Linear Sequence Processing

Deep Learning

Objective

- Processing sequences is very different than processing static examples.
  - Adding temporal dimensions creates several cognitive challenges.
  - Static problems can be visualized at once – dynamic problems require ”playing through” sequences.
  - Feedback loops create circular reasoning.
  - Each time step adds dimensions to the problem space.
  - Understanding how state encodes history is conceptually challenging.
- We start with simple linear systems in this chapter.
- We will build up from basic elements.
- The concepts from linear sequence processing systems that are covered here will provide a foundation for understanding deep sequence processing in the following chapters.

What is a sequence

- An ordered collection of elements, where both their values and their order matter.
- Can appear in a variety of application areas.
- Non-numerical sequences will be converted to numbers before processing by neural networks.

Application	**Example Sequence**
DNA analysis (nucleotides)	{A, T, C, G, G, T, A, C, C, C, T, T}
Protein folding (amino acids)	{Met, Gly, Trp, Ser, Cys, Ile, Ala, Leu, Phe, Leu}
E-commerce (actions)	{homepage, product search, category browse, product view, add to cart, view cart, start checkout, payment, confirmation}
Stock prediction (prices)	{150.25, 151.50, 153.15, 155,75, 154.30}
Health monitoring (heart rate)	{72, 73, 75, 74, 75, 73, 73, 76, 74}
Translation (words/subwords)	{’A’, ’sequence’, ’is’, ’an’, ’ordered’, ’set’, ’of’, ’elements’}

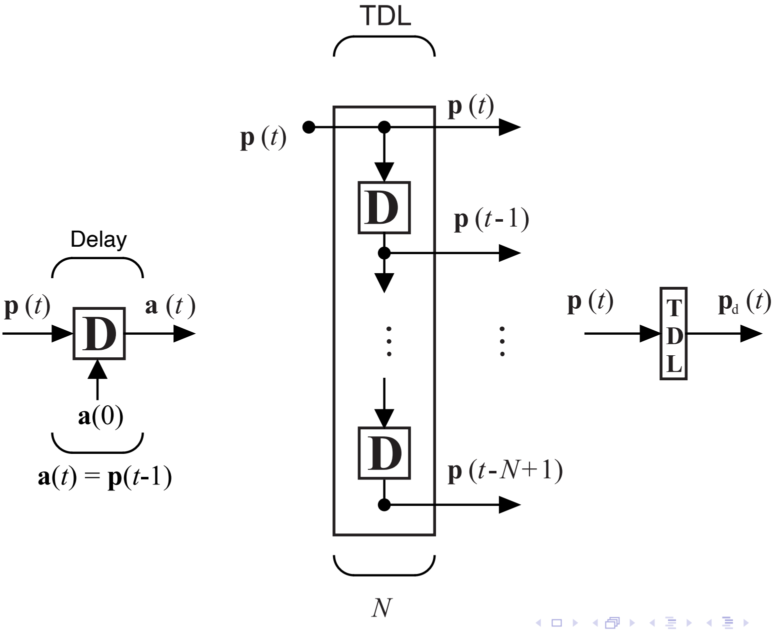
Dynamic systems

- To process sequences, we need systems with memory – dynamic systems.
- Without memory, a system would merely be a static function mapping inputs to outputs.
- The output would be determined solely by current inputs.
- Some dynamic systems have finite memory, while others have infinite memory.
- Infinite memory requires feedback (recurrent connections).
- In this chapter we consider linear dynamic systems.
- We begin with finite memory systems (finite impulse response).
- Then we extend to infinite memory (infinite impulse response).

Names for dynamic sequence processing systems

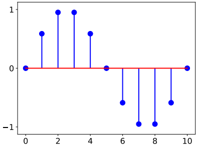
- Digital filters
- Time series models
- Difference equations
- State machines
- Sequence-to-sequence models
- Sequential memory networks
- Stream processors
- Signal processors
- Convolvers
- Sequence encoders/decoders

Memory units - delays and tapped delay lines

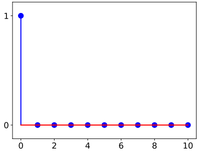


The difference between static and dynamic problems

- Consider the values represented by the blue dots in this figure.

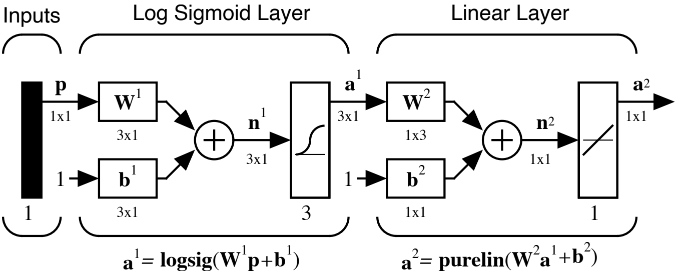


- They could be targets for a static network whose example inputs ranged from 0 to 10. We approximate $y = f(p) = \sin(2\pi p/10)$.
- They could be a time sequence, produced by passing the sequence below through a dynamic system. We approximate $y(t) = 1.62y(t-1) - y(t-2) + 0.6p(t-1)$.



Neural network solution to the static problem

Approximating a Static Function

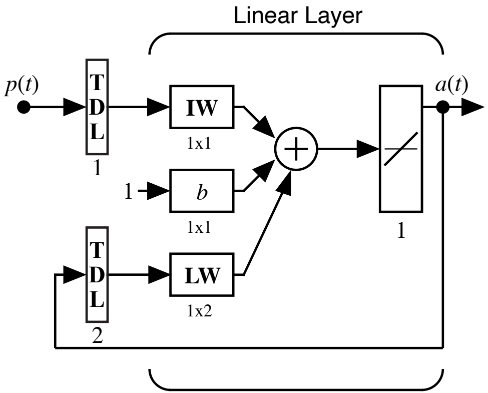


$$\mathbf{W}^1 = \begin{bmatrix} -0.6246 \\ -0.5410 \\ -0.6246 \end{bmatrix}, \mathbf{b}^1 = \begin{bmatrix} 0.3239 \\ 2.7051 \\ 5.9225 \end{bmatrix}$$

$$\mathbf{W}^2 = \begin{bmatrix} -5.6565 & 7.4708 & -5.6565 \end{bmatrix}, \mathbf{b}^1 = \begin{bmatrix} 1.9211 \end{bmatrix}$$

Neural network solution to the dynamic problem

Approximating a Dynamic System



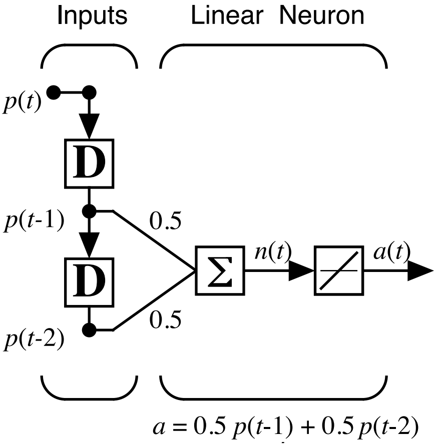
$$\mathbf{IW} = \left[0.6\right], b = \left[0\right], \mathbf{LW} = \left[1.62 \quad -1\right]$$

Summary differences between static and dynamic problems

- Static problems:
  - Example inputs are single scalars, vectors, arrays or tensors.
  - The network has no memory.
  - Each example input is considered in isolation.
  - Static problems use algebraic equations.
  - Static networks don't have feedback
- Dynamic sequence processing problems:
  - Example inputs are sequences of scalars, vectors, arrays or tensors.
  - The network has memory.
  - Each time step adds dimensions to the problem space.
  - The network must use relationships between the elements of each sequence.
  - Effects can be separated from causes by many time steps.
  - Dynamic problems require difference equations.
  - Dynamic networks may use feedback.
  - Feedback creates stability issues.

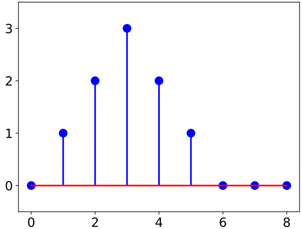
Sequence averaging network

Tapped Delay Line (TDL) into single neuron.

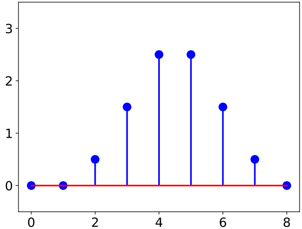


Network response

t	0	1	2	3	4	5	6	7	8
$p(t)$	0	1	2	3	2	1	0	0	0
$a(t)$	0	0	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{5}{2}$	$\frac{3}{2}$	$\frac{1}{2}$	0
TDL	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 3 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 2 \\ 3 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 2 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$



Input Sequence $p(t)$



Output Sequence $a(t)$

Python averaging network class (initialization)

```
class averaging_network:
    def __init__(self, iw, p_tdl=[]):
        self.iw = np.array(iw)

        if len(p_tdl) > 0:
            self.p_tdl = np.array(p_tdl)
        else:
            if len(self.iw) > 0:
                self.p_tdl = np.zeros(len(self.iw) - 1)
            else:
                self.p_tdl = np.zeros(0)
```

Python averaging network class (step and process)

```
class averaging_network:
    def step(self, p):
        a = self.iw[0] * p

        for i in range(len(self.p_tdl)):
            a += self.iw[i + 1] * self.p_tdl[i]

        if len(self.p_tdl) > 0:
            self.p_tdl = np.roll(self.p_tdl, 1)
            self.p_tdl[0] = p

        return a

    def process(self, input_sequence):

        output = np.zeros(len(input_sequence))
        for i in range(len(input_sequence)):
            output[i] = self.step(input_sequence[i])
        return output
```

Example run

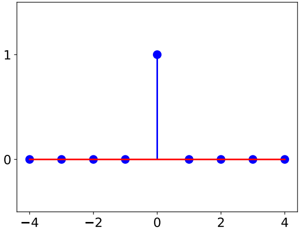
```python
# Input weights
iw = [0, 0.5, 0.5]

# Input sequence
p = [0, 1, 2, 3, 2, 1, 0, 0, 0]

# Define the network
net = averaging_network(iw)

# Run the input through the network
a = net.process(p)

print('Input␣sequence:␣')
print(p)
print('Output␣sequence:␣')
print(a)
```

```
Input sequence:
[0, 1, 2, 3, 2, 1, 0, 0, 0]
Output sequence:
[0.  0.  0.5 1.5 2.5 2.5 1.5 0.5 0. ]
```

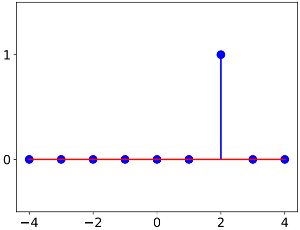
Impulse function (Kronecker delta)

$$\delta(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases}$$

t	-4	-3	-2	-1	0	1	2	3	4
$\delta(t)$	0	0	0	0	1	0	0	0	0
$\delta(t-2)$	0	0	0	0	0	0	1	0	0



Impulse $\delta(t)$



Shifted Impulse $\delta(t-2)$

Representing sequences as impulses

$$p(t) = 1\delta(t-1) + 2\delta(t-2) + 3\delta(t-3) + 2\delta(t-4) + 1\delta(t-5)$$

t	0	1	2	3	4	5	6	7	8
$1\delta(t-1)$	0	1	0	0	0	0	0	0	0
$2\delta(t-2)$	0	0	2	0	0	0	0	0	0
$3\delta(t-3)$	0	0	0	3	0	0	0	0	0
$2\delta(t-4)$	0	0	0	0	2	0	0	0	0
$1\delta(t-5)$	0	0	0	0	0	1	0	0	0
$p(t)$	0	1	2	3	4	1	0	0	0

$$p(t) = \sum_i p(i)\delta(t-i)$$

Finite impulse response system

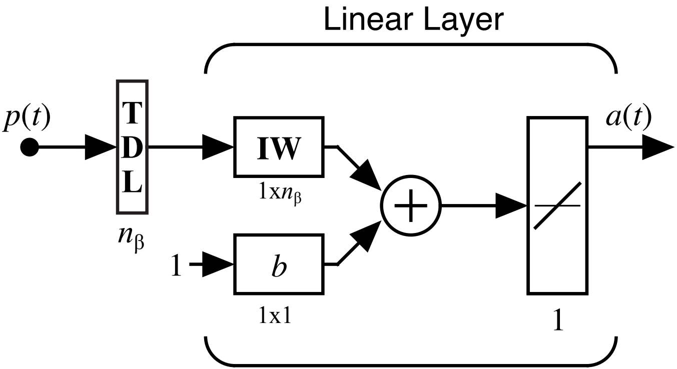
General FIR system

$$a(t) = \sum_{i=1}^{n_\beta} \beta_i p(t-i)$$

Difference Equation

$$a(t) = \beta_1 p(t-1) + \beta_2 p(t-2) + \ldots + \beta_{n_\beta} p(t-n_\beta)$$

FIR network



$$\mathbf{IW} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_{n_\beta} \end{bmatrix}, b = 0$$

System impulse response

FIR System Response to Impulse Input

$$\begin{aligned}
h(t) &= \sum_{i=1}^{n_\beta} \beta_i \delta(t-i) \\
h(1) &= \sum_{i=1}^{n_\beta} \beta_i \delta(1-i) = \beta_1 \\
h(2) &= \sum_{i=1}^{n_\beta} \beta_i \delta(2-i) = \beta_2 \\
&\vdots \\
h(n_\beta) &= \beta_{n_\beta}
\end{aligned}$$

Impulse response table

$$h(t) = \sum_{i=1}^{n_\beta} \beta_i \delta(t-i) = \begin{cases} \beta_t & t = 1, \cdots, n_\beta \\ 0 & \text{otherwise} \end{cases}$$

t	0	1	2	3	4	$\cdots$	n_β	$n_\beta + 1$	$t > n_\beta$
$p(t) = \delta(t)$	1	0	0	0	0	0	0	0	0
$a(t) = h(t)$	0	β_1	β_2	β_3	β_4	$\cdots$	β_{n_β}	0	0

Impulse response and convolution

- Since the FIR coefficients are identical to the impulse response values:

$$a(t) = \sum_{i=1}^{n_\beta} h(i)p(t-i)$$

- This is called a convolution sum.

$$a(t) = h(t) * p(t)$$

- The length of the impulse response is a measure of the memory of the dynamic system.
- In this case, the system looks at the last n_β values of the input.

Derivation of general convolution sum

- We have a sequence $p(t)$, represented as impulses.

$$p(t) = \sum_i p(i)\delta(t-i)$$

- Consider a linear time invariant system $\mathcal{W}$ operating on the sequence.

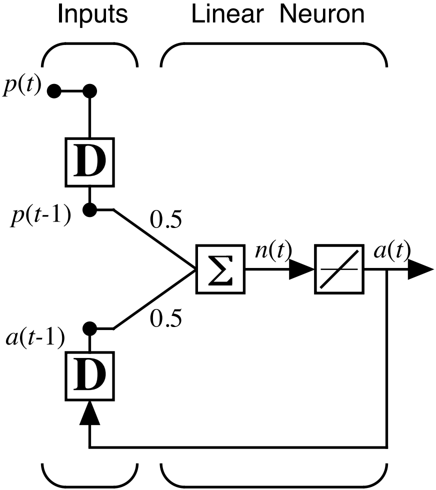
$$\mathcal{W}(p(t)) = \mathcal{W}\left(\sum_i p(i)\delta(t-i)\right)$$

- Because of linearity and time invariance, the operation can be brought inside the summation, and the operation will be consistent at each time step. The response is a convolution of the impulse response and the input.

$$\mathcal{W}(p(t)) = \sum_i p(i)\mathcal{W}(\delta(t-i)) = \sum_i p(i)h(t-i) = h(t) * p(t)$$

Sequence smoothing network

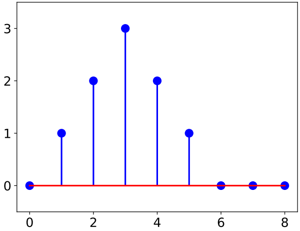
Feedback around a single neuron.



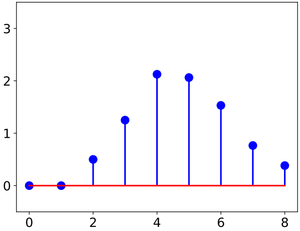
$$a = 0.5\, a(t\text{-}1) + 0.5\, p(t\text{-}1)$$

Network response

t	0	1	2	3	4	5	6	7	8
$p(t)$	0	1	2	3	2	1	0	0	0
$a(t)$	0	0	0.5	1.25	2.125	2.063	1.531	0.766	0.383
p_TDL	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$	$\begin{bmatrix}2\end{bmatrix}$	$\begin{bmatrix}3\end{bmatrix}$	$\begin{bmatrix}2\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$
a_TDL	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0.5\end{bmatrix}$	$\begin{bmatrix}1.25\end{bmatrix}$	$\begin{bmatrix}2.125\end{bmatrix}$	$\begin{bmatrix}2.063\end{bmatrix}$	$\begin{bmatrix}1.531\end{bmatrix}$	$\begin{bmatrix}0.766\end{bmatrix}$



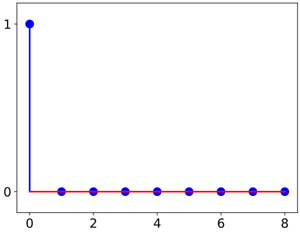
Input Sequence $p(t)$



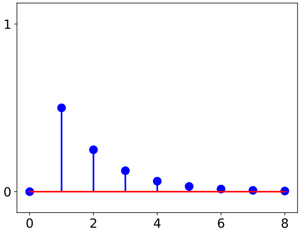
Output Sequence $a(t)$

Impulse response

t	0	1	2	3	4	5	6	7	8
$p(t) = \delta(t)$	1	0	0	0	0	0	0	0	0
$a(t) = h(t)$	0	$\frac{1}{2}$	$\frac{1}{2^2}$	$\frac{1}{2^3}$	$\frac{1}{2^4}$	$\frac{1}{2^5}$	$\frac{1}{2^6}$	$\frac{1}{2^7}$	$\frac{1}{2^8}$
p_TDL	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}1\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$
a_TDL	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}0\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2^2}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2^3}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2^4}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2^5}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2^6}\end{bmatrix}$	$\begin{bmatrix}\frac{1}{2^7}\end{bmatrix}$



Input Impulse $\delta(t)$



Impulse Response $h(t)$

Infinite impulse response

- The feedback (recurrent) connection produced an infinite impulse response.
- Theoretically, the memory of this network is infinite.
- The memory decays exponentially, so the effective memory length is finite.
- As the feedback weight comes closer to 1, the effective memory length gets longer.
- An IIR system can be approximated by an FIR system, if the TDL length for $p(t)$ is large enough.

Infinite impulse response system

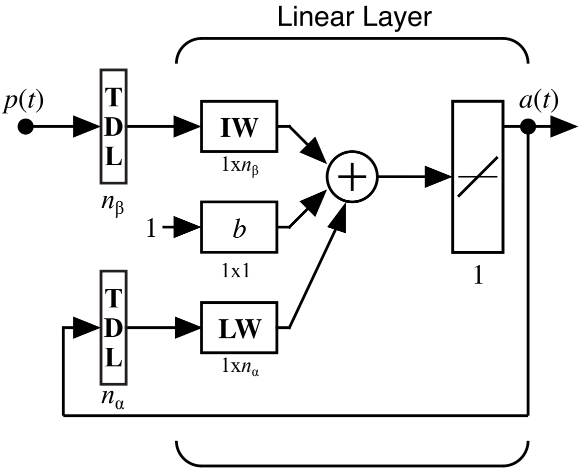
General IIR system

$$a(t) = \sum_{i=1}^{n_\alpha} \alpha_i a(t-i) + \sum_{i=1}^{n_\beta} \beta_i p(t-i)$$

Difference Equation

$$\begin{aligned} a(t) =& \alpha_1 a(t-1) + \alpha_2 a(t-2) + \ldots + \alpha_{n_\alpha} a(t-n_\alpha) \\ &+ \beta_1 p(t-1) + \beta_2 p(t-2) + \ldots + \beta_{n_\beta} p(t-n_\beta) \end{aligned}$$

IIR network



$$\mathbf{IW} = \begin{bmatrix} \beta_1 & \beta_2 & \cdots & \beta_{n_\beta} \end{bmatrix}, \mathbf{LW} = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{n_\alpha} \end{bmatrix}, b = 0$$

Stability of IIR systems

- If we let the delay operator be D: $Da(t) = a(t-1)$, we can define the following operators:

$$\alpha(D) = 1 - \alpha_1 D - \alpha_2 D^2 - \ldots - \alpha_{n_\alpha} D^{n_\alpha}$$
$$\beta(D) = \beta_1 D + \beta_2 D^2 + \ldots + \beta_{n_\beta} D^{n_\beta}$$

- Then we can conveniently represent our linear system as

$$a(t) = \frac{\beta(D)}{\alpha(D)} p(t)$$

- Setting the denominator of the transfer function to zero is the characteristic equation:

$$\alpha(D) = \prod_{i-1}^{n_\alpha} (1 - \lambda_i D) = 0$$

- The system poles (λ_i) must be inside the unit circle for stability.

Example

- Consider the following difference equation.

$$a(t) = a(t-1) - 0.24a(t-2) + p(t-1)$$

- Using the D operator, we can write:

$$a(t) = \frac{D}{1 - D + 0.24D^2} p(t)$$

- The characteristic polynomial can be factored.

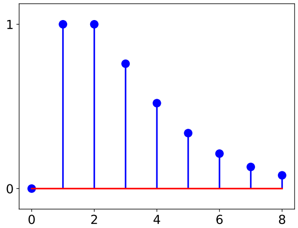
$$\alpha(D) = 1 - D + 0.24D^2 = (1 - 0.6D)(1 - 0.4D)$$

- Because the poles $\lambda_1 = 0.6$ and $\lambda_2 = 0.4$ are inside the unit circle, the system is stable.

Impulse response

- The impulse response is

$$h(t) = 5\,(0.6)^t - 5\,(0.4)^t$$



- The general impulse response has the form:

$$h(t) = A_1\,(\lambda_1)^t + A_2\,(\lambda_2)^t \cdots + A_n\,(\lambda_n)^t$$

- The closer the poles are to the unit circle, the longer the effective memory of the system.

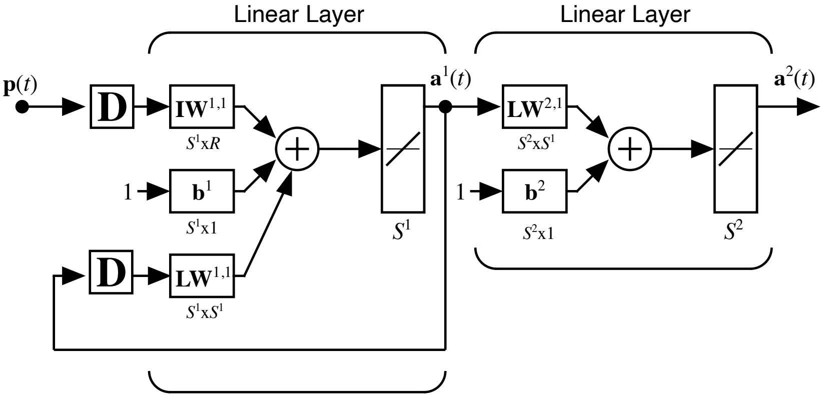
State space representation of linear dynamic systems

- In addition to difference equations, linear dynamic systems can be represented by state space models.
- Instead of a difference equation of order n, we have n first order difference equations.

$$
\begin{aligned}
q_i(t+1) =& \phi_{i,1} q_1(t) + \phi_{i,2} q_2(t) + \cdots + \phi_{i,n} q_n(t) \\
& + \gamma_{i,1} p_1(t) + \gamma_{i,2} p_2(t) + \cdots + \gamma_{i,m} p_m(t) \\
a_j(t) =& \psi_{j,1} q_1(t) + \psi_{j,2} q_2(t) + \cdots + \psi_{j,n} q_n(t) \\
\mathbf{q}(t+1) =& \mathbf{\Phi} \mathbf{q}(t) + \mathbf{\Gamma} \mathbf{p}(t) \\
\mathbf{a}(t) =& \mathbf{\Psi} \mathbf{q}(t)
\end{aligned}
$$

- Where the q_i are the states of the system. Knowing $\mathbf{q}(0)$ allows us to solve for all future states and outputs.
- The eigenvalues of $\mathbf{\Phi}$ are the system poles λ_i.

State space model as neural network

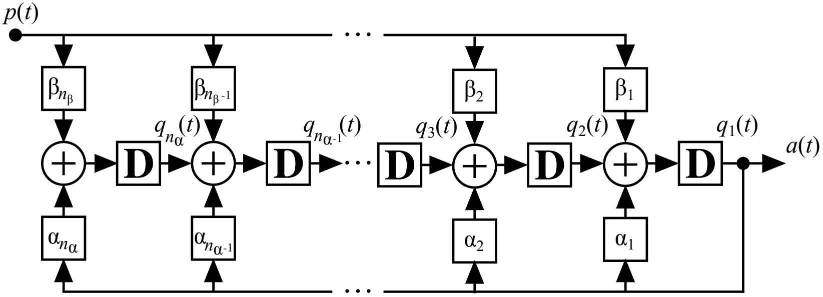


$$\mathbf{LW^{1,1}} = \mathbf{\Phi}, \quad \mathbf{IW^{1,1}} = \mathbf{\Gamma}, \quad \mathbf{LW^{2,1}} = \mathbf{\Psi}, \quad \mathbf{a}^1 = \mathbf{q}, \quad \mathbf{a}^2 = \mathbf{a}$$

Converting from difference equation to state space

Difference Equation

$$
\begin{aligned}
a(t) = & \alpha_1 a(t-1) + \alpha_2 a(t-2) + \ldots + \alpha_{n_\alpha} a(t-n_\alpha) \\
& + \beta_1 p(t-1) + \beta_2 p(t-2) + \ldots + \beta_{n_\beta} p(t-n_\beta)
\end{aligned}
$$

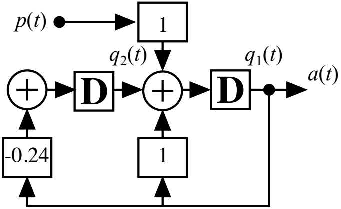


State space canonical form

$$
\begin{aligned}
\mathbf{q}(t+1) &= \begin{bmatrix} \alpha_1 & 1 & 0 & \cdots & 0 \\ \alpha_2 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_{n_\alpha - 1} & 0 & 0 & \cdots & 1 \\ \alpha_{n_\alpha} & & & \cdots & 0 \end{bmatrix} \mathbf{q}(t) + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{n_\beta - 1} \\ \beta_{n_\beta} \end{bmatrix} p(t) \\
a(t) &= \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \mathbf{q}(t)
\end{aligned}
$$

Example

$$a(t) = a(t-1) - 0.24a(t-2) + p(t-1)$$



$$\begin{aligned} q_1(t+1) &= 1q_1(t) + 1q_2(t) + 1p(t) \\ q_2(t+1) &= -0.24q_1(t) + 0q_2(t) + 0p(t) \\ a(t) &= 1q_1(t) + 0q_2(t) \end{aligned}$$

$$\begin{aligned} \mathbf{q}(t+1) &= \begin{bmatrix} 1 & 1 \\ -0.24 & 0 \end{bmatrix} \mathbf{q}(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} p(t) \\ a(t) &= \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{q}(t) \end{aligned}$$