A Computationally Scalable Bayesian Sequential Learning Framework for Time-Series Forecasting

Viraj Parimi[1] , **Isaac Kumar**[1] , **Stephen F. Smith**[1]

[1]Robotics Institute, Carnegie Mellon University
Pittsburgh, PA

{vparimi, isaack, sfs}@cs.cmu.edu

Abstract

Time-series forecasting is a widely used data science technique for predicting the future state of stochastic mechanisms. Application domains that benefit from such practices include stock markets, inventory planning, supply chain management, healthcare, and resource allocation under uncertainty. In recent years, deep learning has increasingly become the method of choice in time-series forecasting applications where historical data is abundant. Alternatively, frequentist approaches are quite popular in applications where historical data is limited, but their underlying assumption of stationary data tends to restrict their performance. This paper considers an alternative approach to small data applications, extending a Bayesian sequential learning technique to overcome this shortcoming. Historically, Bayesian learning approaches have suffered from scalability problems when applied to time-series forecasting due to the Bayesian estimation step's complexity. Contemporary Bayesian approaches utilize Markov Chain Monte Carlo methods such as Metropolis-Hastings and Hamiltonian Monte Carlo for this purpose. While these methods have proved useful for offline time-series analysis, their computational requirements limit their utility in online, high temporal frequency forecasting. We propose substituting Nested Sampling, another method for estimating Bayesian evidence, as a means of achieving a scalable time-series forecasting framework. We demonstrate our approach's effectiveness via comparative experimental analysis on three prediction problems of practical interest.

1 Introduction

Time-series forecasting plays an instrumental role in predicting the evolution of complex systems. Application domains that benefit from such techniques include inventory planning [Kourentzes *et al.*, 2020], supply chain management [Abolghasemi *et al.*, 2020], financial stock markets [Sezer *et al.*, 2020], healthcare [Kaushik *et al.*, 2020], and engineering systems [Yang *et al.*, 2011].

Advances in machine learning have provided new techniques for predicting the behavior of complex systems over time. Forecasting techniques such as support vector machines [Balabin and Lomakina, 2011], hidden or state-observer Markov models [Kong *et al.*, 2011], Bayesian non-parametric models [Ko and Fox, 2009], and deep learning models [Sezer *et al.*, 2020] have been employed for this purpose. For time-series forecasting, deep learning has become the preferred model of choice in recent years [Sejnowski, 2020]. However, as pointed out in [Dargan *et al.*, 2020], deep learning's major challenge is the requirement of abundant data. As the data becomes scarce, simpler frequentist learning models such as Autoregressive Integrated Moving Average (ARIMA) [Box *et al.*, 2015] tend to perform better and are generally still preferred as long as the assumption of stationary data is viable.

Recent work has proposed an alternative, Bayesian sequential learning framework for time-series forecasting in small data applications and demonstrated its effectiveness in a particular setting: predicting bus dwell times at city bus stops [Isukapati *et al.*, 2020]. This approach is interesting because it overcomes the stationary data limitation of ARIMA models. However, like other Bayesian learning approaches, the proposed framework suffers from the high computational cost of performing the Bayesian estimation step on each cycle, which minimally restricts its use to offline analysis settings. It effectively precludes its application to most time series forecasting problems of practical interest.

In this paper, we present a computationally scalable variant of the Bayesian sequential learning framework proposed in [Isukapati *et al.*, 2020]. We focus specifically on improving the efficiency of the Bayesian estimation step of their procedure, which utilizes the Metropolis-Hastings algorithm [Metropolis *et al.*, 1953; Hastings, 1970]. Such Markov Chain Monte Carlo (MCMC) methods are commonly employed for this purpose in Bayesian learning settings, and Metropolis-Hastings is a popular choice. However, despite their power and practical utility, MCMC methods have a couple of limitations that affect their computational efficiency. First, the algorithms' samples are necessarily correlated, thereby resulting in reduced effective sample size and larger error in posterior estimates, which can necessitate a longer sampling process. Second, MCMC methods' efficiency depends on designing a useful likelihood function - a difficult task for high-dimensional spaces. While methodological advances help

mitigate these issues, their usefulness is limited in sequential Monte Carlo methods.

To achieve greater efficiency in the evidence evaluation step, we replace Metropolis-Hastings with another type of sampling procedure: Nested sampling [Skilling, 2006]. Nested Sampling gains efficiency because it evaluates the evidence directly from the prior by computing the cumulative prior function conditioned over the likelihood values. Based on the past samples, it directly discards one sample in each iteration, which corresponds to the least likelihood value. To our knowledge, this is the first time that the Nested Sampling methodology has been used in this context.

We evaluate the revised framework's performance on three distinct time-series forecasting problems from the literature, including the bus dwell time problem studied in [Isukapati *et al.*, 2020]. Metropolis-Hastings and Hamiltonian Monte Carlo (HMC), another well-known MCMC method, are used as benchmarks to evaluate computational efficiency. The Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] deep learning model and an ARIMA model are compared to assess prediction accuracy. Results show that Nested Sampling yields several orders of magnitude speedup over the MCMC methods with minimal accuracy loss. On all three problems, the approach achieves greater accuracy than LSTM and ARIMA while improving on the results reported in [Isukapati *et al.*, 2020] in the process.

The rest of the paper is organized as follows. First, we describe our revised Bayesian sequential learning framework for time series forecasting. Next, we present a comparative experimental analysis of the approach and analyze its scalability and prediction accuracy relative to relevant competing techniques. Finally, we summarize the main contributions of the paper and briefly indicate directions of future work.

2 Bayesian Sequential Learning Framework

We propose a novel extension to a Bayesian sequential learning framework for time-series forecasting that follows a *rolling Bayesian update scheme* inherently. The idea was first proposed in [Isukapati *et al.*, 2020] and involves selecting a likelihood function offline that best fits the available historical data and then adaptively refining the model's parameters in the light of new data. Often, small historical datasets are sufficient in the defensible construction of such likelihoods, thereby eliminating the need for extensive training datasets. This process results in a lightweight framework that naturally adapts to the underlying non-stationary stochastic process while quickly improving by updating its parameters as and when new data arrives.

The framework consists of two significant steps: (1) Select a likelihood function that best fits the empirical data distribution. To perform this, we first chronologically order the dataset and then compute the observed data distribution using the Kernel Density Estimation (KDE) technique. Furthermore, we fit a set of chosen analytic distributions generally taken based on their applicability in survival analysis and compute the Maximum Deviation Test (MDT) scores [Isukapati and List, 2016]. This score is defined as the number of percentile values in an analytic fit within a user-defined threshold of empirical data distribution; (2) Choose a prior distribution for each likelihood function parameter. This is a fairly straightforward process – one can either choose a predictive prior based on a historical dataset, which need not be very large, or an uninformed prior in the absence of such data. A unique feature about any Bayesian approach is that the impact of the prior on the posterior predictive distribution diminishes as more Bayesian updates are made in the light of new data.

Algorithm 1: Bayesian Sequential Learning Framework

Input: D_t, M
/* D_t - current covariate information, */
/* M - most recent parameter samples */
Output: P
/* P - Prediction of the next data point */
Initialize the samplers by defining the chosen likelihood and prior functions
while $t < \infty$ **do**
  if M *is not None* **then**
    $Z, R \leftarrow$ Samples of the parameters using D_t
    Initialize S to store the samples of the variable to predict
    foreach *parameter sample* R_i **do**
      $S_i \leftarrow$ Perform inverse transformation of the parametric distribution using R_i
      Update S to store this new point S_i
    $S_{median} \leftarrow$ median of each sublist in S
    $P \leftarrow$ mean of S_{median}
    Utilize evidence Z to form the posterior distribution
    Yield: P
  $K_M \leftarrow$ Compute set of Kernel Density Estimates for each parameter in M
  $M \leftarrow$ Generate samples of approximate posterior distribution using some sampling methodology using old M, the likelihood function, and p_t

Finally, to bootstrap the system, the framework sets prior distributions for all model parameters in the hierarchy. Once data is observed, a Bayesian update is performed to obtain the set of posterior distributions. These distributions are then used as priors for the next Bayesian update and are used to obtain the posterior predictive distribution of the target variable. Algorithm 1 summarizes specific details of this framework.

As demonstrated in [Isukapati *et al.*, 2020], this framework proved to be quite useful in accurately predicting bus dwell times in comparison to both deep learning and linear regression models. Simultaneously, the methodology relies on Metropolis-Hastings, a MCMC-based sampling technique, for estimating the Bayesian evidence at each step, which is quite expensive computationally and significantly limits the overall scalability of the approach. Many time series forecasting problems require online prediction, as data is acquired incrementally during execution.

To counter this disadvantage, we propose substituting a lesser-known sampling technique called Nested Sampling

[Skilling, 2006] to improve the efficiency of estimating Bayesian evidence at each step of the analysis. As noted earlier, Nested Sampling has the attractive property that it evaluates the evidence directly from the prior by computing the cumulative prior function conditioned over the likelihood values. The evidence then becomes a one-dimensional integral over a unit range in which the integrand is given by the inverse of the cumulative prior function. Historically, Nested Sampling has been used as an alternative evidence estimation method in snapshot characterization of time-series data sets. But this application to time-series forecasting is new. The technical details of this sampling method are summarized below.

Nested Sampling works on the principle that, for a positive-valued random variable, the area above the cumulative distribution function (CDF) is given by the expected value of the random variable.

$$\int_0^\infty (1 - F(x))dx = \int_0^\infty xf(x)dx \equiv E(X)$$

where, X is a random variable with probability density function (PDF) f and CDF F.

Let θ be the set of parameters in a multi-dimensional space, $\pi(\theta)$ denote prior likelihood of these parameters, and L represents the likelihood on data D given θ. Then, the evidence is provided by

$$F(\lambda) = \int_{L(D|\theta)\geq\lambda} \pi(\theta)d\theta$$

$$X(\lambda) \equiv 1 - F(\lambda) = \int_{L(D|\theta)\geq\lambda} \pi(\theta)d\theta$$

$$\int_\theta L(D \mid \theta)\pi(\theta)d\theta = \int_0^1 X^{-1}(p)dp$$

Please note that even if the set of parameters θ is multi-dimensional, the function X^{-1} is always one-dimensional and monotonic. The methodology employs the following steps to preclude the likelihood from diffusing, especially in high-dimensional spaces: 1) It samples the first N points from the previous π and determines their corresponding likelihoods; 2) It uses the minimum of these likelihood values L_i to estimate the $(N-1)^{\text{th}}$ quantile and removes this point from N; 3) It samples a new point from the prior satisfying the condition $L(D|\theta) > L_i$; 4) The new set of N points is treated as a set of N independent random draws from the restricted prior and the smallest likelihood among these points gives the estimate of $X^{-1}((\frac{N-1}{N})^i)$; 5) Multiple repetitions of steps 2 and 3 ensure the sampling of points from higher likelihood regions. The evidence is calculated using a weighted sum approach where each new point's weight is computed as the width between the latest and the preceding points. The process is terminated when the width falls below a user-defined threshold; 6) Numerical integration techniques such as trapezoidal rule are employed to approximate the evidence on sample points generated by step 5.

Finally, previously computed Bayesian evidence in conjunction with the inverse transformation of the parametric distribution representing the target variable is used in calculating the posterior likelihood. Algorithm 2 provides a high-level overview of the Nested Sampling algorithm.

Algorithm 2: Nested Sampling

Input: N, M, L, π
/* N - number of live points, M - number of iterations, L - likelihood function, π - prior function */
Output: Z, θ
/* Z - total evidence, θ - final set S of parameter proposals where $|S| = N$ */
Start with N points $\theta_1, \cdots, \theta_N$ from the prior π
Initialize $Z = 0$, $X_0 = 1$
for $j \leftarrow 0$ **to** M **do**
  Record the lowest of current likelihood values as L_j
  θ_j is the point corresponding to L_j
  $X_j \leftarrow \exp(-\frac{j}{N})$
  /* X_j is an estimate of the prior mass covered by the hyper-volume in parameter space of all points with likelihood greater than θ_j */
  $w_j \leftarrow X_{j-1} - X_j$
  /* w_j is the estimate of the amount of the prior mass between two nested hyper-surfaces */
  $Z \leftarrow Z + L_j w_j$
  Replace θ_j by new sample drawn from prior $\pi(\theta)$ constrained by $L(\theta) > L_j$.
$Z \leftarrow Z + \frac{(L(\theta_1)+L(\theta_2)+\cdots+L(\theta_N))X_j}{N}$
$\theta \leftarrow (\theta_1, \theta_2, \cdots, \theta_N)$
return Z, θ

3 Computational Experiments

The experiments presented below were designed with the following objective: (1) to quantify the computational benefit of incorporating nested sampling, (2) to demonstrate the ability of the framework to learn underlying system characteristics and accurately predict when historical data is sparse, and (3) to demonstrate the ability of the framework to handle both stationary and non-stationary stochastic mechanisms.

Three use-cases from different disciplines were considered: one canonical example, the transportation use-case considered in [Isukapati *et al.*, 2020], and another use-case from astrophysics. Each draws on a publicly available data set, and there is variation in parameter dimensionality across all three use-cases.

The PyMC3 [Salvatier *et al.*, 2016] library was used to implement Metropolis-Hastings and HMC, and the Dynesty [Speagle, 2019] library was used to implement Nested Sampling. For the LSTM model, we utilized the Keras library [Chollet and others, 2015], and for the ARIMA model, we used the Statsmodels library [Seabold and Perktold, 2010]. All experiments were run on a 4th generation Intel Haswell processor running at 3.60GHz clock speed with four cores and two hardware-level threads. The framework itself was developed in Python3.5, and the source code will be made available.

3.1 Mackey-Glass Sequence

The Mackey-Glass time-series [Mackey and Glass, 1977] sequence is a stationary stochastic process that generates time-series samples from a non-linear differential delay equation. The discrete realization [Matous, 2019] of this equation is given by,

$$x_{t+1} = a\frac{x_d}{b + x_d^e} + cx_t$$

where a, b, c, e are treated as the unknown parameters and d represents the $(t-d)^{\text{th}}$ data point. For the purpose of experiments we set $a = 0.2$, $b = 0.8$, $c = 0.9$, $d = 17$, $e = 10$ and the initial value of the sequence $x_0 = 0.1$ and generated a 1000 length sequence. In the context of Bayesian sequential learning framework for time-series forecasting, the Mackey-Glass sequence x is modeled using a Gaussian function $\mathcal{N}(.|x, s^2)$ with parameter space given by $\theta = (a, b, c, e, s)$. On the other hand, this sequence's LSTM model involves constructing LSTM layers with hidden unit sizes of 5 for each cell. A split ratio of 0.2 is used for dividing the sequence data into train, validation, and test datasets. Mean-squared error (MSE) and the Adam optimizer [Kingma and Ba, 2015] are used to train the model. The validation set is used to compute the optimal hyperparameters of the LSTM model. The training process is terminated if the validation accuracy does not improve significantly in the subsequent epochs. For ARIMA, we tune the model to fit the training data to identify the p, d, and q parameters depending on goodness of fit characterized by the model's AIC score.

Figure 1 presents plots for predicted vs actual sequence values. It contains five subplots - one for each method. As can be seen, all three sampling techniques significantly outperformed LSTM and ARIMA in terms of accuracy given by MSE. Table 1 summarizes the accuracy and average computational time statistics for each method. We varied the number of samples between [100, 750] for sampling methods and the look-behind window between [1, 20] for LSTM. The summary tables suggest that the average maximum computational times of Metropolis-Hastings, HMC, and Nested Sampling are around 431, 237, and 0.08 seconds respectively, implying that Nested Sampling performs up to 5 orders of magnitude better than the other benchmarks.

Additionally, it can be observed that the computation times of Nested Sampling scaled linearly as opposed to Metropolis-Hastings and HMC. The LSTM model results are in line with the findings in [Gers *et al.*, 2001] as the model could not capture the sequence's underlying patterns. Furthermore, the ARIMA model also had a hard time capturing the model's trends as it tends to converge to a mean value after a few predictions.

3.2 Bus Dwell Time

Accurate prediction of bus dwell times at bus stops can improve real-time traffic signal control performance by improving the accuracy of the vehicle intersection arrival models used to drive traffic signal decisions. To enable this possibility, [Isukapati *et al.*, 2020] considered this problem using the historical Advanced Vehicle Location (AVL) dataset provided by the Port Authority of Allegheny County for two major bus routes. Following the procedure described in Section 2, [Isukapati *et al.*, 2020] found that the Fisk distribution offered the best fit to this data, and we similarly have adopted this conclusion. The model developed can be described as follows,

$$\begin{aligned}
X &= \exp(Y) \qquad (\text{where } Y \sim \text{Logistic}(\mu, s)) \\
\mu &= \ln(\alpha) = \ln(\boldsymbol{\beta}_{\boldsymbol{\alpha}}^T \boldsymbol{x} + \beta_0) \\
s &= 1/\tau = 1/(\boldsymbol{\beta}_{\boldsymbol{\tau}}^T \boldsymbol{x}) \\
\boldsymbol{\beta}_{\boldsymbol{\alpha}} &= \begin{bmatrix} \beta_\alpha^{\text{on}} & \beta_\alpha^{\text{off}} \end{bmatrix}^T \\
\boldsymbol{\beta}_{\boldsymbol{\tau}} &= \begin{bmatrix} \beta_\tau^{\text{on}} & \beta_\tau^{\text{off}} \end{bmatrix}^T \\
\boldsymbol{x} &= \begin{bmatrix} x_{\text{on}} & x_{\text{off}} \end{bmatrix}^T
\end{aligned}$$

Here β_α, β_τ and β_0 represent the parameters with x representing the covariate information. At any given time, the belief of the two parameters μ and s describe the current belief of bus dwell time distribution. Note that this model also has a 5-dimensional parameter space; hence we make use of the same LSTM model described in 3.1.

Table 2 summarizes the results obtained for the bus dwell time prediction use-case. The Nested Sampling method is seen to outperform both MCMC-based sampling-based methods and improve the results reported in [Isukapati *et al.*, 2020], both in accuracy and computational efficiency, and irrespective of the variance in the underlying dataset associated with different bus stops.

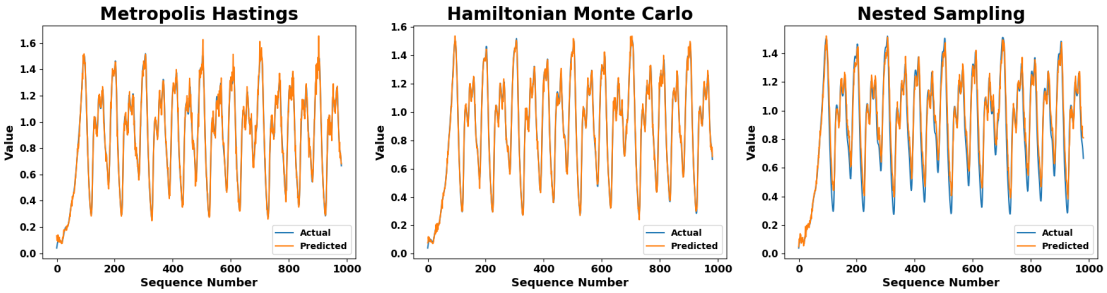
It is interesting to note that although the LSTM model generally under-performed, it did produce better results than the revised framework at Negley Ave at #370. This superior performance can be attributed to the low underlying variance within the data stream at that stop. The ARIMA model ultimately failed to estimate the data trends mostly because it is highly stochastic and non-stationary.

3.3 Radial Velocity

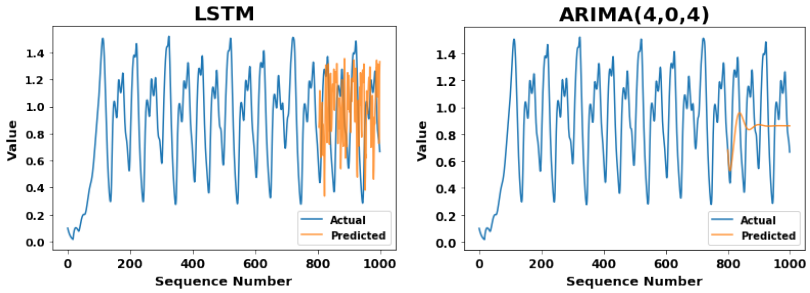
Radial velocity [Sharma, 2017] of an object with respect to a reference point is defined as the rate of change in distance between that point and the object itself. The estimation of radial velocity is a stationary stochastic process, and it is instrumental in discovering new planets. An exoplanet around a companion star causes temporal variations in the radial velocity measurement of that star. One can analyze and deduce the ratio of masses between the planet and the companion star and additional orbital parameters like eccentricity. Mathematically, the radial velocity of a star of mass M in a binary system with an exoplanet of mass m in an orbit of time period T, inclination I, and eccentricity e can be defined as follows,

$$v(t) = \kappa[\cos(f + \omega) + e\cos\omega] + v_0$$

where,



(a) Metropolis-Hastings (b) Hamiltonian Monte Carlo (c) Nested Sampling



(d) LSTM (e) ARIMA

Figure 1: Performance of the methodologies applied on the Mackey-Glass sequence.

Bayesian Sequential Learning Framework							Deep Learning Based			Frequentist Based		
Metropolis Hastings			Hamiltonian Monte Carlo		Nested Sampling		LSTM			ARIMA		
#	MSE	Time(s)	MSE	Time(s)	MSE	Time(s)	ϕ	MSE	Time(s)	ρ	MSE	Time(s)
100	0.0013	67.6050	0.0006	115.1488	0.0906	0.0107	1	0.1264	8.2421	0.5	0.1134	0.5891
250	0.0005	318.0218	0.0004	407.5559	0.0262	0.0258	5	0.0721	18.6681	0.6	0.1130	0.5341
500	0.0010	427.3962	0.0001	88.6842	0.0128	0.0538	10	0.0857	47.9600	0.7	0.1052	0.6728
750	0.0006	431.3002	0.0006	237.6586	0.0104	0.0823	20	0.1245	44.1125	0.8	0.1014	0.8274

Table 1: Summary of model performances under Mackey-Glass Sequence prediction. # represents the number of samples used by the sampling-based algorithms, ϕ represents the look-behind window length for the LSTM model, and ρ represents the split ratio of the dataset which decides on how much training data should be used to fit the frequentist model

$$\kappa = \frac{(2\pi G)^{1/3} m \sin I}{T^{1/3}(M+m)^{2/3}\sqrt{1-e^2}}$$

$$\tan(f/2) = \sqrt{\frac{1+e}{1-e}}\tan(u/2)$$

$$u - e\sin u = \frac{2\pi}{T}(t-\tau)$$

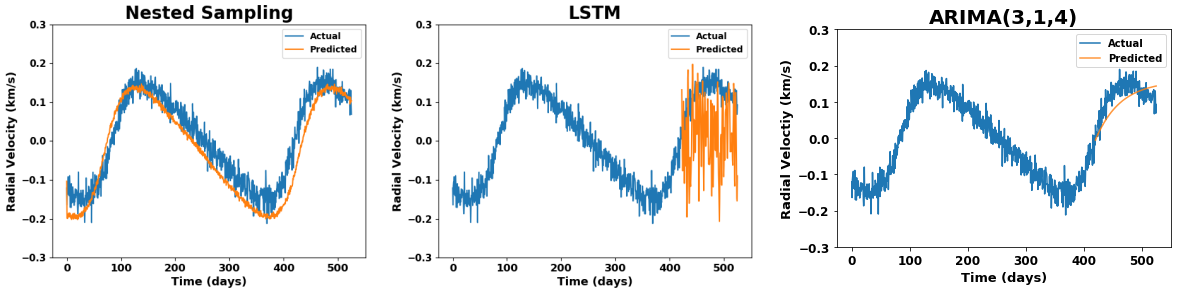
Here, I is the inclination of the orbital plane with respect to the sky, ω is the angle of the pericenter measure from the point where the orbit intersects the plane of sky in radians, τ is the time of passage through the pericenter, v_0 is the mean velocity of the center of mass of the binary system and G is the universal gravitational constant. Since T, I, and e are known, this makes κ a constant as well, which we can fix independently. Additionally, f is defined as the true anomaly function which depends on e, T and τ. This means that by just varying e, κ, T, τ, v_0 and ω one can generate a sequence of radial velocity ($v(t)$) data. Given the above formulation, one can model it via a Gaussian function $\mathcal{N}(.|v, s^2)$. With this we can define the parameter space of this model as $\theta = (v_0, \kappa, \omega, \tau, T, e, s)$

For the purpose of experiments we generated a 1000 length radial velocity data sequence with $v_0 = 0$, $\kappa = 0.15$, $T = 350$, $e = 0.3$, $\tau = 87.5$ and $\omega = -90$. We followed a similar experimentation procedure as in Section 3.1 with only one major change with regards to the LSTM network. The number of hidden units in an LSTM layer for each cell were increased from 5 to 7. We also dropped consideration of MCMC-based sampling variants in the comparative analysis, as their lack of scalability is clear from the first two use-cases.

As indicated in Figure 2, the Nested Sampling method outperforms the LSTM and ARIMA models in terms of accuracy and computational expense. The most important take-

Variance	Bus Stop	Bayesian Sequential Learning Framework									Deep Learning Based			Frequentist Based		
		Metropolis Hastings			Hamiltonian Monte Carlo			Nested Sampling			LSTM			ARIMA		
		#	Acc	Time(s)	#	Acc	Time(s)	#	Acc	Time(s)	#	Acc	Time(s)	#	Acc	Time(s)
Low	Centre Ave Opp Shadyside Hos	600	0.77	9.26	10000	0.74	128.10	900	**0.79**	**0.29**	NA	0.49	22.67	NA	0.00	5.76
	Negley Ave at #370	900	0.89	9.20	10000	**0.91**	139.12	1000	**0.88**	**0.32**	NA	0.88	6.07	NA	0.00	0.33
Medium	Centre Ave at Morewood Ave	1000	0.71	11.39	10000	0.67	73.02	1000	**0.74**	**0.26**	NA	0.38	32.45	NA	0.02	4.67
	Centre Ave at Cypress St	200	0.71	7.50	15000	0.63	244.81	1000	**0.73**	**0.26**	NA	0.44	10.09	NA	0.04	3.14
High	Negley Ave at Centre Ave	700	0.47	8.58	600	0.46	49.44	700	**0.5**	**0.19**	NA	0.17	46.63	NA	0.00	14.92
	Centre Ave at Aiken Ave	1000	**0.59**	10.02	1000	**0.53**	76.46	200	**0.66**	**0.05**	NA	0.22	18.04	NA	0.03	4.48

Table 2: Summary of model performances under Bus Dwell Time prediction use-case. # represents the number of samples used by the sampling-based algorithms.



(a) Nested Sampling (b) LSTM (c) ARIMA

Figure 2: Performance of the methodologies applied on the radial velocity prediction use-case.

Bayesian Sequential Learning Framework			Deep Learning Based			Frequentist Based		
Nested Sampling			LSTM			ARIMA		
#	MSE	Time(s)	ϕ	MSE	Time(s)	ρ	MSE	Time(s)
100	0.0015	0.0899	1	0.0124	7.7335	0.5	0.0130	0.8028
250	0.0021	0.2371	5	0.0054	16.9276	0.6	0.0390	1.0157
500	0.0015	0.4913	10	0.0092	29.6531	0.7	0.0420	0.9140
750	0.0016	0.6958	20	0.0113	46.8196	0.8	0.0016	0.9140

Table 3: Summary of model performances under Radial Velocity prediction use-case. # represents the number of samples used by the sampling-based algorithms, ϕ represents the look-behind window length for the LSTM model, and ρ represents the split ratio of the dataset which decides on how much training data should be used to fit the frequentist model

away from this experiment is that the proposed extension to the Bayesian sequential learning framework for time-series forecasting is quite broadly applicable, and that nested sampling is an efficient alternative to the MCMC-based sampling methodologies in any Bayesian learning context.

4 Conclusions and Future Work

In this paper, we have presented a Bayesian sequential learning framework for time-series forecasting that combines Bayesian inference techniques with Nested Sampling to generate accurate predictions in a computationally efficient manner, especially when the availability of training data is scarce. Computational experiments conducted in 3 distinct time-series forecasting problem domains showed an efficiency gain of several orders of magnitude over an analogous Bayesian learning framework that alternatively utilizes a contemporary MCMC sampling method while retaining comparable prediction accuracy. Results also showed an improvement in accuracy in all problems over a deep learning based LSTM model and a frequentist based ARIMA model, indicating the potential of the approach in small data applications.

Looking forward, we believe that Nested Sampling can be more broadly leveraged in other learning frameworks that currently utilize MCMC-based sampling methods. One focus of future research will be an investigation of its utility in hierarchical Bayesian modeling paradigms.

References

[Abolghasemi *et al.*, 2020] Mahdi Abolghasemi, Eric Beh, Garth Tarr, and Richard Gerlach. Demand forecasting in supply chain: The impact of demand volatility in the presence of promotion. *Computers & Industrial Engineering*, page 106380, 2020.

[Balabin and Lomakina, 2011] Roman M Balabin and Ekaterina I Lomakina. Support vector machine regression (svr/ls-svm)—an alternative to neural networks (ann) for analytical chemistry? comparison of nonlinear methods on near infrared (nir) spectroscopy data. *Analyst*, 136(8):1703–1712, 2011.

[Box *et al.*, 2015] G.E.P. Box, G.M. Jenkins, G.C. Reinsel, and G.M. Ljung. *Time series analysis: Forecasting and control (5th ed)*. John Wiley & Sons, Hoboken, New Jersey, 2015.

[Chollet and others, 2015] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[Dargan *et al.*, 2020] Shaveta Dargan, Munish Kumar, Maruthi Rohit Ayyagari, and Gulshan Kumar. A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*, 27(4):1071–1092, Sep 2020.

[Gers *et al.*, 2001] Felix Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. pages 669–676, 08 2001.

[Hastings, 1970] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[Isukapati and List, 2016] Isaac K Isukapati and George F List. Synthesizing route travel time distributions considering spatial dependencies. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2143–2149. IEEE, 2016.

[Isukapati *et al.*, 2020] I. K. Isukapati, C. Igoe, E. Bronstein, V. Parimi, and S. F. Smith. Hierarchical bayesian framework for bus dwell time prediction. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–10, 2020.

[Kaushik *et al.*, 2020] Shruti Kaushik, Abhinav Choudhury, Pankaj Kumar Sheron, Nataraj Dasgupta, Sayee Natarajan, Larry A Pickett, and Varun Dutt. Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in Big Data*, 3:4, 2020.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[Ko and Fox, 2009] Jonathan Ko and Dieter Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

[Kong *et al.*, 2011] Zhenyu Kong, Omer Beyca, Satish T Bukkapatnam, and Ranga Komanduri. Nonlinear sequential bayesian analysis-based decision making for end-point detection of chemical mechanical planarization (cmp) processes. *IEEE transactions on semiconductor manufacturing*, 24(4):523–532, 2011.

[Kourentzes *et al.*, 2020] Nikolaos Kourentzes, Juan R Trapero, and Devon K Barrow. Optimising forecasting models for inventory planning. *International Journal of Production Economics*, 225:107597, 2020.

[Mackey and Glass, 1977] MC Mackey and L Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[Matous, 2019] C. Matous. Signalz library, 2019.

[Metropolis *et al.*, 1953] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[Salvatier *et al.*, 2016] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using PyMC3. *PeerJ Computer Science*, 2:e55, apr 2016.

[Seabold and Perktold, 2010] Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

[Sejnowski, 2020] Terrence J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *Proceedings of the National Academy of Sciences*, 2020.

[Sezer *et al.*, 2020] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020.

[Sharma, 2017] Sanjib Sharma. Markov chain monte carlo methods for bayesian data analysis in astronomy. *Annual Review of Astronomy and Astrophysics*, 55(1):213–259, Aug 2017.

[Skilling, 2006] John Skilling. Nested sampling for general bayesian computation. *Bayesian Anal.*, 1(4):833–859, 12 2006.

[Speagle, 2019] Joshua S Speagle. dynesty: A Dynamic Nested Sampling Package for Estimating Bayesian Posteriors and Evidences. *arXiv e-prints*, page arXiv:1904.02180, Apr 2019.

[Yang *et al.*, 2011] Hui Yang, Satish TS Bukkapatnam, and Leandro G Barajas. Local recurrence based performance prediction and prognostics in the nonlinear and nonstationary systems. *Pattern Recognition*, 44(8):1834–1840, 2011.