Determining Adaptive Loss Functions and Algorithms for Predictive Models

Michael C. Burkhart(✉) and Kourosh Modarresi

Adobe Inc., 345 Park Avenue, San José, CA 95110, USA
{mburkhar,modarres}@adobe.com

Abstract. We consider the problem of training models to predict sequential processes. We use two econometric datasets to demonstrate how different losses and learning algorithms alter the predictive power for a variety of state-of-the-art models. We investigate how the choice of loss function impacts model training and find that no single algorithm or loss function results in optimal predictive performance. For small datasets, neural models prove especially sensitive to training parameters, including choice of loss function and pre-processing steps. We find that a recursively-applied artificial neural network trained under L_1 loss performs best under many different metrics on a national retail sales dataset, whereas a differenced autoregressive model trained under L_1 loss performs best under a variety of metrics on an e-commerce dataset. We note that different training metrics and processing steps result in appreciably different performance across all model classes and argue for an adaptive approach to model fitting.

1 Introduction

We develop time series estimators for current datasets and use them to iteratively forecast a few steps into the future. We consider the effects of using different training loss functions and different evaluation metrics. The training methodology, including the choice of loss function, seems to impact model performance more than typically reported, potentially due to the relatively smaller amounts of available training data.

2 Datasets

We used time series datasets from the FRED Economic Data portal provided by the Federal Reserve Bank of St. Louis.

2.1 Advance Retail Sales

The Advance Retail Sales: Retail and Food Services Total (RSAFSNA) dataset is a monthly accounting of retail and food services sales totals in the US, provided by the U.S. Bureau of the Census [21]. The data begins in January 1992 and includes

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11537, pp. 595–607, 2019.
https://doi.org/10.1007/978-3-030-22741-8_42

a final estimated value for October 2018. This monthly data is strongly periodic. We split the time series into two contiguous sets: a training set containing 310 observations from Jan. 1992 to Oct. 2017 and a testing set of 12 observations from Nov. 2017 to Oct. 2018. Models learned to use the previous $\ell = 11$ observations to predict the next datapoint. All training and validation was conducted on the training set. See Fig. 1(left) for a plot of the data and Fig. 2 for its autocorrelation and partial autocorrelation plots.

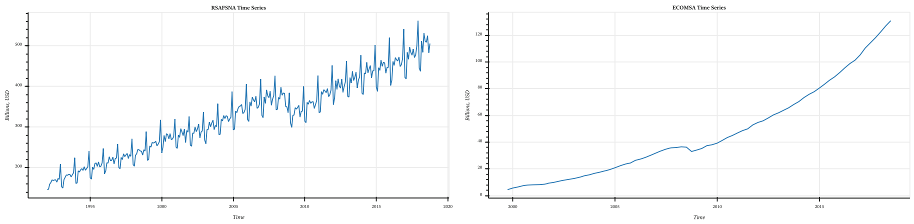

Fig. 1. (Left) Plot of the Advance Retail Sales dataset. Note the strong seasonality for this monthly data. The recession of 2008 is clearly visible in the data. (Right) Plot of the E-Commerce Retail Sales dataset. This coarser, quarterly data displays less seasonality than the Retail Sales numbers. While the 2008 recession remains visible, its relative effect seems less pronounced. Even as total consumption dropped, the share of online consumption increased.

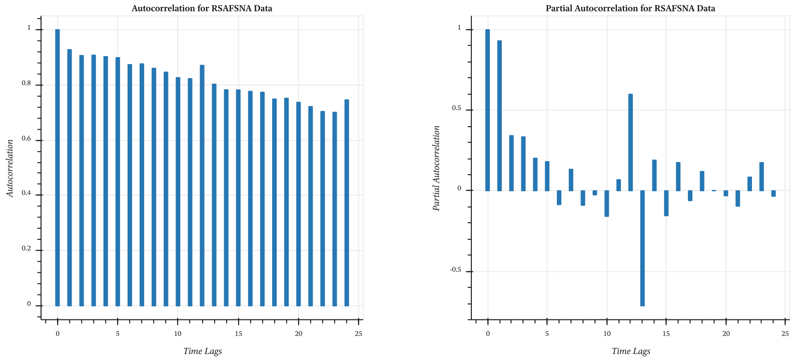

Fig. 2. Diagnostic plots for the Advance Retail Sales dataset. (Left) Autocorrelation function; (Right) Partial Autocorrelation function. The two bumps in the ACF correspond to lags at times 12 and 24, exactly one and two years prior, respectively. Similarly, the bump in the PACF corresponds to the lag at time 12, exactly one year prior.

2.2 E-Commerce Retail Sales

The E-Commerce Retail Sales (ECOMSA) dataset is a quarterly accounting of goods and services sales totals where orders were placed or prices were negotiated online. The U.S. Bureau of the Census provides data from Q4 1999 to Q3

2018 [22]. We split the time series into two contiguous sets: a training set containing 72 observations from Q4 1999 to Q3 2017 and a testing set of 4 observations from Q4 2017 to Q3 2018. Models used the previous 7 observations to predict the next datapoint. All training and validation was conducted on the training set. See Fig. 1(right) for a plot of the data and Fig. 3 for its autocorrelation and partial autocorrelation plots.

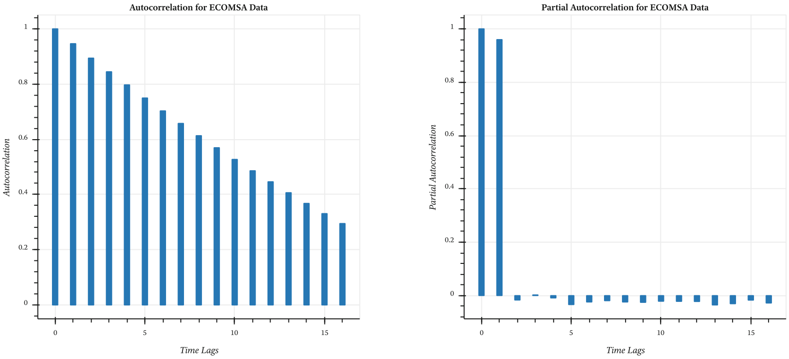

Fig. 3. Diagnostic plots for the E-Commerce Retail Sales dataset. (Left) Autocorrelation function; (Right) Partial Autocorrelation function. Periodicity may be less apparent for this data due to the general strong trend of growth.

3 Methodology

We describe some of the modeling approaches used on this data. Consider a real-valued time series $X_1, X_2, \dots$. Our problem is to use the historical data $X_{t-\ell}, X_{t-\ell+1}, \dots, X_{t-1}$ to predict the next realization of the series X_t. The length ℓ of the historical data used by our model is often referred to as number of lags in the model. Once we have learned a model for X_t given $X_{t-\ell:t-1}$, we can use it iteratively to predict multiple steps into the future by using our predictions in place of known historical data. For example, if we predict $\hat{X}_t$ given $X_{t-\ell:t-1}$, we can then predict $\hat{X}_{t+1}$ given $X_{t-\ell+1:t-1}, \hat{X}_t$. Note however, that the uncertainty associated to this second prediction $\hat{X}_{t+1}$ will be higher than that for $\hat{X}_t$, because we are now using uncertain data to make predictions.

3.1 Autoregressive Model

An autoregressive model with ℓ lags models

$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_\ell X_{t-\ell} + \epsilon_t$$

where $\epsilon_t \sim \mathcal{N}(\mu, \sigma^2)$ for parameters $\alpha_1, \dots, \alpha_\ell, \mu, \sigma$. We often denote such a model $AR(\ell)$. The hyperparameter ℓ is often selected by considering the partial autocorrelation function, that measures the correlation between X_t and X_{t-i} after accounting for the linear dependence of X_t on $X_{t-1}, \dots, X_{t-i+1}$.

3.2 Artificial Neural Network Model

We can use a fully-connected multiple layer artificial neural network $f_\theta : \mathbb{R}^\ell \to \mathbb{R}$ to model

$$X_t = f_\theta(X_{t-1}, \ldots, X_{t-\ell})$$

We learn the parameters θ by passing batches of data to any standard optimizer.

3.3 Long Short-Term Memory Model

A Long Short-Term Memory model [7,9] maintains a stateful representation of history to overcome the vanishing and exploding gradient problems encountered by RNN's when presented with long-term dependencies [2]. See Fig. 4 for an illustration.

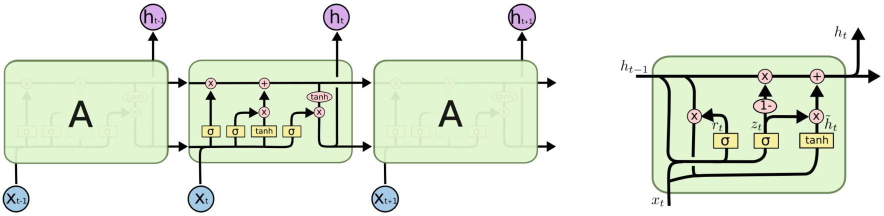

Fig. 4. Schematics for recurrent neural network variants LSTM (left, shown in sequence) and GRU (right, cell only). Images courtesy of Chris Olah; re-used with permission.

3.4 Gated Recurrent Unit Model

A variant of the LSTM, the Gated Recurrent Unit (GRU) model simplifies LSTM architecture [3]. Chung et al. suggested that GRU's can outperform LSTM's on smaller datasets [4]. See Fig. 4 for a side-by-side comparison.

3.5 Temporal Convolutional Network Model

The Temporal Convolutional Network (TCN) model presents an alternative neural network-based approach to time series modeling [1,12]. This convolutional architecture creates connections that are causal (dependent only on previous time steps) and dilated (periodic in time). See Fig. 5 for a schematic. Sometimes such models employ residual blocks [8] that stack networks and include a copy of the first layer in the final layer.

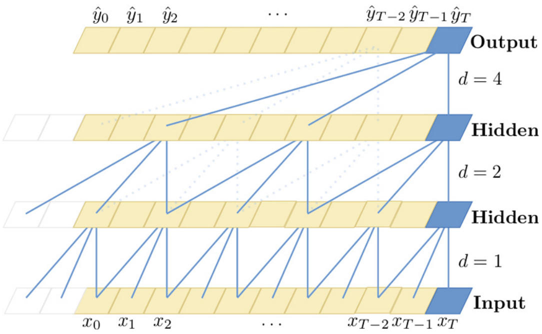
$\hat{y}_0$ $\hat{y}_1$ $\hat{y}_2$ ... $\hat{y}_{T-2}$ $\hat{y}_{T-1}$ $\hat{y}_T$
Output
$d = 4$
Hidden
$d = 2$
Hidden
$d = 1$
Input
x_0 x_1 x_2 ... x_{T-2} x_{T-1} x_T

Fig. 5. Schematic for TCN model. Causal convolutions are those that only go backward in time. Dilated convolutions use only every k-th entry. Stacking dilated convolutions increases the receptive field multiplicatively. Diagram re-used from [1].

3.6 Gaussian Mixture Model

A Gaussian Mixture Model fits the model

$$p(x) = \sum_{i=1}^{k} \varphi_i \cdot \eta(x; \mu_i, \Sigma_i)$$

where $\eta(\cdot; \mu_i, \Sigma_i)$ denotes the p.d.f. of a normal distribution with mean μ_i and covariance Σ_i. This model consists of a mixture of k Gaussian distributions. As both the latent parameters and mixture memberships must be inferred from the dataset, the Expectation Maximization (EM) algorithm is commonly used to fit such a model [5]. Expectation Maximization alternates between computing soft assignments (expectations) of datapoints to mixtures and finding the latent parameters that maximize model likelihood under these assignments. Its estimates converge to the Maximum Likelihood (MLE) parameters [23].

Note that GMM is a generative model. We distinguish generative probabilistic models that learn the joint distribution $p(x, y)$ from discriminative probabilistic models that learn the conditional distribution $p(y|x)$ [14]. For example, we consider most regressive models (including the autoregressive and Gaussian process models described here) to be discriminative. To apply this model for our purposes here, we learned the joint distribution of $(X_{t-\ell:t-1}, X_t)$ as a Gaussian mixture and then for a given test sequence $x_{t-\ell:t-1}$, we predicted

$$\hat{x}_t = \arg \max_{x_t} \{p_\theta(x_{t-\ell:t-1}, x_t)\}$$

under our trained model p_θ.

3.7 Gaussian Process Regression Model

A Gaussian Process (GP) is an infinite collection of random variables for which every finite subset has a multivariate Gaussian distribution [17]. For the purposes of a regression, the GP specifies a covariance structure on function outputs that depends on the function inputs. Given a dataset $\{(x_i, y_i)\}_{i=1}^{n}$, and a kernel function $k_\theta(\cdot, \cdot)$ where θ is a set of tunable hyperparameters, the GP model specifies

$$Y_{1:n}|x_{1:n} \sim \mathcal{N}(\mathbf{0}, K + \sigma^2 I_n)$$

where $K_{ij} = k_\theta(x_i, x_j)$ for each $1 \leq i, j \leq n$, and σ^2 is a tunable hyperparameter for process noise. Learning amounts to finding the hyperparameters θ, σ^2 that maximize the likelihood for our data under this model. We made predictions at a new test point x_* by leveraging the consistency of our stochastic model. We consider

$$\begin{bmatrix} Y_{1:n} \\ Y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_\theta + \sigma^2 I_n & (\boldsymbol{k}_*)^\top \\ \boldsymbol{k}_* & k_{**} \end{bmatrix}\right)$$

where $(\boldsymbol{k}_*)_i = k_\theta(x_i, x_*)$ and $k_{**} = k_\theta(x_*, x_*)$. This implies that

$$Y_*|x_{1:n}, Y_{1:n}, x_* \sim \mathcal{N}\big((\boldsymbol{k}_*)^\top K^{\ -1} Y_{1:n}, k_{**} - (\boldsymbol{k}_*)^\top K^{\ -1} \boldsymbol{k}_*\big)$$

We used the mean of this conditional distribution for predictive purposes, with a squared exponential kernel.

4 Results

We tuned the artificial neural networks by hand. The following choices were made for each network:

- Model architecture: For recurrent neural networks, the size of the hidden layer and whether to include fully connected layers above or below the recurrent layer can significantly alter model performance.
- Regularization: Common approaches include dropout [18], added Gaussian noise [15], batch normalization [10], and Tikhonov regularization [19,20].
- Optimizer: There are a variety of modern optimization strategies that adaptively vary learning rates and add momentum. For this project, we used the Adam optimizer [11].
- Optimization parameters: Learning rate and training batch size were hand-selected.
- Synergistic effects: Choices in one category impact choices in another. For example, learning rate and training batch size are interdependent: adjustments to learning rate should often accompany adjustments to batch size, and vice versa. While the literature presents many individual solutions for deep learning, less is known about how they interact. For example, there has been considerable discussion on where to place dropout in a recurrent architecture [6,16,24]. In TCN's, we saw a clear connection between architecture size and learning rate as well.

For all neural models, we preprocessed data by subtracting the training mean and dividing by the training standard deviation for each datapoint. We trained and predicted on these scaled data, and inverted the transformation before calculating prediction error.

For the models denoted "Δ," we learned and predicted on the time series of differences, and then took a cumulative sum of predicted differences to tabulate our final predictions. Differencing can remove time-based effects in the data (non-stationarities).

4.1 Advance Retail Sales

The AR predictions appear to be overly conservative. Training under the L_1 objective loss results in uniformly worse performance than L_2 (for the non-differenced model) (Fig. 6).

The GRU and LSTM architectures generally perform less well than the ANN architecture (Fig. 7).

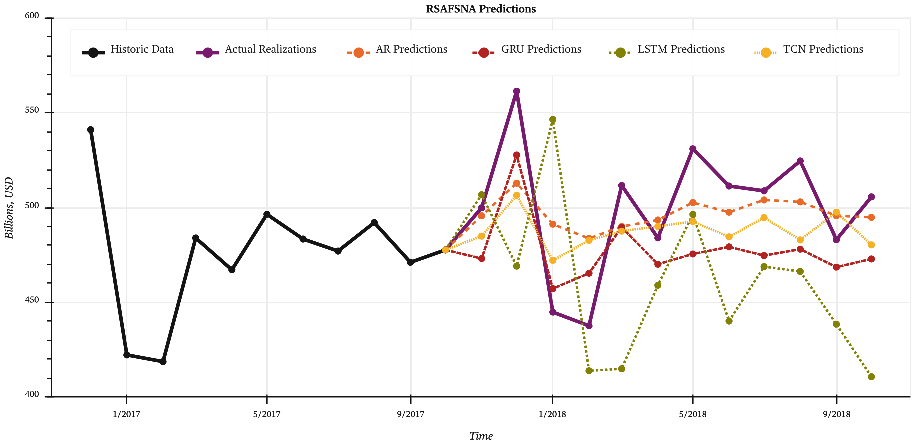

Fig. 6. Model predictions for Advance Retail Sales data on models trained to predict X_t from $X_{t-\ell:t-1}$ For clarity, we only include the L_2 training runs. Full numerical results are in Table 1.

Due to the way we tested the models, a single missed prediction feeds back into the model and compounds the error for subsequent results. For example, the LSTM model badly misses its second prediction and then appears to be off by a time step for the rest of the test. These models have not been given the month of the year as input, only the 11 previous values for retail sales. We see how a single poor prediction that is fed back into the model as input can disrupt the model's sense of seasonality.

Differencing proved especially effective for the recurrent neural network models.

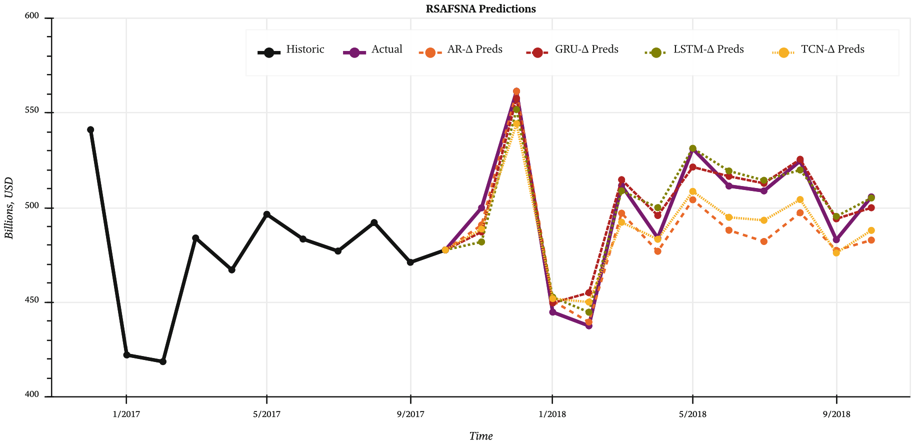

Fig. 7. Model predictions for Advance Retail Sales data on models trained to predict ΔX_t from $\Delta X_{t-\ell:t-1}$, reconstructed from last training point and predicted differences. For clarity, we only include the L_2 training runs. Full numerical results are in Table 1.

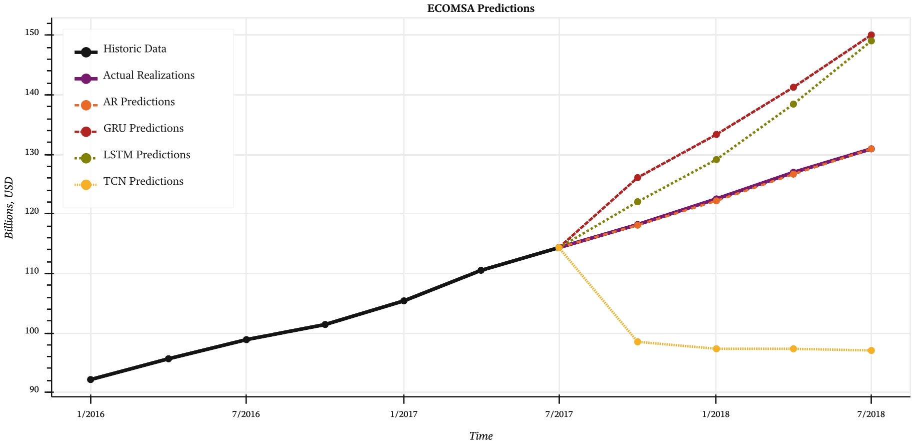

Fig. 8. Model predictions for E-Commerce Retail Sales data on models trained to predict X_t from $X_{t-\ell:t-1}$; For clarity, we only include the L_2 training runs. Full numerical results are in Table 2.

4.2 E-Commerce Retail Sales

For L_2, we used least squares optimization, for which the optimal parameter values have explicit solutions. For L_1 and L_4, we performed gradient descent with momentum to find parameters that minimized training error on the full training set. We initialized at the L_2 optimum and checked for convergence.

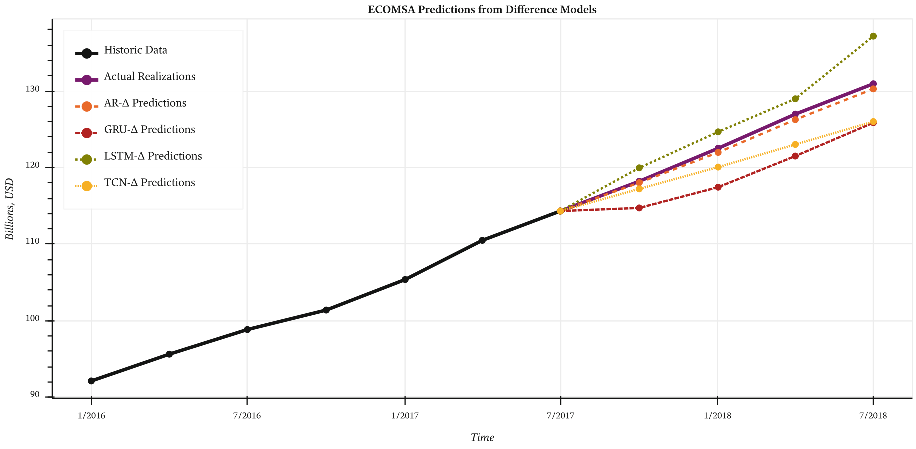

Fig. 9. Model predictions for E-Commerce Retail Sales data on models trained to predict ΔX_t from $\Delta X_{t-\ell:t-1}$, reconstructed from last training point and predicted differences. For clarity, we only include the L_2 training runs. Full numerical results are in Table 2.

Finally, it is worth noting that the performance of the TCN (non-differenced) on this data set was extremely poor (for the non-differenced model), under all objective functions. For any metric we consider, it would be much better to use the null predictions than the TCN predictions. It is possible that the TCN architecture requires a different regularization or training strategy in this case of a very small dataset. One might imagine that the TCN is reverting to the mean of the training data. (We note in Fig. 1 the strong upward trend of this data.) In the future, it may be advisable to include a separate linear trend for datasets like this, and fit the neural networks to residuals. Differencing appears to ameliorate this issue (Figs. 8 and 9).

For the AR model, differencing effectively increases the lag by one time step and removes any linear dependence on time. We see fairly similar results for the differenced and non-differenced versions of the AR and ANN models. However, for the recurrent models (LSTM & GRU) and the TCN, taking differences can prove to be a very effective way to boost model performance.

A Implementation Details

For this project, we used the Miniconda v.4.5.11 package manager with Python v.3.6.7. All neural networks were constructed in Keras v.2.2.4 with a Tensorflow v1.12.0 backend running Intel MKL optimizations.

Table 1. Normalized Model Performance on the Advance Retail Sales dataset. Models were trained to optimize the loss function "obj." (for objective function). Here, "n" denotes the quantity has been normalized by dividing out the norm returned from predicting the final training value for all test data, "Cos." denotes cosine distance, "Corr." denotes Pearson correlation distance, and "Mah." denotes Mahalanobis distance [13].

Model	obj.	n.-MAE	n.-RMSE	n.-L_3	n.-L_4	n.-L_∞	n.-Cos.	MAPE	Corr.	Mah.
AR	L_1	0.828	0.921	0.963	0.97	0.919	1.138	0.06	0.955	6.467
AR-Δ	L_1	0.326	0.351	0.35	0.34	0.278	0.069	0.022	0.033	4.897
AR	L_2	0.642	0.681	0.694	0.689	0.579	0.668	0.045	0.146	3.894
AR-Δ	L_2	0.41	0.435	0.433	0.421	0.326	0.122	0.028	0.062	4.803
AR	L_4	0.638	0.663	0.664	0.656	0.593	0.608	0.045	0.184	4.26
AR-Δ	L_4	0.604	0.6	0.583	0.562	0.447	0.187	0.041	0.094	5.244
ANN	L_1	0.198	0.204	0.204	0.2	0.18	0.039	0.014	0.017	3.998
ANN-Δ	L_1	0.281	0.296	0.293	0.284	0.22	0.06	0.019	0.03	5.209
ANN	L_2	0.261	0.272	0.274	0.271	0.242	0.065	0.018	0.03	3.594
ANN-Δ	L_2	0.288	0.299	0.294	0.285	0.223	0.065	0.02	0.033	4.824
ANN	L_4	0.636	0.643	0.638	0.626	0.583	0.2	0.043	0.1	4.971
ANN-Δ	L_4	0.316	0.33	0.328	0.319	0.256	0.142	0.022	0.073	4.88
GRU	L_1	1.056	1.055	1.061	1.069	1.085	1.076	0.072	1.473	5.682
GRU-Δ	L_1	0.384	0.367	0.351	0.336	0.277	0.076	0.026	0.034	4.668
GRU	L_2	0.843	0.793	0.761	0.735	0.663	0.443	0.058	0.212	6.044
GRU-Δ	L_2	0.216	0.221	0.222	0.22	0.209	0.068	0.015	0.028	5.076
GRU	L_4	0.703	0.694	0.674	0.654	0.604	0.643	0.05	0.396	6.059
GRU-Δ	L_4	0.489	0.476	0.459	0.442	0.37	0.063	0.034	0.03	4.408
LSTM	L_1	1.066	1.027	0.991	0.956	0.809	0.336	0.074	0.137	6.0
LSTM-Δ	L_1	0.281	0.276	0.268	0.259	0.225	0.061	0.019	0.025	4.291
LSTM	L_2	1.652	1.635	1.593	1.538	1.214	2.761	0.115	1.04	6.898
LSTM-Δ	L_2	0.222	0.234	0.236	0.235	0.215	0.078	0.016	0.036	3.948
LSTM	L_4	0.441	0.526	0.583	0.614	0.642	0.332	0.032	0.183	4.828
LSTM-Δ	L_4	0.218	0.258	0.285	0.298	0.309	0.085	0.015	0.043	5.735
TCN	L_1	0.915	0.917	0.911	0.898	0.793	1.182	0.064	1.108	6.752
TCN-Δ	L_1	0.321	0.3	0.284	0.27	0.226	0.084	0.022	0.009	3.651
TCN	L_2	0.798	0.775	0.755	0.734	0.656	0.739	0.055	0.376	5.108
TCN-Δ	L_2	0.401	0.38	0.358	0.339	0.268	0.1	0.028	0.02	3.786
TCN	L_4	1.004	0.991	0.977	0.962	0.916	0.902	0.069	0.57	6.123
TCN-Δ	L_4	0.411	0.396	0.377	0.359	0.29	0.142	0.028	0.017	4.126
GMM	MLE	0.219	0.22	0.218	0.214	0.203	0.063	0.015	0.028	4.516
GMM-Δ	MLE	0.225	0.233	0.233	0.228	0.196	0.042	0.016	0.02	4.829
GPR	MLE	0.729	0.674	0.627	0.588	0.435	0.097	0.051	0.034	6.062
GPR-Δ	MLE	0.209	0.237	0.262	0.279	0.299	0.082	0.015	0.025	4.203

Table 2. Normalized Model Performance on the E-Commerce Retail Sales dataset. Models were trained to optimize the loss function "obj." (for objective function). Here, "n" denotes the quantity has been normalized by dividing out the norm returned from predicting the final training value for all test data, "Cos." denotes cosine distance, "Corr." denotes Pearson correlation distance, and "Mah." denotes Mahalanobis distance.

Model	obj.	n.-MAE	n.-RMSE	n.-L_3	n.-L_4	n.-L_∞	n.-Cos.	MAPE	Corr.	Mah.
AR	L_1	0.058	0.056	0.054	0.053	0.049	0.002	0.005	0.001	0.676
AR-Δ	L_1	0.015	0.014	0.013	0.012	0.011	0.001	0.001	0.0	0.088
AR	L_2	0.017	0.02	0.02	0.02	0.019	0.001	0.001	0.0	0.432
AR-Δ	L_2	0.051	0.05	0.049	0.048	0.043	0.002	0.004	0.0	0.472
AR	L_4	0.02	0.027	0.031	0.033	0.036	0.003	0.002	0.0	0.383
AR-Δ	L_4	0.069	0.071	0.073	0.074	0.077	0.006	0.006	0.0	0.593
ANN	L_1	0.283	0.281	0.285	0.29	0.305	0.062	0.023	0.003	0.847
ANN-Δ	L_1	0.082	0.083	0.083	0.083	0.079	0.007	0.007	0.0	0.36
ANN	L_2	0.206	0.193	0.185	0.18	0.171	0.011	0.017	0.005	0.663
ANN-Δ	L_2	0.215	0.221	0.224	0.225	0.226	0.055	0.017	0.0	1.038
ANN	L_4	0.614	0.645	0.64	0.632	0.593	0.784	0.051	0.152	5.266
ANN-Δ	L_4	0.377	0.396	0.404	0.407	0.412	0.21	0.031	0.0	1.58
GRU	L_1	0.724	0.681	0.659	0.647	0.635	0.123	0.059	0.001	0.692
GRU-Δ	L_1	0.778	0.839	0.879	0.905	0.957	0.909	0.063	0.007	2.434
GRU	L_2	1.263	1.204	1.176	1.162	1.153	0.491	0.104	0.002	1.361
GRU-Δ	L_2	0.462	0.425	0.402	0.386	0.33	0.019	0.038	0.007	1.976
GRU	L_4	1.705	1.629	1.592	1.576	1.567	0.872	0.14	0.003	1.791
GRU-Δ	L_4	0.117	0.12	0.12	0.12	0.111	0.015	0.01	0.001	0.507
LSTM	L_1	1.25	1.208	1.19	1.183	1.182	0.642	0.102	0.001	1.678
LSTM-Δ	L_1	0.248	0.242	0.237	0.233	0.223	0.05	0.021	0.012	2.538
LSTM	L_2	0.97	1.002	1.028	1.047	1.093	0.959	0.079	0.006	2.06
LSTM-Δ	L_2	0.293	0.312	0.332	0.347	0.375	0.129	0.024	0.016	1.904
LSTM	L_4	2.677	2.836	3.01	3.142	3.387	7.211	0.218	0.11	5.676
LSTM-Δ	L_4	0.732	0.922	1.029	1.088	1.178	1.903	0.059	0.032	2.898
TCN	L_1	2.978	2.749	2.61	2.523	2.299	1.554	0.246	1.999	3.625
TCN-Δ	L_1	0.284	0.288	0.29	0.29	0.287	0.084	0.023	0.0	1.132
TCN	L_2	2.625	2.428	2.308	2.234	2.041	1.284	0.216	1.874	3.07
TCN-Δ	L_2	0.299	0.301	0.302	0.301	0.297	0.088	0.024	0.0	1.219
TCN	L_4	3.148	2.904	2.756	2.662	2.406	1.723	0.26	1.968	3.627
TCN-Δ	L_4	0.332	0.335	0.337	0.337	0.335	0.112	0.027	0.0	1.287
GMM	MLE	0.02	0.021	0.021	0.022	0.021	0.001	0.002	0.001	0.425
GMM-Δ	MLE	0.026	0.029	0.031	0.033	0.035	0.005	0.002	0.001	0.177
GPR	MLE	0.19	0.201	0.209	0.214	0.225	0.052	0.015	0.001	0.499
GPR-Δ	MLE	0.345	0.336	0.33	0.327	0.322	0.072	0.028	0.0	1.709

References

1. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. ArXiv e-prints (2018). arXiv:1803.01271
2. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
3. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Conference on Empirical Methods in Natural Language Processing, pp. 1724–1734 (2014)
4. Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS Workshop on Deep Learning (2014)
5. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J. R. Stat. Soc. Ser. B (Methodol.) **39**(1), 1–38 (1977)
6. Gal, Y., Ghahramani, Z.: A theoretically grounded application of dropout in recurrent neural networks. In: Advances in Neural Information Processing Systems, pp. 1019–1027 (2016)
7. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with lstm. Neural Comput. **12**(10), 2451–2471 (2000)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
10. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, vol. 37, pp. 448–456 (2015)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
12. Lea, Colin, Vidal, René, Reiter, Austin, Hager, Gregory D.: Temporal convolutional networks: a unified approach to action segmentation. In: Hua, Gang, Jégou, Hervé (eds.) ECCV 2016. LNCS, vol. 9915, pp. 47–54. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49409-8_7
13. Mahalanobis, P.C.: On the generalized distance in statistics. Proc. Nat. Inst. Sci. India **2**(1), 49–55 (1936)
14. Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. In: Advances in Neural Information Processing Systems, pp. 841–848 (2002)
15. Noh, H., You, T., Mun, J., Han, B.: Regularizing deep neural networks by noise: its interpretation and optimization. In: Advances in Neural Information Processing Systems, pp. 5109–5118 (2017)
16. Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout improves recurrent neural networks for handwriting recognition. In: International Conference on Frontiers in Handwriting Recognition, pp. 285–290 (2014)
17. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)

19. Tikhonov, A.N.: On the stability of inverse problems. Doklady Akademii Nauk SSSR **39**(5), 195–198 (1943)
20. Tikhonov, A.N.: Solution of incorrectly formulated problems and the regularization method. Doklady Akademii Nauk SSSR **151**(3), 501–504 (1963)
21. U.S. Bureau of the Census: Advance retail sales: Retail and food services, total [RSAFSNA] dataset. FRED, Federal Reserve Bank of St. Louis (2018)
22. U.S. Bureau of the Census: E-commerce retail sales [ECOMSA] dataset. FRED, Federal Reserve Bank of St. Louis (2018)
23. Wu, C.F.J.: On the convergence properties of the EM algorithm. Annal. Stat. **11**(1), 95–103 (1983)
24. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. ArXiv e-prints http://arxiv.org/abs/1409.2329 (2014)