Neural Probabilistic Forecasting of Symbolic Sequences With Long Short-Term Memory

Michael Hauser
Department of Mechanical Engineering,
The Pennsylvania State University,
University Park, PA 16802
e-mail: mzh190@psu.edu

Yiwei Fu
Department of Mechanical Engineering,
The Pennsylvania State University,
University Park, PA 16802
e-mail: yxf118@psu.edu

Shashi Phoha
Applied Research Laboratory,
The Pennsylvania State University,
University Park, PA 16802
e-mail: sxp26@arl.psu.edu

Asok Ray1
Professor
Fellow ASME
Department of Mechanical Engineering,
The Pennsylvania State University,
University Park, PA 16802
e-mail: axr2@psu.edu

This paper makes use of long short-term memory (LSTM) neural networks for forecasting probability distributions of time series in terms of discrete symbols that are quantized from real-valued data. The developed framework formulates the forecasting problem into a probabilistic paradigm as \( h_\Theta: X \times Y \rightarrow [0, 1] \) such that \( \sum_{y \in Y} h_\Theta(x, y) = 1 \), where \( X \) is the finite-dimensional state space, \( Y \) is the symbol alphabet, and \( \Theta \) is the set of model parameters. The proposed method is different from standard formulations (e.g., autoregressive moving average (ARMA)) of time series modeling. The main advantage of formulating the problem in the symbolic setting is that density predictions are obtained without any significantly restrictive assumptions (e.g., second-order statistics). The efficacy of the proposed method has been demonstrated by forecasting probability distributions on chaotic time series data collected from a laboratory-scale experimental apparatus. Three neural architectures are compared, each with 100 different combinations of symbol-alphabet size and forecast length, resulting in a comprehensive evaluation of their relative performances.

[DOI: 10.1115/1.4039281]

1 Introduction

Time series forecasting is a well-studied problem across a diverse range of fields [1]. From the perspectives of dynamical systems, accurate forecasting of future states can be implemented within a feedback mechanism in the case where control actions can be used to drive the system to a desired state or away from an undesired state. To this end, linear stationary forecasting models [1] have been widely used. For example, autoregressive moving average (ARMA) models provide a standard benchmark for linear stationary time series forecasting; if the time series is nonstationary, then autoregressive integrated moving average models [1] are used to deal with these nonstationary processes.

Previous work has shown that even feedforward (FF) neural networks significantly outperform the linear ARMA model [2], which provides a valuable baseline for time series forecasting. In that work, a feedforward neural network was used to forecast symbol sequences derived from an experimental combustion apparatus. If the ARMA model made a forecast prediction in the same bin as the true trajectory, then it was deemed a correct prediction, otherwise it was incorrect. Under these conditions the simple feedforward neural network significantly outperformed ARMA. Additionally, it was shown in Ref. [2] that feed forward neural symbolic forecasting outperforms taking an expectation over symbol centroids to yield a mean trajectory, which is analogous to a nonlinear neural network ARMA model. The current paper is built upon the previous work [2] by systematically evaluating the relative performance of different neural architectures.

To overcome the limitations of linear models, forecasting time series with neural networks has been reviewed by Zhang et al. [3]. Neural networks can be made to be highly nonlinear, and with even a single hidden layer they have the ability to approximate any Borel measurable function [4]. Additionally, it is desirable to replace these single-valued hard predictions by forecasting probability distributions over symbolic states [5,6].

A forecasting paradigm requires evaluation of both a predicted value and the bounds on its accuracy [7]. Casting the problem in a symbolic-probabilistic setting provides a wide range of generalities; for example, symbols can be interpreted as emanating from a state space. Furthermore, they facilitate probabilistic predictions without prior assumptions on data distributions (e.g., second-order statistics). However, they lack the fidelity that comes with deterministic predictions, although this shortcoming can be partially alleviated by taking an expectation over the prediction space.

Statistical characterizations of symbol sequences have been extensively investigated within the framework of probabilistic finite state automata [8,9]. Within this framework, symbol emission probabilities are estimated, given a current state of the system. Estimation of this symbol emission probability matrix has been cast in a Bayesian framework as well [10].

This paper formulates the probabilistic forecasting problem in a classification sense by using the concept of deep neural networks such that, conditioned on its history, the desired output is a probability over symbols; the symbols partition the range space and each symbol corresponds to a class. This procedure is significantly different from a deterministic regression of the mean and its associated quantiles.

The technical approach of the proposed method is to estimate symbol emission probabilities largely analogously to what is done in finite state automata with stochastic symbolic emissions from given states; however, it does so with deep neural networks. In contrast to a \( D \)-Markov model [11,12] that generates probabilities of emitting certain symbols given a finite history symbol sequence of length \( D \), the proposed model generates symbol emission probabilities based on a finite history of time series of length \( D \). The proposed model is also able to identify nonlinear relationships between past history and future symbols in the setting of dynamic data-driven application systems [13]. The primary contributions of this paper are in the formulation of a novel approach to probabilistic forecasting, namely, symbolic probabilistic forecasting, not in how to train neural networks.

The paper is organized in five sections including the present section. Section 2 introduces the mathematical preliminaries needed for algorithm development, with special attention paid to the long short-term memory (LSTM) neural network model. Section 3 describes the technical details of the proposed neural probabilistic forecasting algorithm. Section 4 presents the results of
algorithm validation on experimental data, collected from a laboratory-scale combustion apparatus. Section 5 summarizes the contents and concludes the paper. Additionally, Sec. 5 suggests directions of future research to test the neural network models on vector time series data.

2 Mathematical Preliminaries

This section succinctly presents the key mathematical concepts that are necessary to develop the algorithms.

2.1 Probabilistic Formulation. Neural networks are a general class of algorithms that have nested compositions of affine transformations followed by a simple nonlinearity. They can be either strictly FF, or include some type of feedback mechanism. Neural networks with feedback are called recurrent and are usually preferred means of modeling temporal dynamics within a neural network framework.

Classification neural networks are mappings $h_{\theta}: X \times Y \rightarrow \{0, 1\}$ such that $\sum_{y \in Y} h_{\theta}(x, y) = 1$, which is usually interpreted as $h_{\theta}(x, y) := p(Y = y | X = x)$, where $X$ is the finite dimensional state space, $Y$ is the finite space of labels (i.e., symbol alphabet) and $\Theta = \{(W^l, b^l)\}_{l=0}^{L-1}$ is the parameterization of architecture class $H$ such that $h_{\theta} \in H$. The notation $h_{\theta}(x) := h_{\theta}(x, \cdot) := p(Y | X = x)$ will be used to represent the output distribution.

Classification neural networks are trained by minimizing a loss function $l(h_{\theta}, q)$, where $q: X \times Y \rightarrow \{0, 1\}$ such that $\sum_{y \in Y} q(x, y) = 1$ is the true classification model. It is noted that the true distribution $q$ may not be an element of $H$, i.e., $q \notin H$, but because neural network models are very flexible, they may still well approximate $q$.

2.2 Long Short-Term Memory. Long short-term memory (LSTM) networks [18,19] are recurrent neural networks with a specific architecture in the hidden space which allows the network to act as if it has a memory unit it can read from and write to. Figure 1 presents a schematic diagram of the LSTM neural network structure for understanding how the different data elements are related-to and operate-on each other.

The LSTM has an input $x_t$, an output $h_t$ and a cell state $C_t$ that acts as the memory and is the central component of the LSTM. It is noted that the sigmoid function $\sigma(z) := 1/(1 + \exp(-z))$ has a range of 0 to 1, and the hyperbolic tangent function $\tanh(z) := (\exp(z) - \exp(-z))/\exp(z) + \exp(-z)$ has range $-1$ to $1$.

The candidate cell state $C_t$, its weightings $i_t$ and the forget gate activations $f_t$ are computed as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$C_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

where $W$'s and $U$'s are appropriate weight matrices and the $b$'s are bias vectors.

With $*$ defined to be element-wise multiplication, the updated cell state $C_t$ is then found as

$$C_t = i_t \ast C_t + f_t \ast C_{t-1}$$

and the output is obtained as a weighted version of the cell state

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \ast \tanh(C_t)$$

Because Eqs. (1)–(3) and (5) depend only on $x_t$, $h_t$ and $h_{t-1}$, their arguments are computed in parallel.

2.3 Time Series Symbolization. The procedure of time series symbolization is built by constructing a partition of the time series feature space, defining mutually exclusive and exhaustive regions over the feature space. Once a partition is defined over the feature space, the data are symbolized by mapping each data point to a corresponding symbol, which is uniquely identified with that partition region. With $K$ symbols, the symbol set is defined to be $Y := \{y_1, y_2, \ldots, y_K\}$.

For one-dimensional time series, two commonly used partitioning schemes are the uniform partitioning and the maximum entropy partitioning [17]. For general multidimensional time series, the right column clustering, Gaussian mixture models, or other unsupervised clustering techniques can be used [18]. This work uses uniform partitioning.

3 Algorithm Development

This section outlines an algorithm for neural probabilistic forecasting of symbol sequences. Given a time series interval up to the present $\{x(t-t')\}_{t'=0}^{T}$, of length $l$, the algorithm predicts the probability of the symbol $y_k$ at a time instant $t+T$ as $p(Y(t+T) = y_k | X(t) = \{x(t-t')\}_{t'=0}^{T})$. The data can, thus, be organized into input–output pairs as follows:

$$D := \{(x(n-t'), y(n+T))\}_{n=0}^{N-l-1}$$

where $I$ is the memory length of the time series obtained from the first minima of the mutual information with a time-shifted version of itself [19], $N$ is the number of data samples, and $T$ is the future forecasting time-step. It is noted that $l \ll N$. The dataset is carefully organized so that the experiments conducted in Sec. 4.2 use exactly the same conditions for the different neural architectures at a given forecast-length and number-of-symbols pair. This data organization ensures fairness in generating exactly the same conditions for all neural architectures by comparing the plotted true trajectory for a fixed pair.

3.1 Mathematical Formulation. As described in Sec. 2.1, a neural network defines a mapping: $h_{\theta}: X \times Y \rightarrow \{0, 1\}$ such that $\sum_{y \in Y} h_{\theta}(x, y) = 1$, where $X$ and $Y$ are the input and label spaces, and $\Theta = \{(W^l, b^l)\}_{l=0}^{L-1}$ is the parameterization. Because we are dealing with time series of the input and output spaces, the mapping used here is defined as $h_{\theta}: (X \times T_1) \times (Y \times T_2) \rightarrow \{0, 1\}$, where $X$ is the input space, $Y$ is the symbol alphabet set, and $T_1$ and $T_2$ are the forecast shifted time indices of $X$ and $Y$, respectively. The definition of $h_{\theta}$ is as follows:

$$h_{\theta}(\{x(t-t')\}_{t'=0}^{T}) := p(Y(t+T) | X = \{x(t-t')\}_{t'=0}^{T})$$

![Fig. 1 Schematic of the LSTM neural network structure. The nonlinear activations, $\sigma$ and $\tanh$, act elementwise on their respective input vectors. Similarly, multiplication and addition blocks operate on their input pairs elementwise.](image-url)
The best model is taken to be \[ \arg\min_{h_0} L(\Theta; D) \], which is trained by using Adadelta [22] in backpropagation over ten epochs with a batch size of 100. The rationale for using Adadelta is delineated in the following:

1. Adadelta automatically updates the learning rate along each dimension. In this sense, it is more user-friendly than standard stochastic-gradient-descent methods that often require the user to manually tune the learning rate.
2. Adadelta is an extension of Adagrad [23], but with the goal to ensure that the learning rate does not monotonically decrease to zero as the number of updates goes to infinity, as is the case with Adagrad.
3. Root-mean-square (RMS) propagation has been tried in preliminary testing as the third update rule, which is an unpublished technique used in Geoff Hinton’s Coursera lecture series on machine learning. Root-mean-square propagation yielded similar results as Adadelta.
4. While stochastic gradient descent has been used in preliminary testing, Adadelta is consistently found to yield better minima in Eq. (11).

\[ L(\Theta; D) := E_{(x,y)}[l(h_0(x,y), q(x,y))] \]  

The forecasting problem is formulated within a probabilistic framework in terms of the objective function that is chosen as the cross entropy between the true probability distribution \( q(x,y) \) such that, \( \sum_y q(x,y) = 1 \), and the model’s symbol emission probability distribution \( h_0(x,y) = p(\mathcal{Y} = y | X = x) \). The corresponding loss function is

\[ l(h_0(x), q(x)) := - \sum_{k=1}^{K} q(x,y_k) \log(h_0(x,y_k)) \]  

where for brevity, \( x \equiv \{x(t-t')\}_{t \in t_0} \) and \( y_k \) is realized at time \( t + T \). Note that there are \( K \) symbols in the symbol set.

The cross entropy between two probability distributions \( q \) and \( h_0 \) over the same sample space is intuitively understood as the average number of bits required to identify a realization drawn from the sample space if the coding scheme is optimized for an approximated distribution \( h_0 \), as opposed to the true distribution \( q \) [20]. Thus minimizing the cross entropy between these two distributions is a way of minimizing the distance, measured in bits, between them.

In a deterministic regression problem for forecasting \( h_0: X \to Y \), the cost functional of cross entropy is taken to be the loss function [21]

\[ l(h_0(x), y) := y \log(h_0(x)) + (1 - y) \log(1 - h_0(x)) \]  

where it is noted that \( h_0(x) \in \mathbb{R} \) and \( y \in \mathbb{R} \) are not probability distributions, because they are actual values and are not symbolized in a finite state space; hence, they are not probabilistic and takes the form as a deterministic regression. Similarly, the \( \ell_2(\mathbb{R}) \)-norm cost functional is also used for regression tasks, as opposed to determining probabilities of new classes and symbols. If used in a probabilistic sense the \( \ell_2(\mathbb{R}) \)-norm cost functional assumes restrictive second-order statistics.

Training the neural network to make inferences involves taking the expected loss over Eq. (9)

\[ L(\Theta; D) := E_{(x,y)}[l(h_0(x,y), q(x,y))] \]  

4 Results and Discussion

This section presents the results and comparisons of the three proposed neural network algorithms, namely, LSTM, FF, and LSTM-FF running in parallel, on the chaotic experimental data collected from a combustor apparatus that is described in the following.

4.1 Description of the Combustor Apparatus. The apparatus is built upon a swirl-stabilized, lean-premixed, laboratory-scale combustor that has been used to perform the experimental investigation. Figure 2 shows a schematic diagram of the variable-length combustor, consisting of an inlet section, an injector, a combustion chamber, and an exhaust section. High-pressure air is delivered to the apparatus from a compressor after passing through filters to remove any liquid or solid particles that might be present in the inlet air. The air supply pressure is set to approximately 1.34 MPa using a dome pressure regulator. The air is preheated to a maximum temperature of 250°C by an 88 kW electric heater. The fuel for this study is natural gas (approximately 95% methane) that is supplied to the combustor system at a pressure of approximately 1.48 MPa. The flow rates of the air and natural gas are measured by thermal mass flow meters. The desired
equivalence ratio and mean inlet velocity is set by adjusting these flow rates with needle valves.

4.2 Forecasting Physical Phenomena. Combustion and fluid processes, which are governed by physical process dynamics that can be represented by highly coupled, nonlinear partial differential equations, generate chaotic pressure signals; the underlying dynamics are very difficult to predict. The proposed algorithms, which serve the purpose of instability prediction, are dynamic data-driven [13,27], instead of solely relying on constitutive model equations and thermodynamic state relations. The goal here is to predict combustion instabilities for possible usage in a feedback control system to mitigate instabilities.

Tests were conducted at a nominal combustor pressure of 1 atm over a range of operating conditions. At each operating condition, time series of pressure data were collected for 8 s at sampling rate 8192 Hz, which is sufficiently long to capture the dynamic characteristics of the underlying process and which, in addition, can be considered to be approximately statistically stationary. Based on these test data, individual models have been compared by averaging over their element-wise relative error rates; reductions in the average error rate between algorithms have been examined.

Combustion instabilities are commonly defined by the RMS values of the pressure signals inside the combustion chamber of a trailing window, with the intuition that a larger RMS value implies existence of larger pressure fluctuations and thus larger instabilities. Figure 3 shows characteristic probabilistic forecasts, where black corresponds to low probability and white corresponds to high probability. It is seen that the LSTM makes more confident predictions than the FF architecture. As the forecast length is increased from 4 to 10 time steps, both architectures are less confident in their predictions and thus generate more diffuse prediction densities. This is expected because there are more uncertainties in the prediction with increased forecast length. The true trajectory in red in Fig. 3 is compared with the expectation over the forecasted probability distributions, where the centroid of the partitioned region is taken to be the expected value of the random variable.

Figure 4 shows how the different neural network architectures perform across all test data sets. These data sets are composed of all combinations of forecast lengths of 2, 4, 6,..., 20 and the symbol alphabet size (i.e., number of symbols) being 2, 3, 4,..., 11, making 100 unique combinations in total. Increasing the symbol alphabet size (i.e., number of symbols) enhances the fidelity of the model at the possible expense of model accuracy as seen in Fig. 4, because of either the inherent difficulty in forecasting at such a high resolution or the finite data length [10,12]. This general trend is seen across all three tested architectures. It is noted that the design parameters (i.e., fidelity, forecast length and accuracy) are application specific.

Referring to Fig. 4, Table 1 lists the average relative reduction in error over all pairwise combinations, where the relative error reduction is calculated as mean \(1 - A/B\), and \(A/B\) is element-wise division of the matrices \(A\) and \(B\) from Fig. 4. In conclusion the LSTM performs the best, followed closely by LSTM-FF and the last is FF.

Figure 5 shows the errors, weighted by the distance between the predicted symbol and the true symbol. This is perhaps a better performance measure of the architectures because the error rate should be penalized more heavily by the deviation of the predicted symbol from the true symbol. The weighted error can be interpreted as the average distance between the predicted and true...
symbol, scaled between 0 and 1. The general trend reaffirms that the error rates increase as the forecast length increases, as well as when the number of symbols increases. When averaged over all pairwise combinations, the relative reductions in weighted error can be seen in Table 2, where again the relative reduction in weighted error is calculated from Fig. 5 as mean \(1 - \frac{A}{B}\), and \(A/B\) is element-wise division of matrices \(A\) and \(B\).

It is seen that the LSTM algorithm outperforms the FF algorithm, because the recurrent feedback mechanism in LSTM is specifically designed to accommodate time series data. Similarly, the LSTM outperforms the LSTM-FF, likely because the feed forward aspect of LSTM-FF is corrupting the dynamic features learned by LSTM.

5 Summary, Conclusions, and Future Work

This brief paper has developed a neural probabilistic framework for forecasting time series from neural network architectures in terms of discrete symbols as opposed to real values without any major limiting assumptions such as second-order statistics. Although discrete symbolization is subjected to potential loss of resolution compared to continuous data, it enhances performance robustness especially if the signal-to-noise ratio is not high [28]. In particular, the proposed method yields performance robustness by taking the expectation over the forecasted probability distribution. This formulation of probabilistic forecasting is suitable for continuously varying uncertain dynamical systems, in which control actions operate at a slower time scale than that of system dynamics; such systems are prevalent in diverse mechanical engineering applications.

Three neural probabilistic architectures, namely, FF, LSTM, and combined LSTM and FF neural networks running in parallel, have been tested on experimental (time series) data of chaotic pressure oscillations collected from a laboratory-scale combustor apparatus. The performance of these three architectures has been comprehensively compared, where each architecture has been trained for 100 combinations of forecast length and number of symbols, so that application-specific requirements can be optimally chosen. It is found that, on the average, the LSTM performs the best, followed by LSTM-FF, and finally the FF.

The future research should focus on comparison of the aforementioned three neural architectures with AR(1)MA models on user-defined, controlled chaotic datasets. For example, such datasets can be generated from solutions of the Duffing equation [29] that can be studied as a two-dimensional time-series trajectory in the phase-space with multidimensional symbolization techniques (e.g., \(k\)-means), as described in Sec. 2.3. While there are many other issues that need to be resolved by further theoretical and experimental research, the authors suggest comparison of the present work with various other applications (e.g., Cheng et al. [30]) as a topic of future research.

Acknowledgment

The first author has been supported by PSU/ARL Walker Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies. The authors would like to thank Professor Domenic Santavice and Mr. Jihang Li for kindly providing the experimental data used in this work.

Funding Data

- U.S. Air Force Office of Scientific Research (AFOSR)
  (Grant No. FA9550-15-1-0400).

References


Table 1 Relative reduction in error

<table>
<thead>
<tr>
<th></th>
<th>LSTM/FF</th>
<th>LSTM/LSTM-FF</th>
<th>LSTM-FF/FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.57%</td>
<td>0.587%</td>
<td>1.99%</td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Relative reduction in weighted error

<table>
<thead>
<tr>
<th></th>
<th>LSTM/FF</th>
<th>LSTM/LSTM-FF</th>
<th>LSTM-FF/FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.94%</td>
<td>0.997%</td>
<td>2.98%</td>
<td></td>
</tr>
</tbody>
</table>