Channel Tracking and Prediction with Deep Temporal Convolutional Networks

Arick G. Grootveld, Vlad I. Bugayev, Andrew G. Klein
Department of Engineering and Design, Western Washington University
Bellingham, WA 98225
{grootva, bugayev, andy.klein}@wwu.edu

Kirty P. Vedula, D. Richard Brown III
Department of ECE, Worcester Polytechnic Inst.
Worcester, MA 01609
{kpvedula, drb}@wpi.edu

***Abstract*— This paper proposes a temporal convolutional network (TCN) -based neural network for channel tracking and prediction and compares the tracking performance with the Kalman filter (KF) in two settings: (i) A simulated setting with channel obeys a linear autoregressive (AR) model with additive white Gaussian noise where the KF is known to be optimal. (ii) A simulated setting where the ideal assumptions needed to ensure the optimality of the Kalman filter are not satisfied. Here, we assume a mismatch between the presumed linear time-varying AR model and the true model. We also compare our results in the two scenarios with a least squares approach and the Ricatti equation that provides the asymptotic limits. Numerical results are presented to demonstrate the performance achieved by the machine learning approach and the loss of performance experienced by the KF as a function of the amount of model mismatch. The results demonstrate that the machine learning approach achieves performance close to optimal irrespective of the amount of model mismatch.**

TABLE OF CONTENTS

1. INTRODUCTION

Channel tracking and prediction are critical tasks in communication systems with mobility. The ability to accurately track and predict channels is necessary to enable many aspects of modern communication systems including coherent demodulation, rate and power control, and beamforming [1].

Under an ideal AR channel model with linear dynamics and Gaussian noise and known noise covariance matrices, the classical Kalman filter is the optimal estimator and predictor cite. In non-ideal situations, like for example, when the underlying dynamical model does not match the dynamics of the actual channel, the Kalman filter is no longer optimal.

978-1-7281-2734-7/20/$31.00

We propose to solve this using advances in deep learning, specifically using sequence prediction algorithms to the task of channel tracking and prediction. Recurrent neural networks (RNNs) and long short-term memory (LSTM) have historically been popular choices for sequence prediction and demonstrated promising results for their use in channel tracking [2] [3]. However, RNNs have the problem of exploding/vanishing gradients and have a computationally expensive memory and time requirements. Hence, we propose using temporal convolution networks (TCNs) as they proved to often outperform LSTM-based RNNs in sequence prediction problems [4].

2. CHANNEL MODEL

The Gaussian-Markov channel or the auto-regressive (AR) model is one of the . Due to the inherent Gaussian nature of the noise present in the space and air we are surrounded by as well as the causal state dependency on previous states, it makes sense to assume that the channel can be modeled this way. This is exactly what an AR process is, and previous bodies of work have supplied the motivation and justification for us to choose this model [1] [5].

In general, a noisy observation $z[k]$ of an Nth order autoregressive (AR) process $x[k]$ is described by the equations

$$\begin{aligned} x[k] &= \left(\sum_{n=1}^{N} f_n x[k-n]\right) + v[k] \\ z[k] &= x[k] + w[k] \end{aligned}$$

where $v[k]$ and $w[k]$ are both Gaussian, zero-mean noise with variances σ_v^2 and σ_w^2, assumed to be uncorrelated.

Throughout this paper, we constrain the process order to be AR-2 where the process (or state) $x[k]$ and noisy observation $z[k]$ are given by

$$\begin{aligned} x[k] &= f_1 x[k-1] + f_2 x[k-2] + v[k] \\ z[k] &= x[k] + w[k] \end{aligned} \tag{1}$$

In state-space form, this becomes:

$$\begin{aligned} \boldsymbol{x}[k] &= \boldsymbol{F}\boldsymbol{x}[k-1] + \boldsymbol{v}[k] \\ z[k] &= \boldsymbol{H}\boldsymbol{x}[k] + w[k] \end{aligned} \tag{2}$$

where

$$\begin{aligned} \boldsymbol{x}[k] &= \begin{matrix} x[k] \\ x[k-1] \end{matrix} \\ \boldsymbol{v}[k] &= \begin{matrix} v[k] \\ 0 \end{matrix} \\ \boldsymbol{F} &= \begin{matrix} f_1 & f_2 \\ 1 & 0 \end{matrix} \\ \boldsymbol{H} &= \begin{matrix} 1 & 0 \end{matrix} \end{aligned} \tag{3}$$

The AR process states are complex numbers which contain information about the phase and gain modulation inherent in the channel itself. Under the assumption that the transmission frequency band is very narrow in spectrum, we can model the channel as a singular complex number which alters both the phase and magnitude of signals passed through it. The idea is to use a neural network to extract the nonlinear and weighted relationships between samples such that we can make better predictions and estimations of the state than we could otherwise with linear estimators like the least squares and Kalman filter approaches.

3. Data Generation

Using Python computational packages, we designed a data generation script which will output a variable number of AR-n process sequences of a defined length. This data is used for training and testing the least squares (LS), Kalman filter (KF) and TCN state prediction and estimation systems. Separate training and testing subsets of data are created to ensure validity in the numerical results for both the mismatched and non-mismatched AR parameter cases.

The format of the data that is generated is as follows:

$$AR = \begin{bmatrix} P_{10} & P_{11} & \dots & P_{1n} \\ P_{20} & P_{21} & \dots & P_{2n} \\ \vdots & & \ddots & \\ P_{m0} & P_{m1} & \dots & P_{mn} \end{bmatrix}$$

Where the first index for each element corresponds to the process ID and the second index is the sample number in that specific AR process. Each row may contain a different process generated from different AR coefficients or from the same pair of AR coefficients, depending on if that data is being used to test or train the TCN model. The reasoning for this variation is that during the training phase, we wish to expose the model to as many AR processes as possible such that we obtain a network which is better capable of generalizing and extrapolating to other processes. In testing, we wish to run the network one by one on longer processes, so that we can compare to the KF and LS solutions when they converge. These batches of AR processes are also run through the least squares algorithm as well as the Kalman filter in order to cross examine the performance of all methods with the same datasets.

An important consideration to take when generating an AR-n process are the values of the AR coefficients themselves. If the pair of these values cause the F matrix to have eigenvalues that are greater than 1, the system will become unstable. Figure **??** displays the values which provide stability and instability, as well as the centroid which corresponds to the currently chosen coefficients.

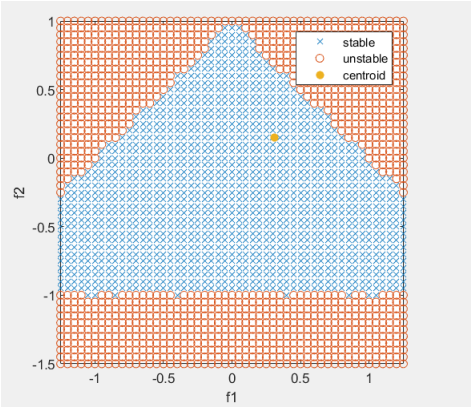


Figure 1. AR Coefficient Stability Region

For the mismatched data generation case where each row of the AR process matrix contains a different AR process generated by a normally distributed deviation from the chosen AR coefficient means, we obtain the heatmap of chosen coefficient values shown in Figure **??**. We discard any values which cause the eigenvalues of the F matrix to be larger than one, therefore the actual mean value of the coefficients deviates from what we use to generate the overall dataset.

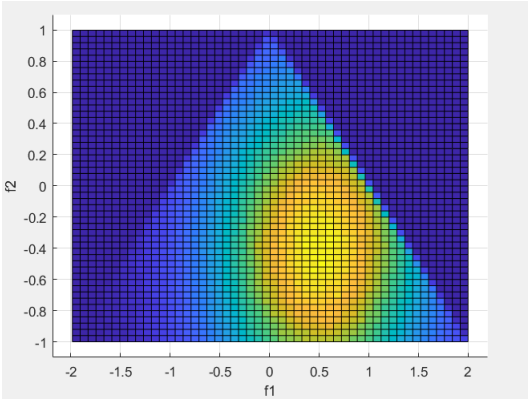


Figure 2. AR Coefficient Heatmap

4. Linear Estimators

The most common approaches to accurately tracking the channel described in Section 2 use linear estimators such as the least squares solution or the Kalman filter. The least squares solution is the most straightforward of the two and simply tries to find a line of best fit to the training data which is then used in further evaluations for test data. The Kalman filter on the other hand requires knowledge of the channel statistics such that a probabilistic estimation and prediction can be made of the current and next states.

Riccati Equation

The Riccati equation allows us to have a lower bound for the best possible performance in channel estimation we could possibly have with a model mismatch, assuming access to an infinite dataset. In this specific case, this equation will always converge to a lower MSE than any other channel estimation

method we are using (LS, KF, TCN). In the matched model case, the Riccati will simply verify that the KF is working properly, as they both converge to and provide the same solution. The Riccati in the discrete time domain is implemented with the following equation:

This matrix has to be in vector form, wherever necessary.
$X = A^{\mathrm{T}}XF - (F^{\mathrm{T}}XH)(R + H^{\mathrm{T}}XH)^{-1}(H^{\mathrm{T}}XF) + Q$

Least Squares

The least squares filter coefficients are obtained which transform the measured state matrix (Z) into the best prediction or estimation ($\hat{x}$) that we can make given a sequence history of N and a filter order of M. The prediction/estimation and least squares coefficient solutions (a_{ls}, b_{ls}) are given by the following equations:

$$\begin{aligned} \hat{x_{est}} &= Z a_{ls} \\ \hat{x_{pred}} &= Z b_{ls} \\ a_{ls} &= (Z^{\mathrm{T}}Z)^{-1}Z^{\mathrm{T}}x_{est} \\ b_{ls} &= (Z^{\mathrm{T}}Z)^{-1}Z^{\mathrm{T}}x_{pred} \end{aligned} \tag{4}$$

where

$$\begin{aligned} x &= [x[k] \dots x[k-M]]^{\top} \\ \hat{x_{pred}} &= [\hat{x}[k-1] \dots \hat{x}[k-M-1]]^{\top} \\ \hat{x_{est}} &= [\hat{x}[k] \quad \hat{x}[k-1] \quad \dots \quad \hat{x}[k-M]]^{\top} \\ a_{ls} &= [a_0 \quad a_1 \quad \dots \quad a_N]^{\top} \\ b_{ls} &= [b_0 \quad b_1 \quad \dots \quad b_N]^{\top} \end{aligned} \tag{5}$$

$$\boldsymbol{Z} = \begin{bmatrix} z[k-1] \dots z[k-N-1] & & \\ z[k-2] \dots z[k-N-2] & & \\ \vdots & & \\ & & \ddots \\ z[k-M-1] \dots z[k-M-N-1] & & \end{bmatrix} \tag{6}$$

The least squares filter coefficients are computed once using a training data sequence consisting of many segments of either one AR process or many, depending on the model being used. Both the mismatched and matched model cases are evaluated, with training occurring on large collections of AR processes and testing on individual, longer sequences.

Kalman Filter

Given knowledge of the channel parameters and statistics via data collection, we can implement the Kalman filter approach to channel tracking. We wish to estimate the current state as well as predict the future state of the AR process. To realize the Kalman filter in this situation, we used the following state update equations:

$$\begin{aligned} \hat{x}[n|n-1] &= \mathbf{F}\hat{x}[n-1|n-1] \\ \mathbf{M}[n|n-1] &= \mathbf{F}\mathbf{M}[n-1|n-1]\mathbf{F}^{\mathrm{T}} + \mathbf{Q} \\ \mathbf{K}[n] &= \frac{\mathbf{M}[n|n-1]\mathbf{H}}{\sigma_w^2 + \mathbf{H}^{\mathrm{T}}\mathbf{M}[n|n-1]\mathbf{H}} \\ \hat{x}[n|n] &= \hat{x}[n|n-1] + \mathbf{K}[n](z[n] - \mathbf{H}^{\mathrm{T}}\hat{x}[n|n-1]) \\ \mathbf{M}[n|n] &= (\mathbf{I} - \mathbf{K}[n]\mathbf{H}^{\mathrm{T}})\mathbf{M}[n|n-1] \end{aligned}$$

Notation	Description
$\hat{x}[n\|n-1]$	State prediction
$\hat{x}[n-1\|n-1]$	Correction value
$\mathbf{M}[n\|n-1]$	Minimum prediction MSE
$\mathbf{M}[n-1\|n-1]$	Minimum estimation MSE
Q	Process noise
$\mathbf{K}[n]$	Kalman gain
σ_w^2	Measurement noise covariance

Generally, if the Kalman filter has perfect knowledge of the real AR coefficients, the MMSE of this system will approach the Riccati equation MMSE with longer datasets corresponding to closer matching between the two values. In our case, we tested both circumstances; when the Kalman filter has perfect knowledge of the coefficients and also when there is a slight deviation.

5. Temporal Convolutional Network

The temporal convolutional network (TCN) is the neural network architecture we chose to design and test against the least squares and Kalman filter solutions. The TCN has proven to be more successful in sequence identification problems than recurrent neural networks (RNN) and long-short term memory neural networks (LSTM). The structure and general design of this neural network was previously created by another research team [4] which we used and altered for channel estimation specific inputs and outputs.

The TCN structure is essentially a one dimensional, fully connected network with causal convolutions. The main differences between a traditional convolutional neural network and the TCN are the diluted causal convolutions between network layers and the ability to specify a desired output sequence length. To accomplish this, the network contains hidden layers which are all of the same length and uses zero padding to keep the dilutions feasible and maintain the output size specification. Convolution dilution happens in this network to increase the receptive field of each convolutional layer, providing the network the ability to use sequence elements that are further back in the input history.

The TCN consists of multiple "residual blocks" which act as fundamental construction elements in this architecture. One instance of this network may consist of many residual blocks, all daisy-chaining in one long sequence. These residual blocks consist of two layers of diluted, causal convolutions and a non-linearity (ReLU) on the output of each layer. The residual blocks contain an identity mapping which aids in training for slight variations around the input sequence by explicitly weighting the output with a scaled version of the input before any convolutions took place.

Due to the complex nature of the channel values and lack of PyTorch support for complex valued neural networks, the imaginary and real components had to be split into separate floating point numbers in order to effectively train the network. This splitting into real and imaginary components has already been done by previous research teams with favorable results [6]. The loss function for this network is the mean-

squared error (MSE), and for complex values this is calculated in the following way:

$$MSE \quad = \quad E(|x - \hat{x}|^2)$$

Where x and $\hat{x}$ are both complex valued states (actual and computed states, respectively).

6. RESULTS

Static AR Processes

	MMSE	Least Squares	Kalman Filter	TCN
Estimation	ay	doe	ay	doe
Prediction	aye	doe	ayez	doez

Varied AR Processes

	MMSE	Least Squares	Kalman Filter	TCN
Estimation	ay	doe	ay	doe
Prediction	aye	doe	ayez	doez

7. CONCLUSION

This is the conclusion.

All source code for this experiment is available at — `https://github.com/aspectlab/deepchannel`

ACKNOWLEDGMENTS

This work was supported by a Research Experience for Undergraduates (REU) Supplement to National Science Foundation award CNS-1836695.

REFERENCES

[1] J. Kang, C. Chun, I. Kim, and D. I. Kim, "Channel tracking for wireless energy transfer: A deep recurrent neural network approach," *arXiv:1812.02986*, 2018.

[2] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

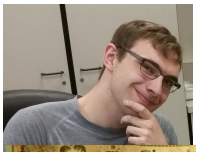
[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

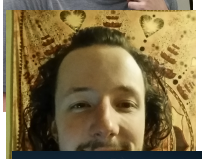
[5] K. E. Baddour and N. C. Beaulieu, "Autoregressive models for fading channel simulation," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270)*, vol. 2. IEEE, 2001, pp. 1187–1192.

[6] N. Taşpinar and M. N. Seyman, "Back propagation neural network approach for channel estimation in ofdm system," pp. 265–268, 2010.

BIOGRAPHY



Arick Grootveld *is currently pursuing a B.S. in electrical engineering from WWU and expects to graduate in June 2020. His current research activities include...*



Vlad Bugayev *is currently pursuing a B.S. in electrical engineering from WWU and expects to graduate in June 2020. His current research activities include ...*

Andrew G. Klein *received the B.S. degree from Cornell University, Ithaca, NY, USA, the M.S. degree from the University of California, Berkeley, CA, USA, and the Ph.D. degree from Cornell University, all in electrical engineering. Previously, he was an Assistant Professor with the Worcester Polytechnic Institute, Worcester, MA, USA, from 2007 to 2014, and he was a Post-Doctoral Researcher with Supélec/LSS, Paris, France, from 2006 to 2007. He joined the Department of Engineering and Design, Western Washington University, Bellingham, WA, USA, in 2014, where he is currently a Professor.*

D. Richard Brown III *received the B.S. and M.S. degrees from the University of Connecticut in 1992 and 1996, respectively, and the Ph.D. degree from Cornell University in 2000, all in electrical engineering. From 1992 to 1997, he was with General Electric Electrical Distribution and Control. He was a Faculty Member with the Worcester Polytechnic Institute, Worcester, MA, USA, in 2000. He was a Visiting Associate Professor with Princeton University from 2007 to 2008. From 2016 through 2018, he was with the Computing and Communication Foundations Division, National Science Foundation, as a Program Director.*

Kirty Vedula *received the M.S. degree in Electrical and Computer Engineering from Rutgers University in* 2014. *He joined the PhD in Electrical and Computer Engineering program in* 2015. *He also worked as an Research and Development Intern at Philips, Witricity and Mathworks. His research interests are in signal processing, wireless communication systems, machine learning and deep learning.*