



TIME-SERIES PREDICTION AND ELMAN NETWORK

INT301 Bio-computation, Week 10, 2021



Time Series

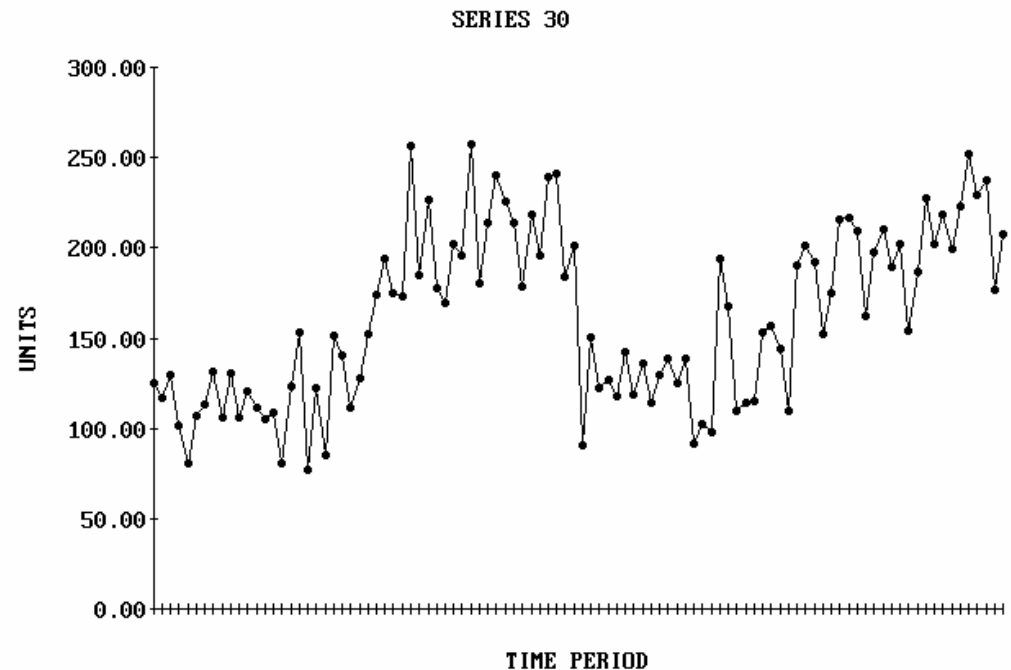
- What is a time series?

- Time Series is a series of timely ordered, comparable observations y_t recorded in equidistant time intervals

- Examples:

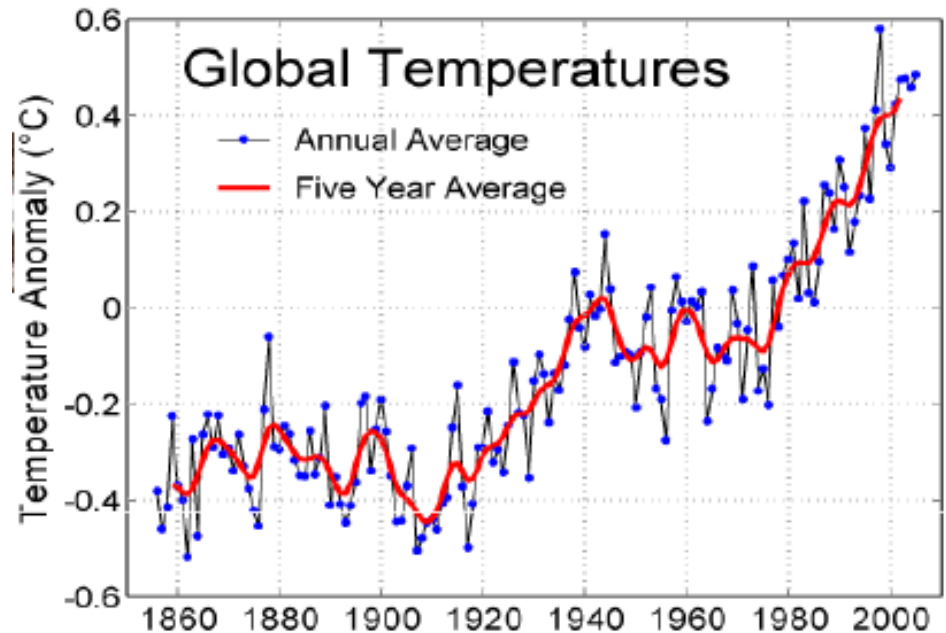
- Stock prices
 - Temperature readings

Y_t represents the t^{th} period observation, $t=1,2 \dots n$

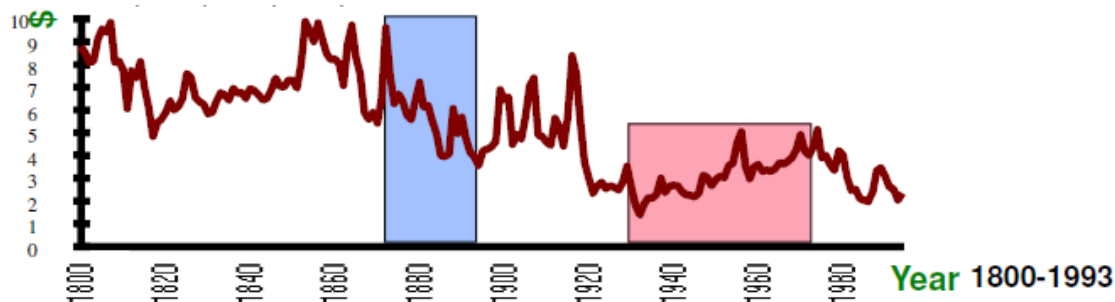


Examples of Time Series

Global
Warming



Real Copper Prices in \$ per KG



Electroencephalography (EEG)





Time Series Models

- Modeling of a time series
 - Values of a time series at successive time indices are often correlated. Otherwise, prediction is impossible.
 - The statistical properties do not change with respect to time
 - WSS (wide-sense stationarity) random process
 - The mean and autocovariance do not vary with respect to time
 - Variance is finite for all times

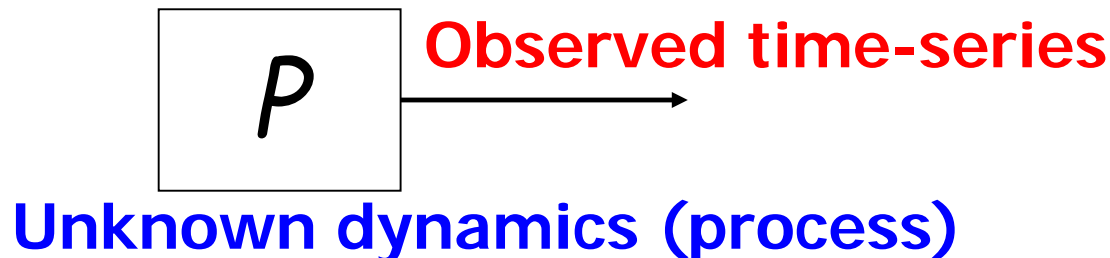


Time Series Models

- In statistics and machine learning, a time-series is often described by a sequence of vectors (or scalars) which depend on time t :

$$\{x(t_0), x(t_1), \dots, x(t_{i-1}), x(t_i), x(t_{i+1}), \dots\}$$

It's the output of some process P that we are interested in:





Time Series Models

- The continuous signal $x(t)$ is sampled at discrete points to get a series
- In uniform sampling, if sampling period is Δt

$$\{x[t]\} = \{x(0), x(\Delta t), x(2\Delta t), x(3\Delta t), \dots\}$$

Problem of Predicting the Future

Extending backward from time t , we have time series $\{x[t], x[t-1], \dots\}$. From this, we now want to estimate x at some future time

$$\hat{x}[t+s] = f(x[t], x[t-1], \dots)$$

s is called the **horizon of prediction**. At beginning, let's predict just one time sample into the future, $\Rightarrow s = 1$.

This is a function approximation problem.

Here's how we'll solve it:

1. Assume a **generative model**.
2. For every point $x[t_i]$ in the past, train the generative model with what preceded t_i as the inputs and what followed t_i as the desired.
3. Now run the model to predict $\hat{x}[t+s]$ from $(x[t], x[t-1], \dots)$



Time Series Prediction

- Time series examples
 - Financial (e.g. stocks, exchange rates)
 - Physically observed (e.g. weather, river flow)
- Why is it important?
 - Preventing undesirable events by forecasting the event, identifying the circumstances preceding the event, and taking corrective action so the event can be avoided;
 - Forecasting undesirable, yet unavoidable, events to preemptively lessen their impact;
 - Profiting from forecasting (e.g., financial markets)

Time Series Prediction

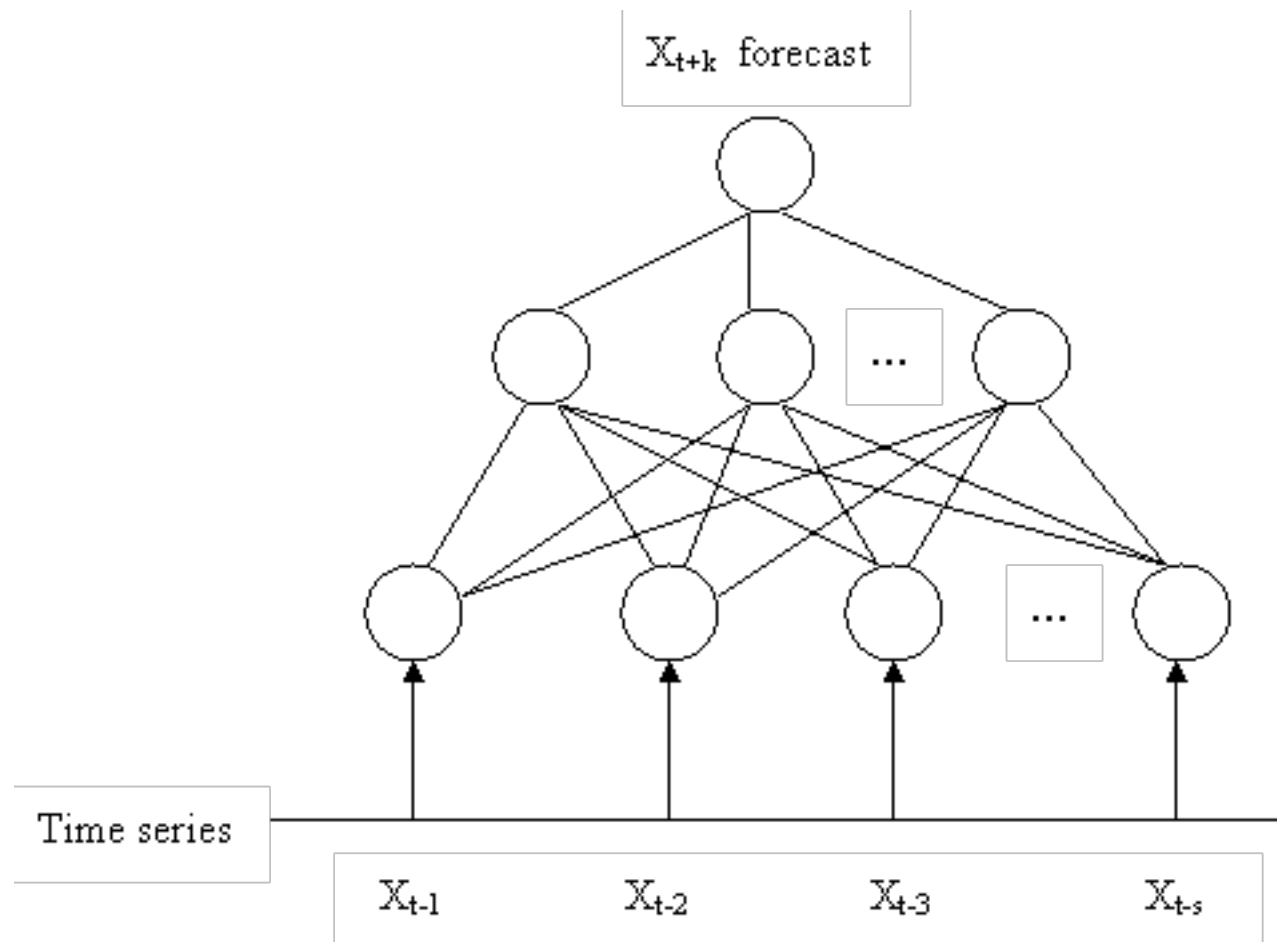
- Why is it difficult?
 - Limited quantity of data
 - observed data series sometimes too short to partition
 - Noise
 - erroneous data points
 - Non-stationarity
 - fundamentals change over time, nonstationary
 - Forecasting method selection
 - statistics, artificial intelligence
- Neural networks have been widely used as time series forecasters: most often these are feed-forward networks which employ a *sliding window* over the input sequence.
- The neural network sees the time series X_1, \dots, X_n in the form of many mappings of an input vector to an output value.

Time Series Prediction with ANN



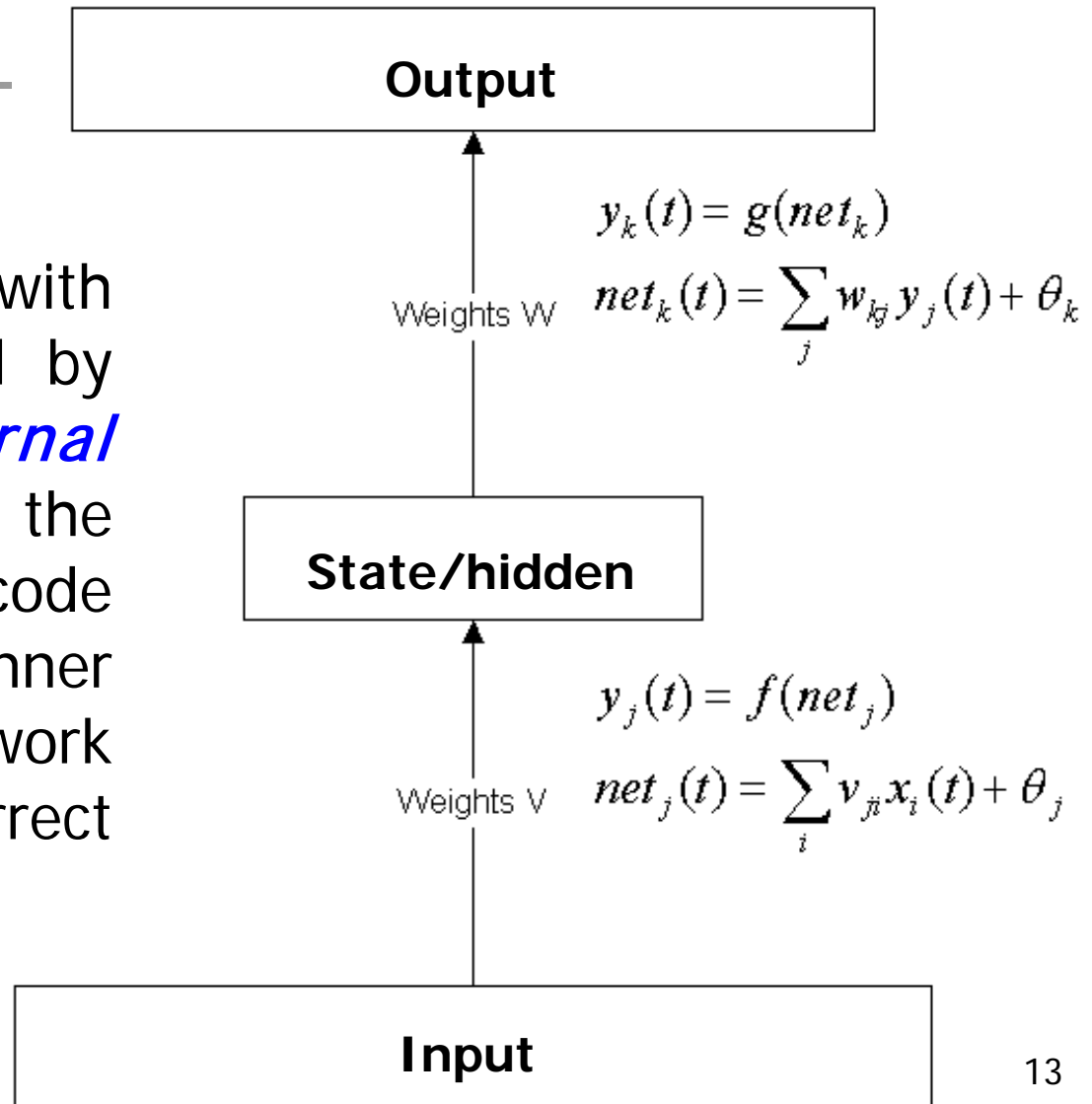
- A number of adjoining data points of the time series (the input window $X_{t-s}, X_{t-s+1}, \dots, X_t$) are mapped to the interval $[0,1]$ and used as activations for the input of the input layer.
- The size **s** of the input window corresponds to the number of input units of the neural network.
- In the forward path, the activations are propagated over hidden layers to output units. The error used for the BP training is computed by comparing the value of the output unit with the value of the time series at time $t+1$.
- Training a MLP network with BP learning algorithm usually requires that all representations of the input set (called one *epoch*) are presented many times.

Time Series Prediction with ANN



Revisit of feed forward NN

A feed forward NN with hidden nodes, trained by e.g. BP, develops *internal representations* for the input signals that recode those signals in a manner that enables the network to produce the correct output.





Motivation for dynamic networks

- Sometimes, we require our model to be sensitive to inputs that were presented some time ago.
- In other words, our requirement is not met by a function (no matter how complex it may be) and demands **a model which maintains a *state*, or *memory***, over several presentations.
- Which applications need memory?
 - **sequential input** (e.g. language understanding, robot exploration, ...),
 - **sequential output** (e.g. speech, route planning, ...), or combinations of the above.



Sequence Learning

- MLP & RBF networks are *static networks*, i.e., they learn a mapping from a single input signal to a single output response for an arbitrary large number of pairs.
- *Dynamic networks* learn a mapping from a single input signal to a sequence of response signals, for an arbitrary number of pairs (signal, sequence).
- Typically the input signal to a dynamic network is an element of the sequence and then the network produces the rest of the sequence as a response.
- *To learn sequences we need to include some form of memory (short term memory) to the network.*

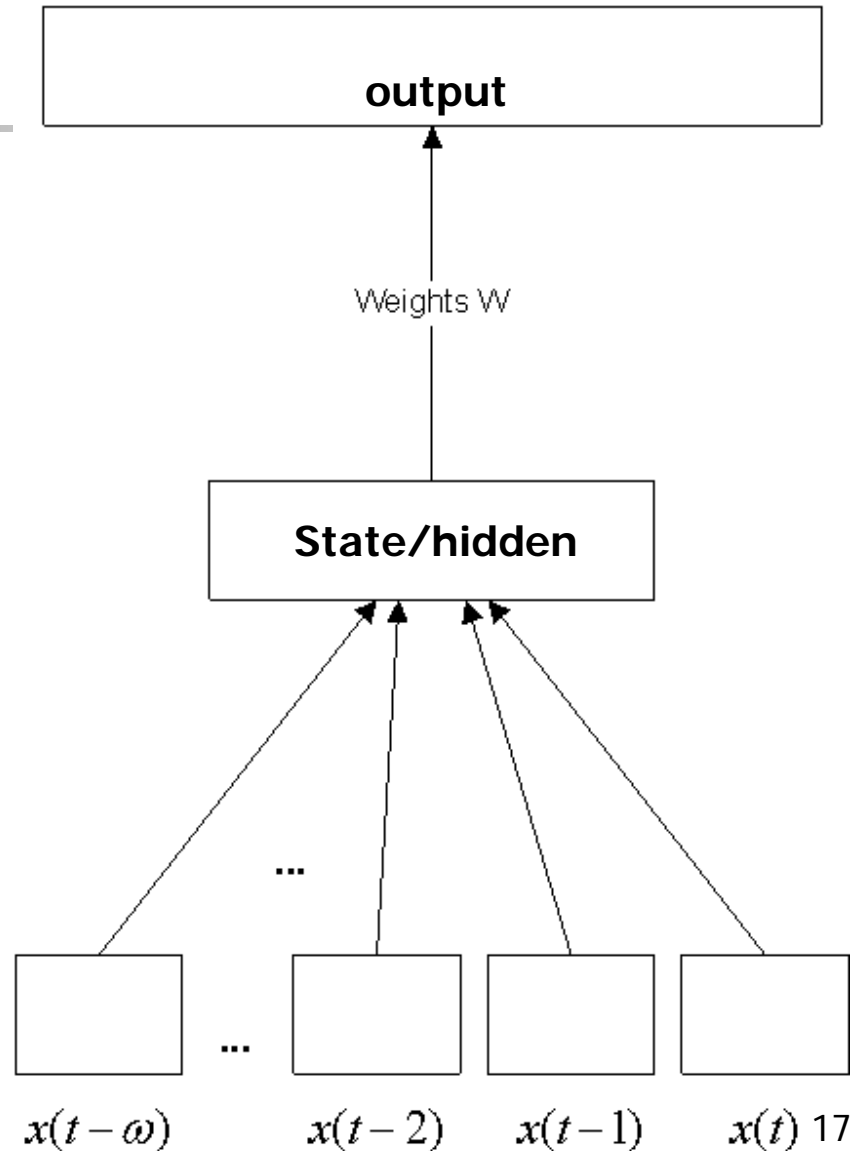


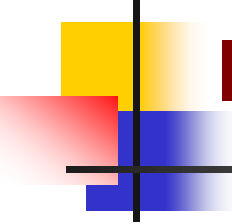
Sequence Learning

- We can *introduce* *memory effects* with two principal ways:
 - **Implicit**: e.g., Time lagged signal as input to a static network or as recurrent connections
 - **Explicit**: e.g., Temporal Backpropagation Method
- In the implicit form, we assume that the environment from which we collect examples of (input signal, output sequence) is *stationary*. For the explicit form the environment could be *non-stationary*, i.e. the network can track the changes in the structure of the signal.

Time Delayed Networks

- Signals can be **buffered externally** and presented at **additional input nodes** (input banks).
- This is called **Time Delayed Networks**
- Time Delayed Networks illustrate the implicit representation approach for sequence learning, which combine a static network (e.g. MLP / RBF) with a memory structure.



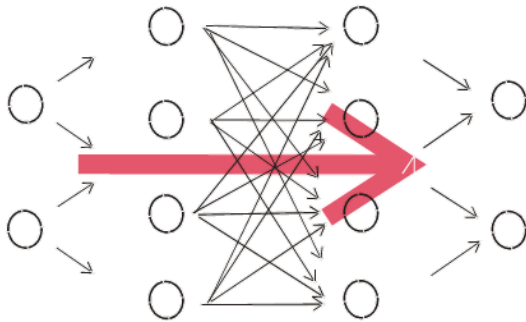


Dynamical Neural Networks

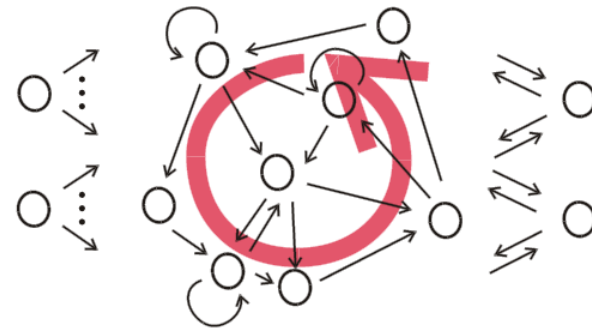
Introducing time

- **Recurrent networks** can explicitly deal with inputs that are presented *sequentially*, as they would almost always be in real problems.
- Recurrent networks are fundamentally different from feed forward networks in that they **not only operate upon an input but also a state**.
- The ability of the net to reverberate and sustain activity can serve as a **working memory**.
- Such nets are **Dynamical Neural Networks**.

From feed-forward to recurrent NN



- Connections only "from left to right", no connection cycle
- Activation is fed forward from input to output through "hidden layers"
- No memory



- At least one connection cycle
- Activation can "reverberate", persist even with no input
- System with memory
- Mathematically, RNNs implement dynamical systems.

Elman Network

- Elman neural network was proposed by Jeffrey L. Elman, 1990.
- It is a simple recurrent neural network
- Elman network has a powerful prediction capability

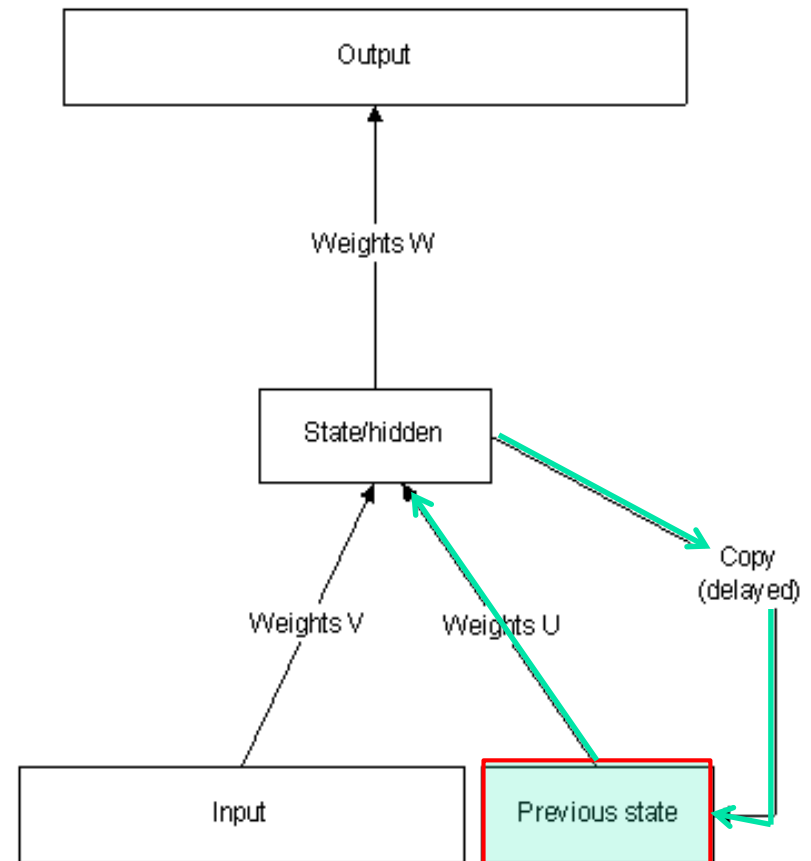


Elman Network

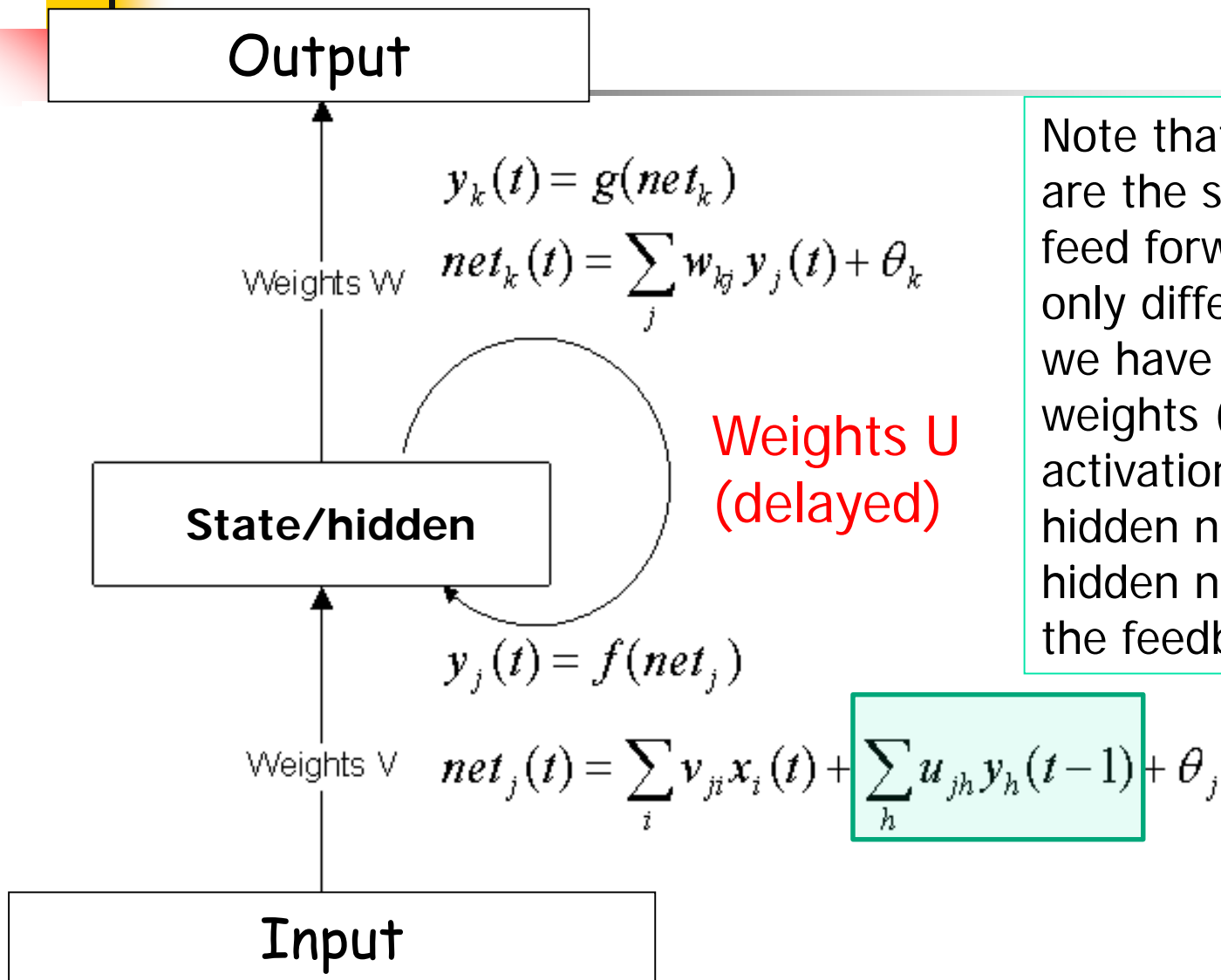
- Elman network which typically distinguishes between a **state-output function** and an **input-state mapping**.

$$y = F_w(z)$$
$$z^t = F_v(z^{t-1}, x)$$

where y is the output, x is the input and z is the state. t is an index in time.



In Elman net, the internal representations are now engaged in the next step as an additional input.



Note that all equations are the same as for the feed forward case. The only difference is that we have another set of weights (U) which feed activations from the hidden nodes to the hidden nodes. Importantly, the feedback is delayed.

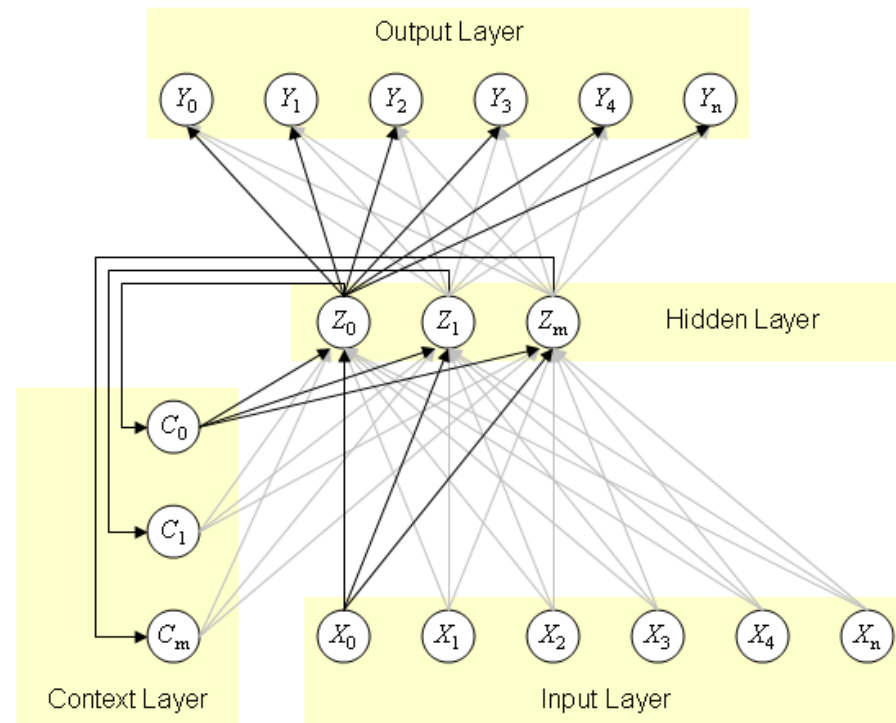
Elman Network

Four layers of Elman network

- **Input layer**
- **Hidden layer** forming internal representation
- **Context layer**

The value of the context neuron is used as an extra input signal for all the neurons in the hidden layer one time step later. In an Elman network, the weights from the hidden layer to the context layer are set to one and are fixed because the values of the context neurons have to be copied exactly

- **Output layer** linear combination





Learning algorithm for Elman Network

- Elman network adopts BP algorithm
 - The error function:

$$E = \sum_{k=1}^n [y(k) - d(k)]^2$$

where $d(k)$ is the expected output (target)



THANK YOU



VISIT US

WWW.XJTLU.EDU.CN



FOLLOW US

@XJTLU



Xi'an Jiaotong-Liverpool University

西交利物浦大學