followed.

Our second experiment demonstrates the versatility of Neural Decomposition by applying to another real-world dataset: monthly totals of international airline passengers as reported by Chatfield [24]. We use the first six years of data (72 samples) from January 1949 to December 1954 as training data, and the remaining six years of data (72 samples) from January 1955 to December 1960 as testing data. The training data is preprocessed through a $log(x)$ filter and the outputs are exponentiated to obtain the final predictions. As in the first experiment, we compare our model with LSTM, ESN, ARIMA, SARIMA, and SVR. The predictions of the four most accurate models (ND, LSTM, ESN, and SARIMA) are shown in Figure 2; ARIMA and SVR yielded poorer predictions and are therefore omitted from the figure. SVR, not shown, predicts a flat line after the first few time steps and generalizes the worst out of the four predictive models. The ARIMA model found by grid-search was ARIMA(2,1,3). ARIMA, not shown, was able to learn the trend, but failed to capture any of the dynamics of the signal. Grid-search found ARIMA(1,0,0)(1,1,0)[12] for the SARIMA model. Both SARIMA (shown in magenta) and ND (shown in green) are able to accurately predict the shape of the future signal, but ND performs better. Unlike SARIMA, ND learns that the periodic component gets bigger over time. ESN, shown in

cyan, performs similarly to the ARIMA model, only predicting the trend and failing to capture seasonal variations. The LSTM network, with a hidden layer of size 64 found by grid-search, failed to capture any meaningful seasonality in the training data. Instead, LSTM immediately predicted a valley and a peak that did not actually occur, followed by a poor estimation of the mean.

Our third experiment demonstrates that ND can be used on irregularly sampled time-series. We use a series of oxygen isotope readings in speleothems in a cave in India from 1489 AD to 1839 AD as reported by Sinha et. al [28]. Because the time intervals between adjacent samples is not constant (the interval is about 1.5 years on average, but fluctuates between 0.5 and 2.0 years), only ND and SVR models can be applied. ARIMA, SARIMA, ESN, and LSTM cannot be applied to irregular time-series because they assume a constant time interval between adjacent samples; these five models are therefore not included in this experiment. We used 250 training samples from July 1489 to April 1744 and tested on 132 testing samples from August 1744 to December 1839. SVR predicts a steep drop in value that does not exist in the testing data, while ND accurately predicts the general shape of the testing data. Due to space contraints, we do not include a plot of this experiment.

Table I presents an empirical evaluation of each model for the three real-world experiments. Using the RMSE metric, we compare Neural Decomposition to ARIMA, SARIMA, SVR with a radial basis function, ESN, and LSTM. We found that on the unemployment rate problem (Figure 1), ND yielded the best model, followed by LSTM and ESN. On the airline problem (Figure 2), ND performed significantly better than all of the other approaches. On the oxygen isotope problem, ND outperformed SVR, which was the only other model that could be applied to the irregular time-series. In each problem, the accuracy of the best algorithm is shown in bold.

V. Conclusion

We presented Neural Decomposition, a neural network technique for time-series forecasting. Our method decomposes a set of training samples into a sum of sinusoids augmented with additional components to enable our model to generalize and extrapolate beyond the input set. Each component of the resulting signal is trained so that ND can find a simpler set of constituent signals. ND uses careful initialization, input preprocessing, and regularization to facilitate the training process. We showed results that demonstrate that our approach is superior to popular techniques LSTM, ESN, ARIMA, SARIMA, and SVR in several cases, including unemployment rate, airline passengers, and an unevenly sampled time-series of oxygen isotopes.

References

[1] M. S. Gashler and S. C. Ashmore, "Training deep fourier neural networks to fit time-series data," in *Intelligent Computing in Bioinformatics - 10th International Conference, ICIC 2014, Taiyuan, China, August 3-6, 2014. Proceedings*, 2014, pp. 44–55.

[2] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, March 2013, pp. 871–882.

[3] C. E. Elger and K. Lehnertz, "Seizure prediction by non-linear time series analysis of brain electrical activity," *European Journal of Neuroscience*, vol. 10, no. 2, pp. 786–789, February 1998.

[4] T.-M. Choi, Y. Yu, and K.-F. Au, "A hybrid sarima wavelet transform method for sales forecasting," *Decision Support Systems*, vol. 51, no. 1, pp. 130–140, 2011.

[5] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting," *Omega*, vol. 29, no. 4, pp. 309–317, August 2001.

[6] K. jae Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1-2, pp. 307–319, September 2003.

[7] S.-M. Chen, "Forecasting enrollments based on fuzzy time series," *Fuzzy Sets and Systems*, vol. 81, no. 3, pp. 311–319, August 1996.

[8] P. Bloomfield, *Fourier analysis of time series: an introduction*. John Wiley & Sons, 2004.

[9] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[10] P. Vanicek, "Approximate spectral analysis by least-squares fit." *Astrophysics and Space Science*, vol. 4, no. 4, pp. 387–391, 1969.

[11] G. Dorffner, "Neural networks for time series processing," in *Neural Network World*. Citeseer, 1996.

[12] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996.

[13] C. Chatfield, *Time-series forecasting*. CRC Press, 2000.

[14] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems*, vol. 31, no. 1-3, pp. 91–103, 2001.

[15] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International journal of forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[16] G. P. Zhang, "Time series forecasting using a hybrid arima and neural network model," *Neurocomputing*, vol. 50, pp. 159–175, January 2003.

[17] O. Nerrand, P. Roussel-Ragot, D. Urbani, L. Personnaz, and G. Dreyfus, "Training recurrent neural networks: Why and how ? an illustration in dynamical process modeling." 1994.

[18] E. M. Stein and T. S. Murphy, *Harmonic analysis: real-variable methods, orthogonality, and oscillatory integrals*. Princeton University Press, 1993, vol. 3.

[19] B. G. Quinn, "Estimating the number of terms in a sinusoidal regression," *Journal of time series analysis*, vol. 10, no. 1, pp. 71–75, 1989.

[20] R. Hecht-Nielsen, "Theory of the backpropagation neural network," *International Joint Conference on Neural Networks*, 1989.

[21] W. Zuo, Y. Zhu, and L. Cai, "Fourier-neural-network-based learning control for a class of nonlinear systems with flexible components," *IEEE Transactions on Neural Networks*, 2009.

[22] A. Silvescu, "Fourier neural networks," in *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, vol. 1, 1999, pp. 488–491.

[23] K. Minami, H. Nakajima, and T. Toyoshima, "Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network," *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING*, vol. 46, no. 2, February 1999.

[24] C. Chatfield, *The Analysis of Time Series: An Introduction*, 6th ed. Chapman and Hall/CRC, July 2003.

[25] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "PyBrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.

[26] M. Lukoševičius, *A practical guide to applying echo state networks*, 2nd ed., ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7700, pp. 659–686.

[27] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2015.

[28] A. Sinha, G. Kathayat, H. Cheng, S. F. M. Breitenbach, M. Berkelhammer, M. Mudelsee, J. Biswas, and R. L. Edwards, "Trends and oscillations in the indian summer monsoon rainfall over the last two millennia," *Nat Commun*, vol. 6, 02 2015.