Time Series Analysis – Lecture 9

Recurrent and Temporal Convolutional Networks

Fredrik Lindsten, Linköping University

2019-10-14

Aim and outline

Aim:

- Introduce two popular deep-learning-based methods for time series analysis
- Highlight some formal connections with classical models (state space and auto-regressive) that you have seen in the course.

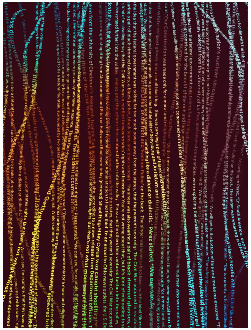
Outline:

1. Basics of neural network models — the multi-layer perceptron
2. Linear Gaussian state space models on innovation form
3. A nonlinear generalization — Recurrent Neural Networks
4. Nonlinear auto-regressive models
5. Temporal Convolutional Networks

ex) Generating text

Input (human-written)	In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Model completion (machine-written)	`The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.`
	`Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.`
	`Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.`
	`Pérez and the others then ventured further into the valley. ''By the time we reached the top of one peak, the water looked blue, with some crystals on top,'' said Pérez.`



`https://openai.com/blog/better-language-models/`

State Space Models $\Longrightarrow$ Recurrent Neural Networks

Linear state space models

Linear state space model:

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t,$$
$$x_t = C\mathbf{z}_t + \nu_t.$$

Limitation:

The next state $\mathbf{z}_{t+1}$ as well as the observation x_t depend **linearly** on the current state $\mathbf{z}_t$.

The model flexibility is limited.

Going nonlinear

Aim: Increase the flexibility of the model by replacing the linear functions by **generic** and **flexible** nonlinear functions.

Linear function: $y = A\mathbf{z}$, where the matrix A is the **parameter**.

Nonlinear function: $y = f_\theta(\mathbf{z})$. Here, θ is a vector of **parameters** determining the shape of the function $f_\theta(\cdot)$.

ex) Let $\theta = (\theta_1, \theta_2, \theta_3)$, and

$$f_\theta(z) = \frac{\theta_1}{\theta_2 + z^{\theta_3}}.$$

Neural networks

Recall: We want to use **generic** and **flexible** nonlinear functions.

This is precisely what **neural networks** provide!

Fully connected, 1-layer network:

We **construct** a function $f_\theta : \mathbb{R}^p \mapsto \mathbb{R}$ by

$$\begin{aligned} \mathbf{h} &= \sigma(W^{(1)}\mathbf{z} + b^{(1)}) \\ y &= W^{(2)}\mathbf{h} + b^{(2)}. \end{aligned}$$

That is,

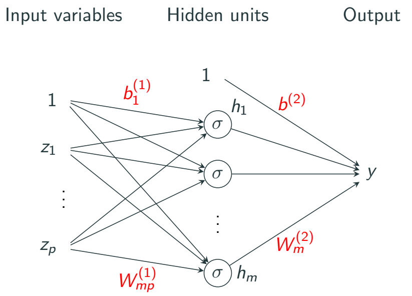
$$f_\theta(\mathbf{z}) = W^{(2)}\sigma(W^{(1)}\mathbf{z} + b^{(1)}) + b^{(2)}$$

Neural network – graphical illustration

The equations

$$\mathbf{h} = \sigma(W^{(1)}\mathbf{z} + b^{(1)})$$
$$y = W^{(2)}\mathbf{h} + b^{(2)}.$$

can be illustrated graphically.

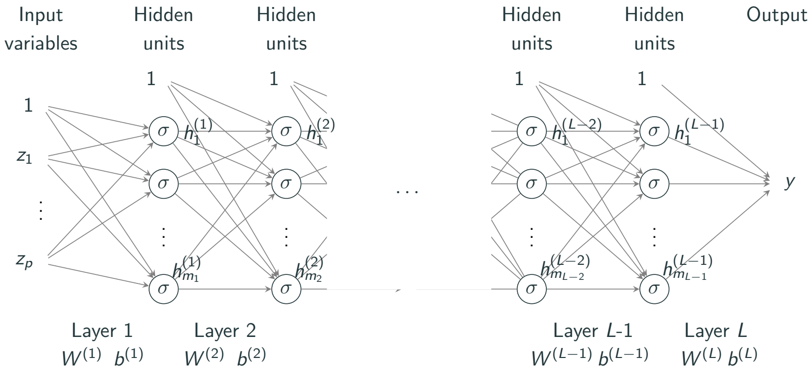


- The variables $\mathbf{h} = (h_1, \ldots, h_m)$ are referred to as a **hidden layer**.
- The function $\sigma(\cdot)$ is an element-wise nonlinearity, referred to as an **activation function**. Typical choices are

$$\sigma(x) = \tanh(x) \qquad \text{or} \qquad \sigma(x) = \text{ReLU}(x) = x\mathbb{1}(x \geq 0)$$

- The model **parameters** are the weight matrices and bias vectors $\theta = \{W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$.

Multi-layer perceptron



Innovation form

Linear state space model:

$$\begin{aligned} \mathbf{z}_t &= A\mathbf{z}_{t-1} + e_t, \\ x_t &= C\mathbf{z}_t + \nu_t. \end{aligned}$$

Innovation form. There exists an **equivalent** representation given by

$$\begin{aligned} \mathbf{h}_t &= W\mathbf{h}_{t-1} + Ux_{t-1}, \\ x_t &= C\mathbf{h}_t + \nu'_t. \end{aligned}$$

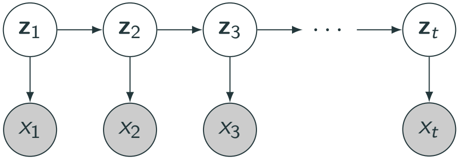
(Assuming stationarity for simplicity.)

Proof. Let $\mathbf{h}_t = m_{t|t-1}$, the Kalman predictive mean.

Innovation form

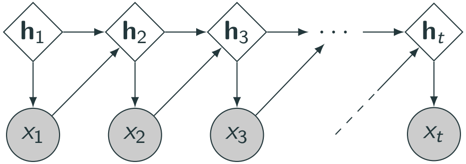
Original form:

$$\mathbf{z}_t = A\mathbf{z}_{t-1} + e_t,$$
$$x_t = C\mathbf{z}_t + \nu_t.$$



Innovation form:

$$\mathbf{h}_t = W\mathbf{h}_{t-1} + Ux_{t-1},$$
$$x_t = C\mathbf{h}_t + \nu'_t.$$



The hidden state variables can be **deterministically and recursively computed** from the data.

Going nonlinear

Doesn't this look suspiciously similar to an MLP. . . ?

$$\begin{aligned} \mathbf{h}_t &= W\mathbf{h}_{t-1} + Ux_{t-1}, \\ x_t &= C\mathbf{h}_t + \nu'_t, \end{aligned}$$

for some **nonlinear activation function** $\sigma(\cdot)$.

This is a basic **Recurrent Neural Network (RNN).**

Learning the parameters

The model parameters are the weight matrices and bias vectors:

$$\begin{aligned} \mathbf{h}_t &= \sigma(W\mathbf{h}_{t-1} + Ux_{t-1} + b), \\ x_t &= C\mathbf{h}_t + c + \nu'_t, \end{aligned}$$

with $\theta = \{W, U, b, C, c\}$.

Note:

- The parameters are the same for all time steps ("weight sharing").
- The fact that there is no state noise means that we can compute

$$p_\theta(x_t \mid x_{1:t-1}) = N(x_t \mid C\mathbf{h}_t + c, \sigma^2_{\nu'}).$$

Learning the parameters

We can thus learn the parameters θ directly by optimizing the negative log-likelihood,

$$L(\theta; x_{1:T}) = -\sum_{t=1}^{T} \log p_\theta(x_t \mid x_{1:t-1}),$$

using gradient-based optimization.

The gradient $\nabla_\theta L(\theta; x_{1:T})$ is computed using the chain rule of differentiation, propagating information from $t = 1$ to $t = T$ and then back again.

$\Longrightarrow$ **Back-propagation through time.**

RNN extensions

- GRU/LSTM
- Non-Gaussian likelihood (e.g., for discrete data)
- Conditioning on context (input)
- Stochastic hidden layers
- Bidirectional connections
- . . .

Autoregressive Models $\Longrightarrow$ Temporal Convolutional Nets

Autoregressive models

State space models and RNNs use a latent state vector to model temporal dependencies.

An alternative is to model the dependency of the current data point x_t on the past data points $x_{1:t-1}$ by a **direct functional relationship**.

Auto-regressive model, AR(p)**:**

$$x_t = \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + w_t, \qquad w_t \sim N(0, \sigma_w^2).$$

The AR model is linear in the parameters:

- ▲ Learning of parameters easy $\Leftrightarrow$ linear regression
- ▼ Flexibility/ability to model complex temporal dependencies is limited
- ▼ Memory/receptive field is just p time steps

Going nonlinear

Nonlinear auto-regressive model, NAR(p):

$$x_t = \sigma(\phi_1 x_{t-1} + \cdots + \phi_p x_{t-p}) + w_t, \qquad w_t \sim N(0, \sigma_w^2),$$

for some nonlinear activation function σ.

- ▲ Flexibility increased...
- ▼ ...but only slightly!
- ▼ Memory/receptive field is *still* just p steps

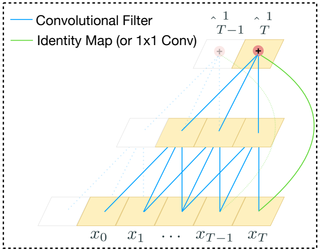
We can address these issues with a **multi-layer network architecture!**

Temporal Convolutional Network

2-layer TCN:

$$h_{t-1} = \sigma(\phi_1^{(1)} x_{t-1} + \cdots + \phi_p^{(1)} x_{t-p}),$$
$$x_t = \sigma(\phi_1^{(2)} h_{t-1} + \cdots + \phi_p^{(2)} h_{t-p}) + w_t,$$

with $w_t \sim N(0, \sigma_w^2)$.

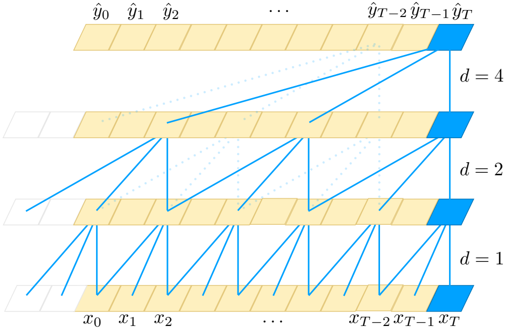


Can extend to multiple hidden layers, $h_t^{(1)}$, $h_t^{(2)}$, . . .

- ▲ Multiple layers $\Rightarrow$ very flexible models
- ▲ Receptive field increases with depth. . .
- ▼ . . . but only linearly

TCN with dilated convolutions

By using **dilated convolutions** we can increase receptive field **exponentially** with depth.



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

RNN vs TCN

RNNs are still the *de facto* standard deep learning approach to time series modeling, *but...*

...TCNs have outperformed them on many benchmark problems

Shaojie Bai, J. Zico Kolter, Vladlen Koltun. **An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.** *arXiv.org: 1803.01271*, 2018.

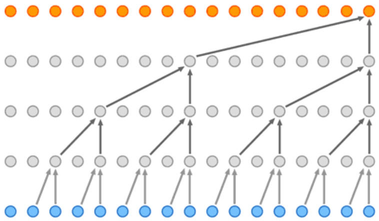
...and have other advantages too:

- Easier parallelization
- Better control over receptive field
- Lower memory requirements during training
- ...

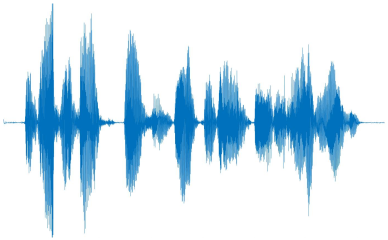
...but also some disadvantages:

- More difficult to reuse model for multiple tasks
- Larger memory requirements after deployment
- ...

ex) WaveNet



WaveNet by DeepMind powers Google's text-to-speech technology.



Deep Learning for TSA

So is this what we should always do for modeling sequential data?!

No!

- Only makes sense to use something as complex as RNN or TCN when classical methods fail **— Try Simple Things First!**
- Methods based on deep learning work best if we have multiple sequences, or one long sequence that can be split into segments
- For a single univariate time series, classical methods (ARIMA, state space, . . .) often work better.