Recurrent Neural Networks (RNN)

Consider time series data (x_t, y_t)

So far, we have written the time series prediction problem as a nonlinear function (neural net) applied to a fixed memory

$$y_t \approx f(x_{t-1}, x_{t-2}, \ldots, x_{t-T})$$

Today we will focus on methods that exploit the sequential nature of time series to allow for longer memory

Background: Dynamical Systems

$s_t = f(s_{t-1}; \beta)$ where s_t is the "state of the system" at time t

Consider the recurrence: $s_3 = f(s_2; \beta) = f(f(s_1; \beta); \beta)$

example: Linear Autoregressive Models

$$AR(1): \quad s_t = \beta_0 + \beta_1 s_{t-1} + \varepsilon_t \quad \leftarrow \text{Noise/innovation}$$

$$= \beta_0 + \beta_1 \beta_0 + \beta_1^2 s_{t-2} + \varepsilon_t + \beta_1 \varepsilon_{t-1}$$

$$= \dots$$

$$= \beta_0 (1 + \beta_1 + \beta_1^2 + \dots) + \underbrace{(\varepsilon_t + \beta_1 \varepsilon_{t-1} + \beta_1^2 \varepsilon_{t-2} + \dots)}_{\text{recursion can provide memory}}$$

Idea: RNN is a nonlinear dynamical system driven by an external input / innovation x_t

$$\hookrightarrow \quad s_t = f(s_{t-1}, x_t, \beta)$$

$$= f(f(s_{t-2}, x_{t-1}, \beta), x_t, \beta)$$

$$= \dots$$

$$= g(x_t, x_{t-1}, \dots, x_1, \beta) \quad \text{for some function } g$$

Remembers the entire past