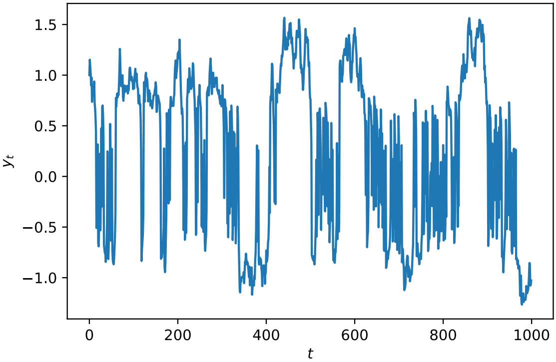
Time Series and Sequence Learning

Nonlinear auto-regressive models

Fredrik Lindsten, Linköping University

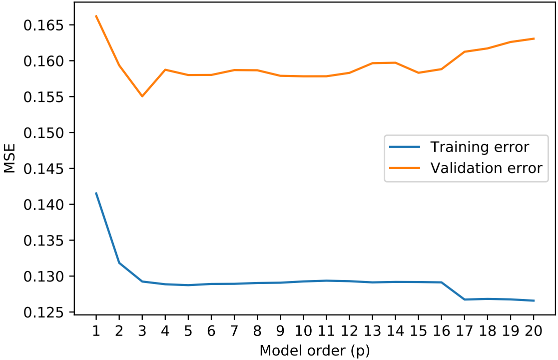
ex) Toy data



Can we model this using an AR model?

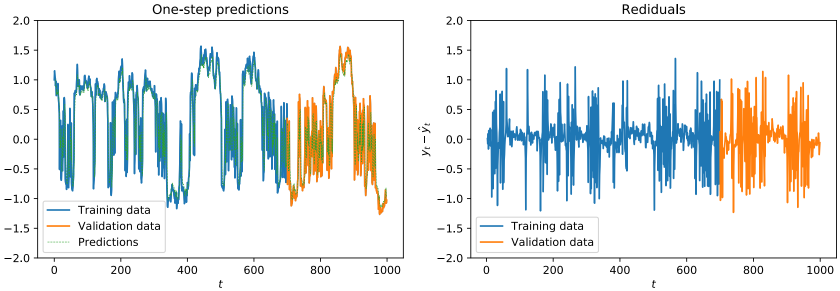
ex) Toy data, cont'd

We fit AR(p) models of different order to the first 700 time points, and validate using the remaining 300.



The model of order $p = 3$ gives the lowest validation error.

ex) Toy data, cont'd



- Residuals are not iid
- Residuals occasionally take large values (same order as data)

Validation MSE $= 0.155$

Limitations of AR models

Auto-regressive model, AR(p):

$$y_t = a_1 y_{t-1} + \cdots + a_p y_{t-p} + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2).$$

The AR model is linear in the parameters:

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

Going nonlinear

Nonlinear auto-regressive model, NAR(p):

$$y_t = f_\theta(y_{t-1}, y_{t-2}, \ldots, y_{t-p}) + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

for some nonlinear function f_θ.

Here, θ is a vector of parameters determining the shape of the function f_θ.

ex) SETAR is a regime switching AR model, e.g.,

$$y_t = \begin{cases} a_{11} y_{t-1} + \cdots + a_{1p} y_{t-p} + \varepsilon_t, & \text{if } y_{t-d} \leq c \\ a_{21} y_{t-1} + \cdots + a_{2q} y_{t-q} + \varepsilon_t, & \text{if } y_{t-d} > c \end{cases}$$

Designing the nonlinearity

How do we select the function f_θ?

1. By "physical" insights
2. To enable certain sought-after properties (e.g., regime switching)
3. ...

Our approach: Increase the flexibility of the model by replacing the linear functions by generic and flexible nonlinear functions.

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

Neural networks

Neural network: A way of specifying a flexible family of nonlinear functions $y = f_\theta(\mathbf{z})$. Here, θ is a set of parameters determining the shape of the function f_θ.

Fully connected, 2-layer network:
We construct a function $f_\theta : \mathbb{R}^p \mapsto \mathbb{R}$ by

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

That is,

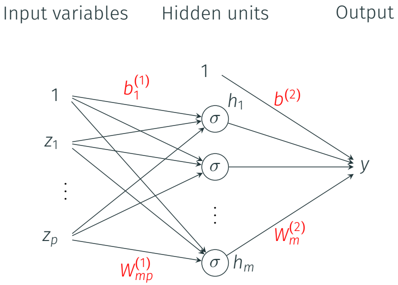
$$f_\theta(\mathbf{z}) = W^{(2)}\sigma(W^{(1)}\mathbf{z} + b^{(1)}) + b^{(2)}$$

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

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

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

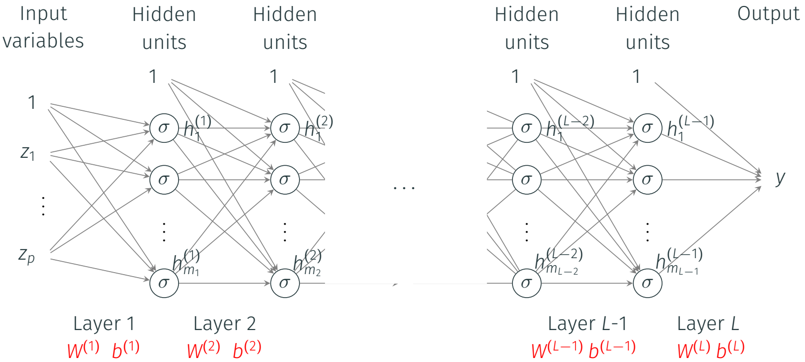
Multi-layer perceptron

More complicated nonlinear functions can be modeled by adding additional hidden layers.

Multi-layer perceptron (MLP):
We construct a function $f_\theta : \mathbb{R}^p \mapsto \mathbb{R}$ by

$$\begin{aligned} \mathrm{h}^{(1)} &= \sigma(W^{(1)}\mathrm{z} + b^{(1)}), \\ \mathrm{h}^{(j)} &= \sigma(W^{(j)}\mathrm{h}^{(j-1)} + b^{(j)}), \qquad\qquad j = 2, \ldots, L-1, \\ y &= W^{(L)}\mathrm{h}^{(L-1)} + b^{(L)}. \end{aligned}$$

Multi-layer perceptron



Nonlinear AR, revisted

Nonlinear auto-regressive model, NAR(p):

$$y_t = f_{\theta}(y_{t-1}, y_{t-2}, \ldots, y_{t-p}) + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma_\varepsilon^2),$$

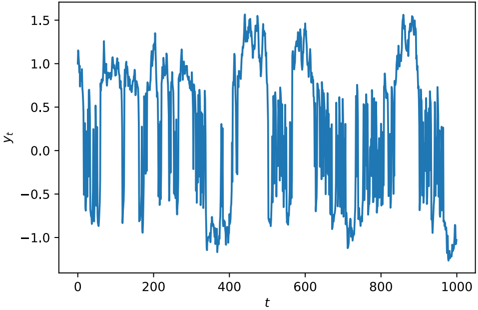
We can model f_θ with a MLP.

Training: Minimize, e.g., sum of squared prediction errors by nonlinear optimization

$$L(\theta) = \frac{1}{n-p} \sum_{t=p+1}^{n} \left(y_t - f_\theta(y_{t-1}, y_{t-2}, \ldots, y_{t-p})\right)^2$$

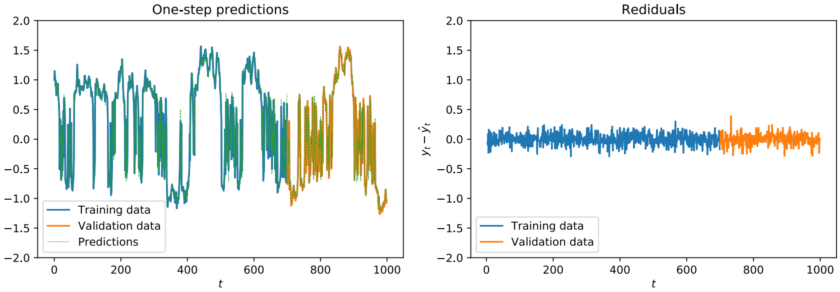
ex) Toy data, revisited

Recall the toy data from before,



We fit a NAR model $y_t = f_{\theta}(y_{t-1}, y_{t-2}, y_{t-3}) + \varepsilon_t$ where f_{θ} is a 2-layer network with $m = 10$ hidden units and ReLU activation functions.

ex) Toy data, revisited



Validation MSE $= 0.0095$.

Summary

Nonlinear AR = nonlinear regression, with lagged values as regressors.

- ▲ Flexible dependencies on past values
- ▼ Learning requires nonlinear optimization
- ▼ More hyper-parameters to tune (order p, number of hidden layers L, number of hidden neurons m, ...)
  - Grid search possible, but can be expensive
  - Some guidance on order selection from first fitting a linear AR model

Try simple things first!