Shallow and Deep Methods for Time-Series Modeling

Kyu-Young Kim [1]

Abstract

Over the past decade or so, deep learning models have been replacing traditional statistical models in machine learning systems in an increasing number of domains including computer vision and natural language processing. One use case in which deep learning models arguably have not been adopted to a similar extent is modeling time-series data, where the goal is to construct a model of a series of temporal data points in order to make accurate future predictions. In this work, we conduct a range of experiments to compare and contrast traditional statistical methods and deep learning methods for time-series modeling. We present experimental results that illustrate strengths and weaknesses of both types of approaches in terms of ease of training, data efficiency, generalization performance, and propose a more principled approach to time-series modeling.

1. Introduction

Deep learning's presence in machine learning applications has rapidly increased over the last decade, replacing many traditional statistical models. This rapid adoption is in part due to the following two notable characteristics: (a) deep learning models use multi-layer architectures that allow automatic learning of useful input representations for the given task (LeCun et al., 2015), reducing the need for hand-engineering of input features, and (b) deep learning models in general improve in performance with more data.

Time-series modeling deals with the statistical learning problem of constructing models of a series of data points with temporal dimension. Arguably, this is one domain in which deep learning has seen less adoption. This is conceivably in part due to that the aforementioned advantages of deep learning are not nearly as applicable in modeling of time-series data. For instance, in case of univariate datasets, the automatic learning of input representations is essentially irrelevant. Moreover, it is not uncommon to find time-series datasets containing only tens of data points, which is far less than what deep learning models are typically trained on.

In this work, we conduct an experimental study in which we investigate deep learning and classical time-series methods to gain insight into the characteristics of both methods in the context of modeling data with temporal dimension. We expect this and a similar line of future work to be important for understanding such aspects of deep learning as data efficiency and generalization performance.

2. Related Work

In (Barker et al., 2018) the authors at Microsoft propose a machine learning framework that extends classical time-series models such as Autoregressive Integrated Moving Average (ARIMA) and Exponential Time Smoothing (ETS) models to forecast 100% of Microsoft's revenue in 2016, which was roughly $85B. This speaks to both the level of adoption of classical models and the scale at which the models are used in time-series modeling.

In (Koenecke), the author explores both classical models and deep learning models to conduct a more direct comparison of the approaches on real revenue data from Microsoft. In the experiments, the author found deep learning models, if properly trained, for instance, using an appropriate curriculum learning strategy, to significantly outperform the production system that uses classical models. This is somewhat surprising in that rather simple variants of common deep learning models exceeded the models currently in production at companies like Microsoft. It is conceivable though that the sheer amount of data played to deep learning models' advantage.

3. Methods

3.1. Exponential Smoothing

Exponential smoothing is a statistical approach to forecasting that uses weighted averages of past observations with exponentially decaying weights. Conceptually, this means that the method applies higher weights to recent observations when computing the averages to make predictions.

Simple Exponential Smoothing. Simple exponential smoothing (SES) is the simplest form of exponential smoothing that is suitable for data the shows no apparent trend or seasonal patterns. Though not the most sophisticated, the model can often serve as a reasonable baseline. In mathe-

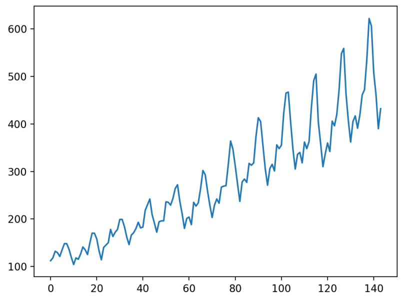


Figure 1. A time-series with a near linear trend and a seasonality.

matical terms, the model can be described as

$$\hat{y}_{T+1|T} = \sum_{t=1}^{T} \alpha(1-\alpha)^{T-t} y_t$$

where $\alpha \in [0, 1]$ is the smoothing parameter that can be estimated based on the observed training data. For prediction, SES employs a constant forecast function $\hat{y}_{T+h|T} = \hat{y}_{T+1|T}$. Hence, while the one-step forecast can be reasonably correct, accuracy is likely to decrease as we try to predict further out into the future.

Holt's Method. Holt's exponential smoothing method extends SES to handle data with a trend, which is a long term increase or decrease in the dependent variable. This addresses the shortcoming of SES that makes flat forecasts. The method can be summarized with the following set of equations (Hyndman & Athanasopoulos, 2021):

$$\begin{aligned}\hat{y}_{t+h|t} &= \ell_t + hb_t \\ \ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1}\end{aligned}$$

where $\alpha, \beta \in [0, 1]$ are the smoothing paramters for ℓ and b, respectively. Conceptually, ℓ estimates the level and β the slope of a given series, and the (linear) forecasting described above is then a sum of the current level at t and the product of the slope at t and h, the forecasting horizon.

Holt-Winters Method. Holt-Winters method extends the Holt's method to allow modeling data with seasonality, which is a pattern that repeats over a fixed interval. Figure 1 shows a sample time-series dataset that has an approximately linear trend with seasonality of length about 10 steps. The following formulation is a natural extension to the Holt's method with the seasonality equation (Hyndman & Athanasopoulos, 2021):

$$\begin{aligned}\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\ \ell_t &= \alpha(y_t - s_{t-m}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ b_t &= \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \\ s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}\end{aligned}$$

where k is the quotient of $\frac{h-1}{m}$, m the length of the seasonal pattern, and $\gamma \in [0, 1]$ the smoothing parameter for seasonality. Conceptually, the model attempts to decompose a given time-series into three components: level, trend, and seasonality.

3.2. Autoregressive Integrated Moving Average

ARIMA models are another family of statistical models that try to learn the autocorrelations, which are measures of how an observation in a time-series is correlated with the past observations, in the data.

Autoregressive Models. In an autoregressive model, we attempt to describe the dependent variable using a linear combination of a set of past observations of that variable. In other words,

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t$$

where c is the intercept term and ϵ_t the noise at time t. The model above is an autoregressive model of order p, often referred to as $AR(p)$ model.

Moving Average Models. In contrast to an autoregressive model, a moving average model describes the dependent variable with a linear combination of residuals, or forecast errors, as in

$$y_t = c + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

Conceptually, the model in making future forecasts attempts to correct the errors made on past steps (Brownlee). The model above is a moving average model of order q, referred to as $MA(q)$.

ARIMA Models. We obtain an ARIMA model when we combine autoregressive and moving average models and apply it to an optionally differenced time-series data. Differencing is an operation done to turn a non-stationary time-series into a stationay one by computing the differences between often consecutive observations (e.g., $y'_t = y_t - y_{t-1}$). Conceptually, if we fit a model to a once differenced time-series, we would be modeling the changes in the original time-series. An ARIMA model can be described as

$$y'_t = c + \sum_{i=1}^{p} \phi_i y'_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t$$

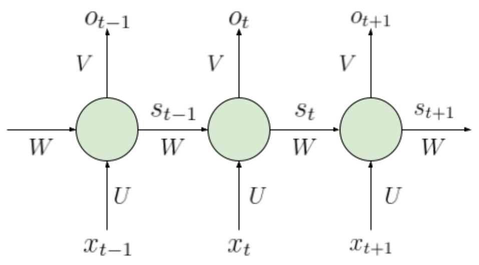


Figure 2. Unrolled recurrent neural network.

where y' indicates that the time-series has been differenced $d \geq 0$ times and is referred to as $ARIMA(p, d, q)$. This basic form of ARIMA model can be extended to model seasonality in time-series as well as to incorporate covariates. Finally, it is common to examine autocorrelation function (ACF) plots and to run statistical hypothesis tests of stationarity in order to choose appropriate values for the parameters p, d, and q (Hyndman & Athanasopoulos, 2021).

3.3. Recurrent Neural Network

RNN models are a special type of deep learning models that are designed to model such data as speech and text that are sequential in nature. RNNs process the input data one element at a time, maintaining a state representation that describes the history of the past elements encountered. The model uses this latent representation to predict the output (LeCun et al., 2015). RNN models are typically trained with the sum of cross-entropies across tokens as the loss, that is,

$$loss = -\sum_{t=1}^{T}\sum_{j=1}^{v} y_j \log \hat{y}_j$$

where T is the length of the sequence and v the size of vocabulary.

Figure 2 is a simple illustration of how an RNN model processes inputs, passes state information across time steps, and outputs predictions. There are many variants including GRU, LSTM, and layer-normalized LSTM that try to address different shortcomings of the vanilla RNN model such as the ability to model long-range dependencies.

4. Experiments

4.1. Dataset

The dataset used in the experiments is the weather data created by the Max Planck Institute for Biogeochemistry that contains 14 weather related features such as the temperature in degree Celsius collected every 10 minutes over multiple years since 2003: `https://www.bgc-jena.mpg.de/wetter`. That the set contains multiple years of data

Column	Variable
H2OC (mmol/mol)	Water vapor concentration
T (degC)	Air tempereature in celsius
Tdew (degC)	Dew point temperature in celsius
Tpot (K)	Potential temperature in kelvin
VPact (mbar)	Actual water vapor pressure
VPdef (mbar)	Water vapor deficit
VPmax (mbar)	Saturation water vapor pressure
wv (m/s)	Wind velocity in m/s
max. wv (m/s)	Max wind velocity in m/s
wd (deg)	Wind direction in degrees
rh (%)	Relative humidity
p (mbar)	Air pressure
sh (g/kg)	Specific humidity
rho (g/m**3)	Air density
Date Time	Date and time of the record

Table 1. Columns in the weather dataset (Kolle, 2008).

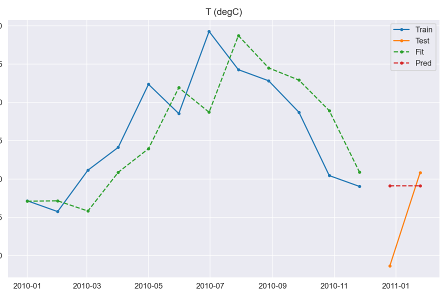
points allows to us to explore data efficiency of our chosen methods. Moreover, the availability of multiple covariates allows us to explore their effects on model fitting and forecasting. We pre-processed data to, for instance, remove invalid data points and to decompose velocity columns into x and y directions.

4.2. Results

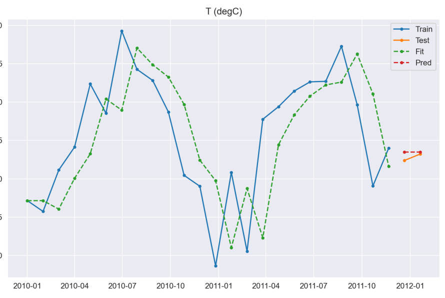
Below we show the results from a series of experiments in which we analyze the methods with regards to dataset size, forecasting accuracy, and affect of incorporating covariates. We mean absolute error (MAE), $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, as the main metric to estimate forecasting accuracy. For experiments, we used the statsmodels (Seabold & Perktold, 2010) and TensorFlow (Abadi et al., 2015) Python packages.

Dataset Size. Figure 3 shows SES models trained on monthly data over one year and two years, forecasting two steps into the future. Note that even in the case of only 12 data points to fit, the model predctions were quite reasonably, though flat and did not accurately capture some of the sudden changes. In case an additional year of data points were provided, the model predictions became noticeably better as seen in 2(b). However, when yet an additional year of data were used, the predictions became worse. This is in part due to that SES model places higher weights on recent data points is not well justified for seaonsal data like weather.

We attempted to train LSTM models on the same monthly data, but it was practically impossible to train using only a dozen or two data points. Instead, we took all data points prior to the start of when we want to make predctions and constructed training examples by taking samples of consec-

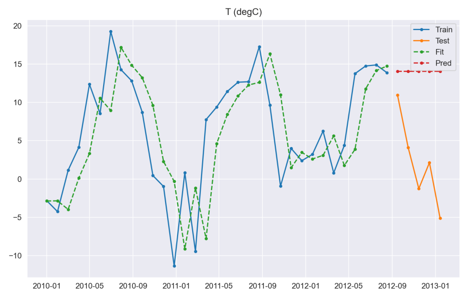


(a) 1 year of monthly data.

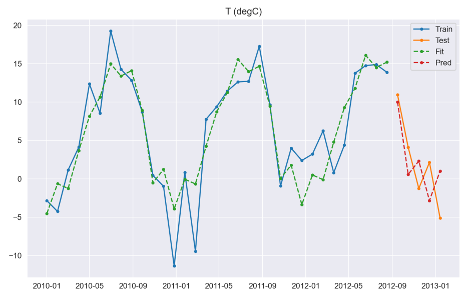


(b) 2 years of monthly data.

Figure 3. Simple exponential model on monthly data.



(a) Simple exponential smoothing.



(b) Holt-Winters.

Figure 4. Holt-Winters and SES on 3 years of monthly data.

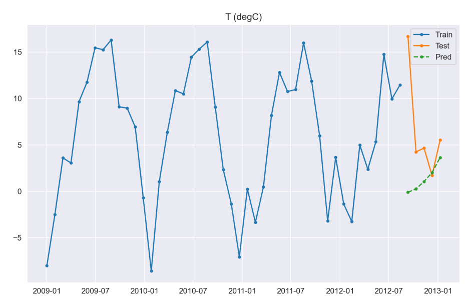
utive data points of an appropriate length. This means given a set of n time-series data points, we can construct a set of $n - (s + t) + 1$ unique training examples, where s is the length of the input sequence and t the length of the output. We can then construct batches examples for training by sampling from this set. The results from training LSTM models in this manner are discussed in the following sections.

Forecasting Accuracy. Forecasting accuracy depends not only on that the training dataset has sufficient data points to capture some statistical patterns to learn, but also on how accurate the model assumptions are for the given dataset. For instance, for SES model, which simply puts higher weights on more recent observations, we saw lower accuracy when trained on a larger set of data points. In contrast, Holt-Winters model specifically tries to model seasonal patterns. Figure 4 shows a comparison between SES and Holt-Winters models trained on 3 years of monthly data and forecasting 5 steps into the future. With a more appropriate set of assumptions for the dataset, the Holt-Winters model was able to capture more smoothly the yearly seasonal pattern in the dataset and made much more reasonable forecasts than the SES model. The MAEs achieved by the SES and Holt-Winters models were 11.89 and 3.83, respectively.

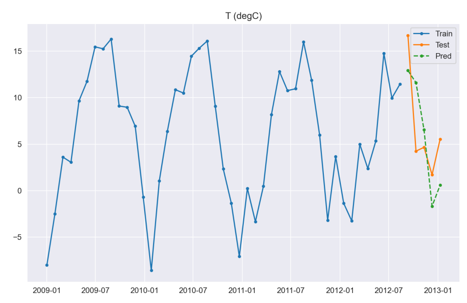
We also used LSTM models to forecast 5 steps into the future trained as previously described. Although the results are not strictly comparable due the difference in the training schemes, LSTM models trained with appropriate settings of the learning rate, number of epochs, batch size, etc. attained reasonable level of accuracy. Figure 5 shows LSTM models trained for 1k and 2k epochs over the training set, and they reached MAEs of 5.32 and 4.27, respectively. The two graphs suggest that hyperparamter settings can have significant impact on model accuracy. When appropriate hyperparamters are chosen, LSTM models trained in the manner described above demonstrated reasonable performance.

Covariates. To experiment with additional independent variables, we chose two covariates – rh (%) and rho (g/m**3) – based on the Spearman's correlation coefficient as shown in Figure 6. The covariates were chosen based on how correlated they are with the dependent variable, but we excluded the ones with too high correlations as adding them might lead the model to learn trivial relationships.

Figure 7 shows the performance of seasonal ARIMA models trained with no covariates and with rh (%) and rho (g/m**3) as additional exogenous variables. It is evident that adding relevant covariates noticeably improves the model's capability to fit the data and to more accurately forecast. While the model with no covariates learned the overall pattern of the dependent variable as demonstrated by the smooth curve, the model with the two covariates was able to capture more variability such as the sharp peaks around 2014 and to also

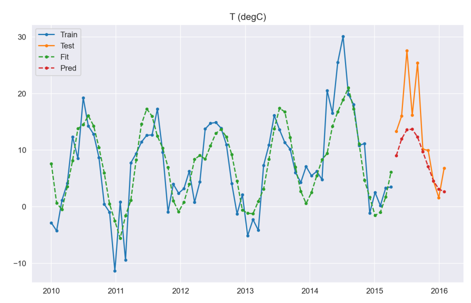


(a) Trained for 1k epochs.

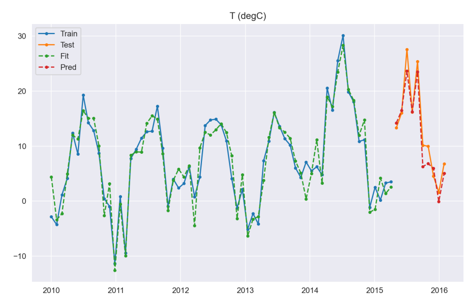


(b) Trained for 2k epochs.

Figure 5. LSTM models on monthly data.



(a) No covariates.



(b) With rh (%) and rho (g/m**3) covariates.

Figure 7. Seasonal ARIMA models.

forecast with a higher accuracy. The two models achieved MAEs of 4.68 and 1.90, respectively.

It is worth noting that one potential limitation of the ARIMA model with covariates above is that it requires the values of the variables to be available for forecasting. In some pratical settings, the values of the covariates at the time when we need to predict the dependent variable may not be

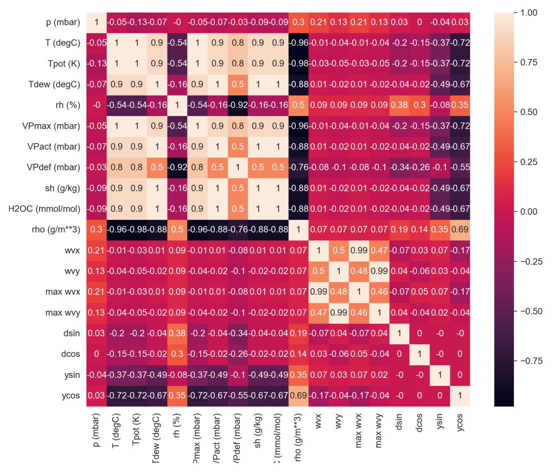


Figure 6. Spearman's correlation coefficient.

available. In such a situation, LSTM models can be naturally employed to learn to additionally predict the covariates.

5. Conclusion

In this paper, we applied both classical and deep learning models to the problem of modeling time-series data and analyzed such aspects as data efficiency, forecasting accuracy, and effects of incorporating covariates to the models. We observed that in case the size of the dataset to train on is on the order of tens of data points, a statistical model that makes assumptions appropriate for the data is more desirable than, for instance, an RNN type model. On the other hand, if at least hundreds of data points are available and the dependent variable has intricate relationships with multiple covariates, an RNN type model of an appropriate size can be a natural choice in more practical settings. As a more principled approach to time-series modeling, some of the statistical analyses commonly done for classical models such as analyzing the ACF and PACF graphs can also be employed to decide, for instance, the length of the input sequence even for RNN type models.

6. Contributions

I am the sole member of this group and contributed to all parts of the project.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Barker, J., Gajewar, A., Golyaev, K., Bansal, G., and Conners, M. Secure and automated enterprise revenue forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11385.

Brownlee, J. A gentle introduction to autocorrelation and partial autocorrelation. URL https://tinyurl.com/krmud7tm.

Hyndman, R. and Athanasopoulos, G. *Forecasting: principles and practice, 3rd edition*. OTexts: Melbourne, Australia, 2021. URL OTexts.com/fpp3.

Koenecke, A. Applying deep neural networks to financial time series forecasting. URL https://infosci.cornell.edu/~koenecke/files/Deep_Learning_for_Time_Series_Tutorial.pdf.

Kolle, O. Documentation of the weather station on top of the roof of the institute building of the max-planck-institute for biogeochemistry, 2008. URL https://www.bgc-jena.mpg.de/wetter/Weatherstation.pdf.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539. URL https://doi.org/10.1038/nature14539.

Seabold, S. and Perktold, J. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.