Generalization in fully-connected neural networks for time series forecasting

Anastasia Borovykh [1] **Sander Bohte** [1] **Cornelis W. Oosterlee** [1 2]

Abstract

In this work we study the generalization capabilities of fully-connected neural networks trained in the context of time series forecasting. Time series do not satisfy the typical assumption in statistical learning theory of the data being i.i.d. samples from some data-generating distribution. We use the input and weight Hessians, that is the smoothness of the learned function with respect to the input and the width of the minimum in weight space, to quantify a network's ability to generalize to unseen data. While such generalization metrics have been studied extensively in the i.i.d. setting of for example image recognition, here we empirically validate their use in the task of time series forecasting. Furthermore we discuss how one can control the generalization capability of the network by means of the training process using the learning rate, batch size and the number of training iterations as controls.

1. Introduction

Forecasting time series is an exceptionally difficult task due to the risk of overfitting on the dataset, in particular in the case of overparametrized networks (Zhang et al., 2016), (Zhang et al., 1998). In other words, when using the past to predict the future one has to be certain to have succeeded in extracting a signal from the past that will propagate to the future, and not simply fitted a complex function on the past. Neural networks, while being powerful function approximators that are relatively easy to optimize, can lead to poor extrapolation in time series forecasting due to the latter. Due to their ability to approximate almost any function it is of the essence to ensure that the network is learning the signal of interest instead of the noise (Zhang et al., 2016)(Geirhos et al., 2018).

There are different ways of measuring the learning capability, or complexity, of a neural network (Bartlett et al., 2017), (Neyshabur et al., 2015), (Li et al., 2018) (Arora et al., 2018). In this work we focus on sensitivities with respect to the input and weights. The output sensitivity with respect to the inputs has been succesfully used as a metric for generalization capabilities in an i.i.d. setting (Novak et al., 2018). It has also been proposed that the Hessian with respect to the weights can be used as a measure for generalization (Hochreiter & Schmidhuber, 1997) (Smith & Le, 2018) (Sagun et al., 2018). The model complexity can also be influenced using the training algorithm by biasing the model into configurations that are more robust to noise (Arora et al., 2019) (Gunasekar et al., 2018)(Neyshabur et al., 2017). In particular, it has been observed that certain parameters of stochastic gradient descent (SGD) can be used to control the generalization error and data fit (Jastrzkebski et al., 2017) (Seong et al., 2018) (Chaudhari & Soatto, 2018) (Li et al., 2019).

The novelty of our contribution consists in a thorough empirical analysis of what the capability to generalize means for time series forecasting with fully-connected neural networks. While generalization capabilities for i.i.d. datasets (e.g. images) have been studied extensively, the problem is more complex for the time series: the dataset is can be much smaller, the signal-to-noise ratio might be low and the distribution can be non-stationary. Understanding what it means for a neural network to have good generalizibility, i.e. learning a consistently present pattern instead of overfitting on noise or on a changing pattern, and how this can be achieved through the learning algorithms will be the main task of this paper. In the first part of this work we present some theoretical insights regarding generalization and the input and weight Hessians, while in the second part numerical results for various artificial and real-world time series are presented.

2. Generalization

Let the inputs to the neural network be given by $x \in \mathbb{R}^{n_0}$. Let $W^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$ be the weight matrix in layer l with element $w_{i,j}^{(l)}$ connecting neuron i in layer l and j in layer $l-1$. Define w as the vectorized total weights in the network, so that $w \in \mathbb{R}^d$ with $d = n_0 n_1 + n_1 n_2 + ... + n_{L-1} n_L$, with d thus being the dimension of the weight space. In the rest of this paper the dimension $\mathbb{R}^d$ refers to *column* vectors.

[1]CWI Amsterdam, the Netherlands [2]TU Delft, the Netherlands. Correspondence to: Anastasia Borovykh <anastasia.borovykh@cwi.nl>.

ICML 2019 Time Series Workshop, Long Beach, California, 2019.

Each layer $l = 1, ..., L$ outputs,

$$a^{(l)} = f(z^{(l)}) = f\left(W^{(l)}a^{(l-1)}\right),$$

where $f(\cdot)$ is the non-linear activation function, in the first layer $a^0 = x$ and the network output is given by $\hat{y}(x, w) = z^L$. The network is trained by optimizing the mean squared error (MSE) loss, $\hat{\mathcal{L}}(x, w, y) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}(x^i, w) - y^i\right)^2$, where (x^i, y^i) for $i = 1, ..., N$ is the train dataset and $\hat{y}(x^i, w)$ the neural network output. Generalization is the relationship between a trained networks' performance on train data versus its performance on test data. This is a highly desirable property for neural networks, where ideally the performance on the train data should be similar to the performance on similar but unseen test data. In general, the generalisation error of a neural network model $\hat{y}(x, w)$ can be defined as the failure of the hypothesis $\hat{y}(x, w)$ to explain the dataset sample. It is measured by the discrepancy between the true error $\mathcal{L}$ and the error on the sample dataset $\hat{\mathcal{L}}$, i.e. $\mathcal{L}(x, w, y) - \hat{\mathcal{L}}(x, w, y)$. This can also be cast in terms of noise robustness, i.e. making sure the network is not overfitting on noise. In this case we require, for some input perturbation $\epsilon \in \mathbb{R}^{n_0}$, the change in the loss function to be small,

$$|\mathcal{L}(x + \epsilon, w, y) - \mathcal{L}(x, w, y)| < \delta. \tag{1}$$

2.1. The weight Hessian

The first metric for the generalization capabilities will be the weight Hessian of the loss function with respect to the weights $H^w(\mathcal{L}) \in \mathbb{R}^{d\times d}$ which has elements $h^w_{ij} = \partial_{w_i}\partial_{w_j}\mathcal{L}(x, w, y)$.

Controlling the weight Hessian It has been mentioned in prior research (Jastrzkebski et al., 2017), (Mandt et al., 2017), (Smith & Le, 2018), (Chaudhari & Soatto, 2018) that a relationship exists between the test error and the learning rate and batch size used in the SGD updating scheme with batch size M. Under an i.i.d. assumption on the data we can envoke the central limit theorem to rewrite the updates of SGD as, $w_{t+1} = w_t - \eta g - \frac{\eta}{\sqrt{M}}\epsilon$, where $\epsilon \sim \mathcal{N}(0, K)$ with $\epsilon \in \mathbb{R}^d$. If convergence has been reached, i.e. (1) holds, by a (second-order) Taylor expansion method for the loss function evaluated on the full training data we can obtain an expression for the weight Hessian,

$$H^w(\mathcal{L}(w_t)) \approx \frac{\delta + 2\nabla_w\mathcal{L}(w_t)^T\left(g - \frac{1}{\sqrt{M}}\epsilon\right)}{\eta\left|g - \frac{1}{\sqrt{M}}\epsilon\right|^2},$$

where $\nabla_w\mathcal{L}(w) \in \mathbb{R}^d$ denotes the gradient of the total loss. From this expression we see that the Hessian at convergence is small if a large learning rate or a small batch size has been used.

2.2. The input Hessian

Besides the weight Hessian, the input Hessian will also be used as a metric for out-of-sample performance. This measures the sensitivity of the output function with respect to the changes in the input data, such that a Hessian with small eigenvalues means a smoother output function. Define the elements of the input Hessian $H^x(\mathcal{L}) \in \mathbb{R}^{n_0\times n_0}$ of an input $x \in \mathbb{R}^{n_0}$ as $h^x_{ij} = \partial_{x_i}\partial_{x_j}\mathcal{L}(x, w, y)$. The Hessian is averaged over the data samples x^i, $i = 1, ..., N$ in the train dataset in order to obtain an average sensitivity metric over the input space.

Relation between input and weight Hessian The output of a two-layer neural network with added input noise can be written as,

$$\begin{aligned}\hat{y}(x + \epsilon, w) &= W^{(2)}f((W^{(1)} + \tilde{\epsilon})x) \\ &= (W^{(2)} + \tilde{\epsilon}^2)f((W^{(1)} + \tilde{\epsilon}^1)x),\end{aligned}$$

for $\tilde{\epsilon} = W^{(1)}\epsilon x^T/||x||_2^2$, and some $\tilde{\epsilon}^2 \in \mathbb{R}^{1\times n_1}$ and such that $\tilde{\epsilon}^1 < \tilde{\epsilon}$. In other words, noise in the input can be rewritten as noise in the weights. Let $\hat{\epsilon}^1$, $\hat{\epsilon}^2$ be the vectorized forms of $\tilde{\epsilon}^1$, $\tilde{\epsilon}^2$. Then,

$$\begin{aligned}&\mathcal{L}(x + \epsilon, w, y) \\ &= \mathcal{L}(x, w, y) + [(\hat{\epsilon}^1)^T, (\hat{\epsilon}^2)^T]\nabla_w\mathcal{L}(x, w, y) \\ &+ \frac{1}{2}[(\hat{\epsilon}^1)^T, (\hat{\epsilon}^2)^T]H^w(\mathcal{L}(x, w, y))[(\hat{\epsilon}^1)^T, (\hat{\epsilon}^2)^T]^T.\end{aligned}$$

From this expression we see that if the Hessian with respect to $W^{(1)}$ is not sufficiently small, for a deeper network the remaining noise can be damped by sufficient flatness of the loss function in the directions of $W^{(2)}$.

The effects of SGD Consider now a first-order Taylor expansion in x for some noise $\tilde{\epsilon}$

$$\begin{aligned}&\nabla_w\mathcal{L}\left(x + \frac{1}{\sqrt{M}}\tilde{\epsilon}, w, y\right) \\ &\approx \nabla_w\mathcal{L}\left(x, w, y\right) + \frac{1}{\sqrt{M}}\nabla_w\tilde{\epsilon}^T\nabla_x\mathcal{L}\left(x, w, y\right) \\ &=: \nabla_w\mathcal{L}\left(x, w, y\right) + \frac{1}{\sqrt{M}}\epsilon.\end{aligned}$$

In other words, the SGD update rule can be rewritten as, $w_{t+1} \approx w_t - \eta\tilde{g}$, where

$$\tilde{g} := \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\nabla_w\mathcal{L}\left(x + \frac{1}{\sqrt{M}}\tilde{\epsilon}, w, y\right)\right].$$

Therefore, the noise from the stochastic gradient descent can be related to noise in the input and SGD can be interpreted to minimize a jittered cost function. In turn, by a derivation similar to (Reed et al., 1992), a jittered cost function can be related to optimizing a regularized loss function, so that SGD can be seen to impose smoothness assumptions on the output function with respect to the input.

2.3. How low can we go?

As was shown in the previous work, under certain – albeit restrictive (i.e. i.i.d. data) – assumptions on the deep neural network, the loss surface is given by a Gaussian random field on a high-dimensional space (Choromanska et al., 2015). For the high-dimensional Gaussian random fields, as shown in (Becker et al., 2018), the lower the train loss the lower the entropy and thus the sharper the minima. In other words, wider minima lie at higher loss values in the loss surface. A higher complexity solution, i.e. one with lower entropy, can also be expected to be less robust to noise. Optimization in neural networks can then be seen as finding an optimal trade-off between the data fit (i.e. how low in the loss surface we are) and the solution complexity (i.e. how high the entropy in loss space is, or how robust the solution is to noise). This trade-off is sometimes referred to as the information bottleneck (Achille & Soatto, 2018).

Dependence on noise Consider a time series y^i, $i = 0, ..., N$ with a signal to noise ratio of $(1-\alpha) : \alpha$. Suppose the neural network output should be resistant to noise in the signal. In the non-i.i.d. setting also to certain patterns present in one part of the series but not in another. Here, we mostly focus on robustness to noise, and assume that the non-i.i.d. property comes from the time dependence and the noise which can change in distribution. In this case the loss function should satisfy the following objective, $|\mathcal{L}(x + \alpha\epsilon, w, y) - \mathcal{L}(x, w, y)| < \delta$. Relating this to the train and test set, we assume that $x + \alpha\epsilon$ is the test data with a noise component different from that in the train data x. By a Taylor expansion over the input variable x one obtains after taking expected values,

$$\mathbb{E}[\mathcal{L}(x + \alpha\epsilon, w, y) - \mathcal{L}(x, w, y)] \approx \frac{1}{2}\alpha^2 Tr\left(H^x(\mathcal{L}(x, w, y))\right),$$

where we have used the fact that $\epsilon_i \sim \mathcal{N}(0, 1)$ i.i.d.. From this it follows that, $Tr\left(H^x(\mathcal{L}(x, w, y))\right) = 2\frac{\delta}{\alpha^2}$. In other words, the amount of noise in the input the neural network has to be resistant to is inversely proportional to the input (and thus weight) Hessian.

Obtaining better generalizable minima As discussed in the previous sections, certain hyperparameters (see e.g. (Jastrzkebski et al., 2017)) can be used to control the trade-off between generalization and complexity. In particular, the learning rate can result in wider minima due to a higher noise coefficient in SGD (as shown in Section 2.1); similarly the batch size results in a higher amount of noise in SGD, and can be related to a smaller weight Hessian; lastly, the number of iterations determines how low we go in the loss surface, and by a theoretical derivation in (Becker et al., 2018) it can be shown that higher points in the loss surface have higher entropy and thus can be more robust to noise. We will study these effects in more detail in the numerical section.

3. Numerical results

As metrics we will use the traces of the weight and input Hessians, where in the case of the input Hessian the value is averaged over the samples in the train dataset.

3.1. Artificial data

We consider a network of 10 hidden layers with 500 nodes per layer and 10.000 iterations are used; the network input consists of $(y_{i-4}, ..., y_i)$ and is trained to forecast y_{i+1}. The dataset consists of 100 samples.

Random noise An overparametrized neural network can fit random noise almost perfectly. In Figure 1 we show that as the MSE decreases, the input and weight Hessians increase. In other words, as expected, these metrics measure the complexity of the solution.

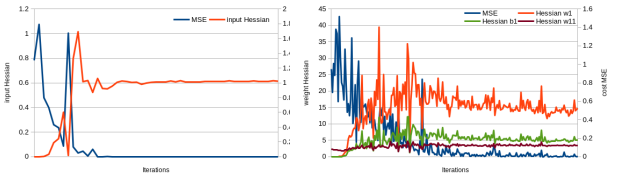

Figure 1. The convergence of the network for the input Hessian (R) and the weight Hessian (R) on random noise data; the loss decreases with iterations, but the input and weights Hessians increase; these can indicate when a network starts overfitting on the noise and be used to make a trade-off between data fit and complexity.

Sine with noise Consider now the function $y_i = \sin(0.1t_i) + c\epsilon_i$, with $t_i \in \{0, 1, ..., 100\}$ and $\epsilon_i \sim \mathcal{N}(0, 1)$. We consider the effects of the hyperparameters on the generalization error. From Figure 2 we observe that a smaller number of iterations or a lower batch size results in general in wider minima, which in turn give a smaller generalization error. In Figure 3 we plot the test error for the learning rate. Using a larger learning rate results in wider minima while a smaller learning rate tends to converge to sharper minima. While the Hessian is correlated with the test error, a too high learning rate can also underfit the data. [1]

3.2. Real-world data

We will use a network with 2 hidden layers and 100 nodes per layer.

Index forecasting Financial data is highly non-linear, non-stationary and has a very low signal-to-noise ratio. The input data will consist of $n = 5$ historical daily absolute

[1] We used batch gradient descent and scaled the gradient by its L_2 norm in order to avoid the gradient size influencing the minima width.

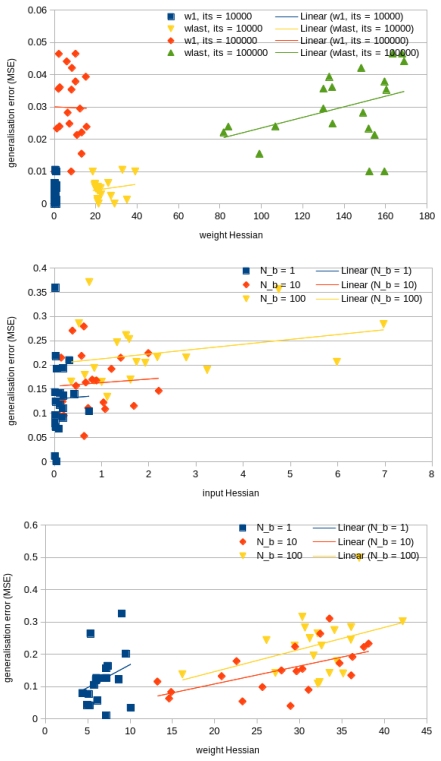

Figure 2. The influence of the hyperparameters, number of iterations for $c = 0.1$ on the weight Hessian (T) and batch size for $c = 0.3$ on the input (CB) and weight (B) Hessian on the generalization error. As expected, a smaller number of iterations or a lower batch size results in lower Hessians which in turn result in lower generalization error.

returns of the S&P500 index, the CBOE 10 year interest rate, and the volatility index. In Table 1 we present the MSE and hit rate (computed as the number of up or down movements predicted correctly) for different hyperparameters averaged over 20 sampled networks. Financial returns are highly noisy and non-linear and distinguishing the signal in the data from noise remains challenging. Nevertheless the results show that the number of iterations and the batch size seem to be able to bias the model into minima that have more (additive) noise resistance and the Hessians seem to indicate the model complexity and thus noise resistance.

Table 1. **The MSE and hit rate for different batch sizes and iteration numbers for financial data.** The batch size and number of iterations seem to be able to control the MSE and hit rate. The input Hessian and the weight Hessian (here the sum of first and last layer traces) correspond with the errors.

N_{it}	N_b	MSE	HIT RATE	$Tr(H^x)$	$Tr(H^W)$
10000	100	2.80	0.49	0.73	50.76
5000	100	2.65	0.513	0.73	50.34
10000	300	2.76	0.500	0.83	52.07
5000	300	2.71	0.505	0.74	51.1

Weather forecasting Here we train a network for predicting the daily minimum temperature in Melbourne, Australia.

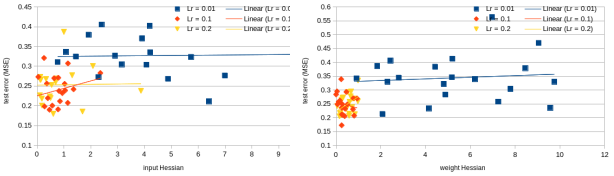

Figure 3. The influence of the learning rate on the generalization error for $c = 0.3$ and the input (T) and weight (B) Hessian. A smaller learning rate results in higher Hessians, however the effect of the learning rate on test set performance seems less visible than that of the other hyperparameters.

The input data will consist of $n = 20$ historical daily obervations of the temperature. The results for the MSE for different training hyperparameters are presented in Table 2 (again averaged over 20 networks). As expected, training with fewer iterations and using smaller batch sizes results in a smoother output function with respect to the input and causes the training algorithm to converge to wider minima.

Table 2. **The MSE for different batch sizes and iteration numbers for weather data.** A higher trace of the input or weight Hessian corresponds to a worse test set MSE due to overfitting on the noise. As usual, training longer and using larger batch sizes resuls in more overfitting.

N_{it}	N_b	MSE	$Tr(H^x)$	$Tr(H^{W^{(1)}})$	$Tr(H^{W^{(3)}})$
10000	10	0.44	0.078	6.05	34.9
5000	10	0.36	0.023	5.85	27.5
10000	100	0.49	0.11	36.7	40.2
5000	100	0.36	0.030	38.4	31.9
10000	200	0.50	0.10	42.6	40.9
5000	200	0.37	0.032	56.7	31.7

4. Conclusion

In this work we studied the generalization capabilities of neural networks trained for the purpose of time series forecasting. We showed that there is a correspondence between good generalization capability and small traces of the input and weight Hessians of the loss function at the minima found after training. We showed that the learning rate, the batch size and the number of iterations used in the training algorithm to bias the network into minima that posess a certain structure. The typical assumption in statistical learning theory of having i.i.d. samples from some data-generating distribution does not hold in time series, and while in this work we gave *empirical* insight into the performance on non-i.i.d. data it will be of interest to derive *theoretically* the loss surface structure for non-i.i.d. data (e.g. similar to (Choromanska et al., 2015)) and studying the link between the generalization error and the Hessian (see e.g. (Kuznetsov & Mariet, 2019)). Understanding how to make deep learning algorithms work in order to generalize in a non-i.i.d. setting is still a relevant and active topic of research which we aim to address in future work.

References

Achille, A. and Soatto, S. Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, 19(50):1–34, 2018.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Becker, S., Zhang, Y., and Lee, A. A. Geometry of energy landscapes and the optimizability of deep neural networks. *arXiv preprint arXiv:1808.00408*, 2018.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10. IEEE, 2018.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The loss surfaces of multilayer networks. pp. 192–204, 2015.

Geirhos, R., Temme, C. R., Rauber, J., Schütt, H. H., Bethge, M., and Wichmann, F. A. Generalisation in humans and deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 7549–7561, 2018.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 9482–9491, 2018.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Jastrzkebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Kuznetsov, V. and Mariet, Z. Foundations of sequence-to-sequence modeling for time series. *AISTATS)*, 2019.

Li, J., Luo, X., and Qiao, M. On generalization error bounds of noisy gradient methods for non-convex learning. *arXiv preprint arXiv:1902.00621*, 2019.

Li, X., Lu, J., Wang, Z., Haupt, J., and Zhao, T. On tighter generalization bound for deep neural networks: Cnns, resnets, and beyond. *arXiv preprint arXiv:1806.05159*, 2018.

Mandt, S., Hoffman, M. D., and Blei, D. M. Stochastic gradient descent as approximate Bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

Neyshabur, B., Tomioka, R., and Srebro, N. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Reed, R., Oh, S., and Marks, R. Regularization using jittered training data. In *Neural Networks, 1992. IJCNN., International Joint Conference on*, volume 3, pp. 147–152. IEEE, 1992.

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *ICML Workshop Track*, 2018.

Seong, S., Lee, Y., Kee, Y., Han, D., and Kim, J. Towards flatter loss surface via nonmonotonic learning rate scheduling. *UAI*, 2018.

Smith, S. L. and Le, Q. V. A Bayesian perspective on generalization and stochastic gradient descent. 2018.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, G., Patuwo, B. E., and Hu, M. Y. Forecasting with artificial neural networks: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.