Machine Learning

Lecture 9

Recurrent multi-layer neural networks

Recurrent Networks

- Feed forward networks:
  - Information only flows one way
  - One input pattern produces one output
  - No sense of time (or memory of previous state)
- Recurrency
  - Nodes connect back to other nodes or themselves
  - Information flow is multidirectional
  - Sense of time and memory of previous state(s)
- Biological nervous systems show high levels of recurrency (but feed-forward structures exists too)

Recurrent Connections and Sequences

- A sequence is a succession of patterns that relate to the same object.
- For example, letters that make up a word or words that make up a sentence.
- Sequences can vary in length. This is a challenge.
- How many inputs should there be for varying length inputs?
- Several feed-forward alternatives: shift registers, tapped delay lines, etc

The simple recurrent network

- Jordan network has connections that feed back from the output to the input layer and also some input layer units feed back to themselves.
- Useful for tasks that are dependent on a sequence of a successive states.
- The network can be trained by backpropogation.
- The network has a form of short-term memory.
- Simple recurrent network (SRN) has a similar form of short-term memory.

Jordan recurrent network

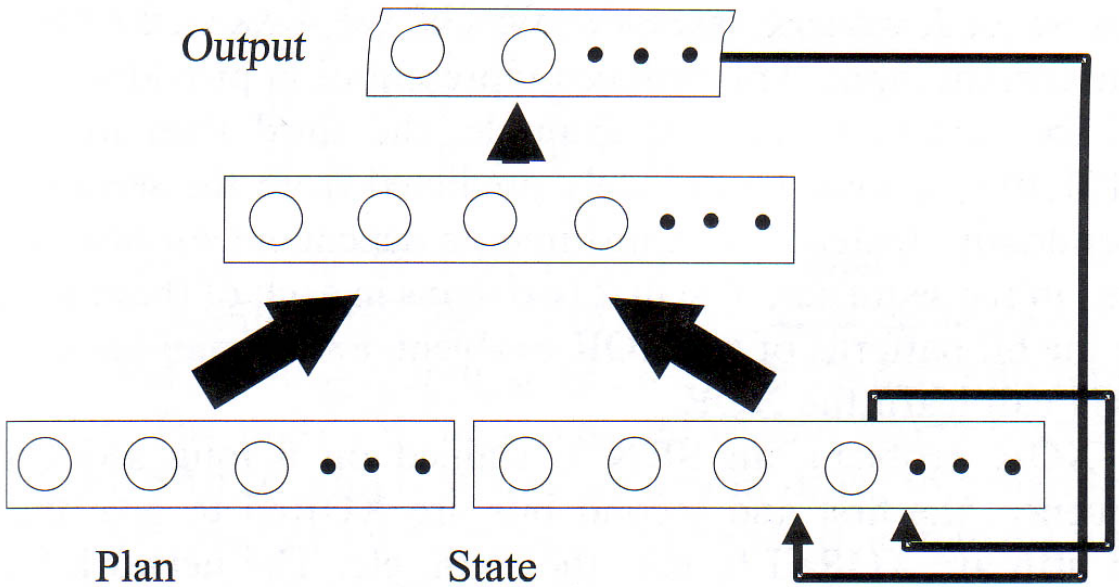
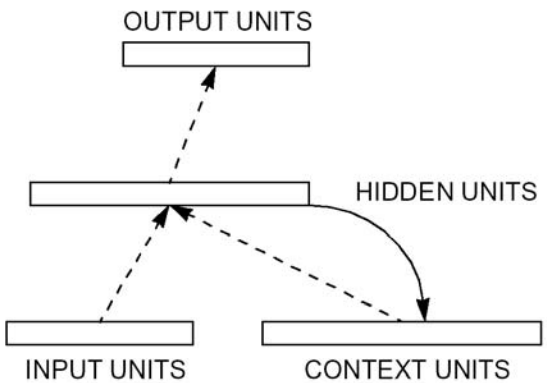
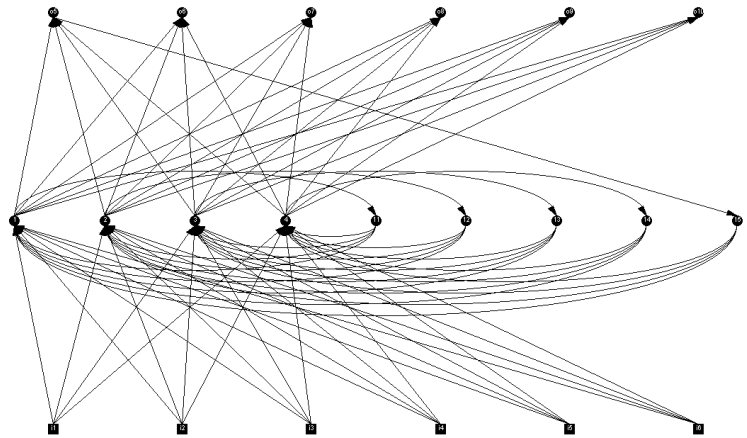


Figure 5.6 Jordan network.

Elman Nets (SRN)

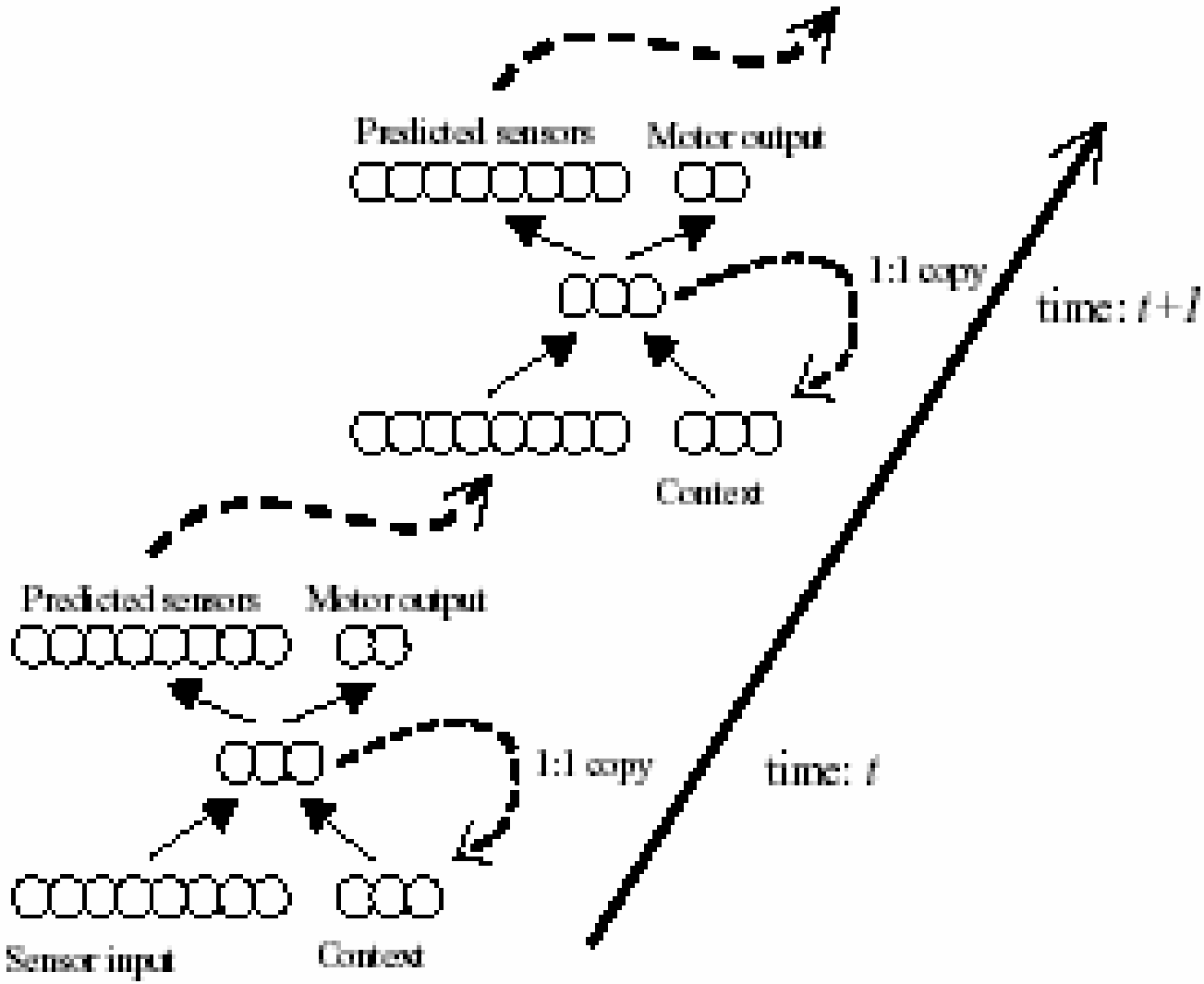
- *Elman nets* are feed forward networks with partial recurrency





- Unlike feed forward nets, Elman nets have a *memory* or *sense of time*

Robot sensory prediction with SRN



Short-term memory in SRN

- The context units remember the previous internal state.
- Thus, the hidden units have the task of mapping both an external input and also the previous internal state to some desired output.

- An SRN can predict the next item in a sequence from the current and preceding input.
- In an SRN the hidden units are fed back into the input layer.
- That is: The hidden units are copied into input units. Input units supply weighted sums into hidden units in the next time step.
  ****
- The hidden units represent **an internal reduced representation of the data in the sequence that precedes the current input**.

- The reduced representation provides context which is essential for certain tasks.

- As an example SRN can learn to solve XOR problem.

- Input – 101000011110101…..
  - Bit 3 is XOR of bit 1 and 2, bit 6 is XOR of 4 and 5, and so on

- An SRN with 1 input, two context, two hidden and one output unit was trained on a sequence of 3000 bits.

  input: 1 0 1 0 0 0 0 1 1 1 1 0 1 0 1. . .

  output: 0 1 0 0 0 0 1 1 1 1 0 1 0 1 ?. . .

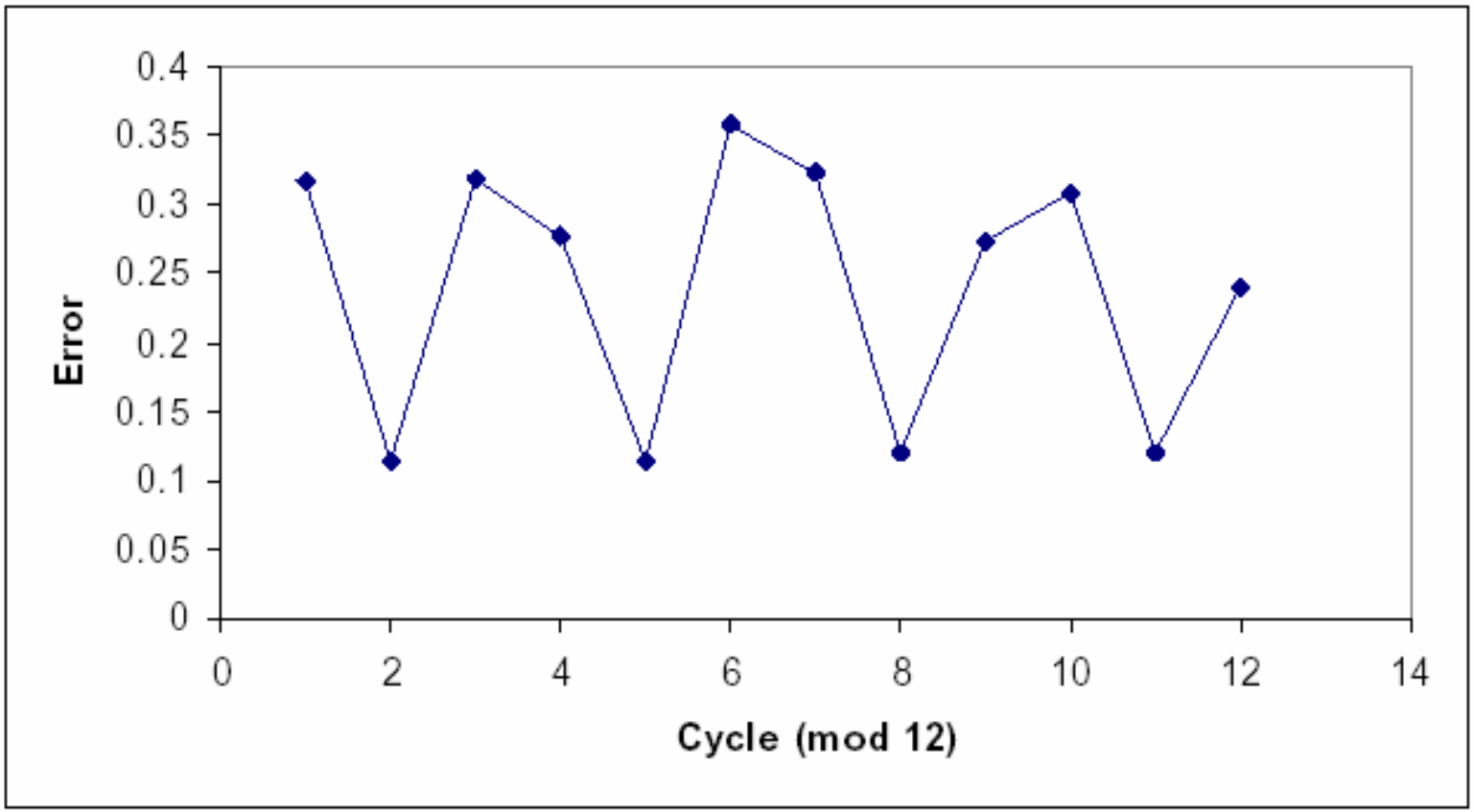


Figure 3. Graph of root mean squared error over 12 consecutive inputs in sequential XOR task. Data points are averaged over 1200 trials.

A recurrent backpropogation network

- A backpropogation network need not be strictly feedforward and can have recurrent connections.
- A unit can feed to itself, to units in the same or lower levels.
- A recurrent connection feeds back activation that will affect the output from the network during subsequent iterations.
- For every recurrent network there is a feedforward network with identical behavior.

A recurrent backpropogation network (Connections from output to input layer)

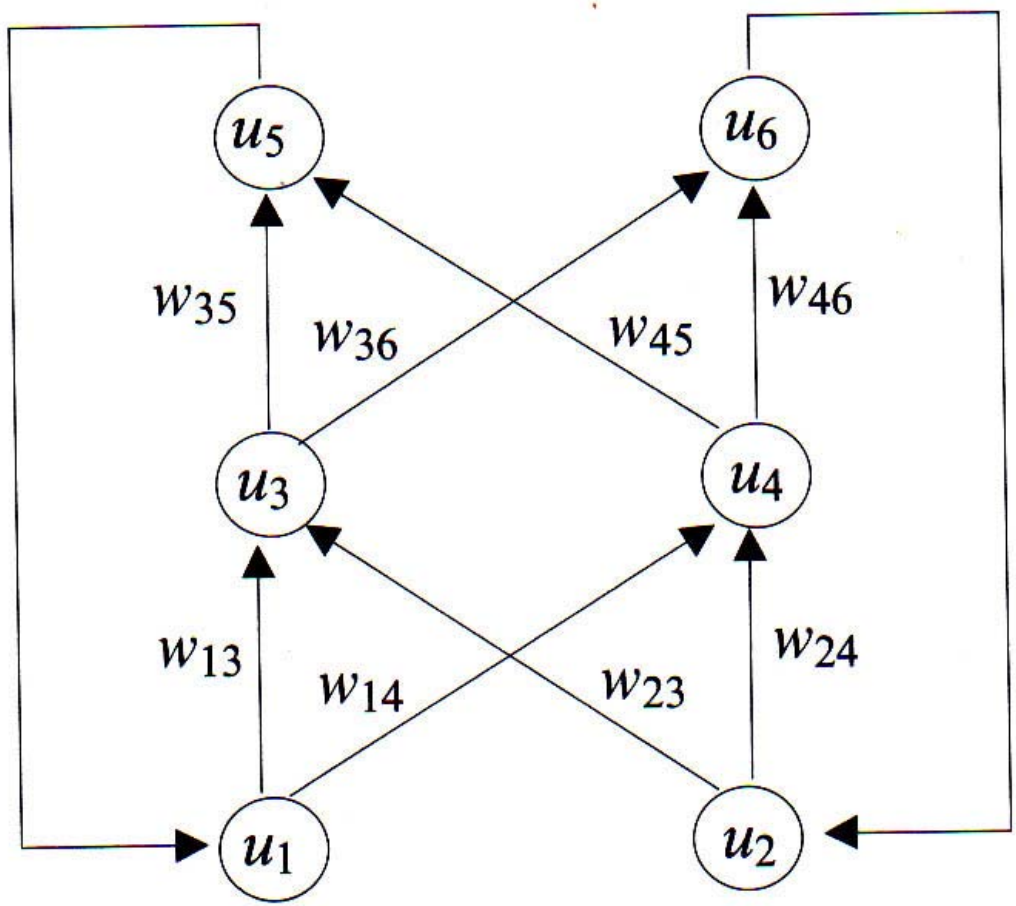
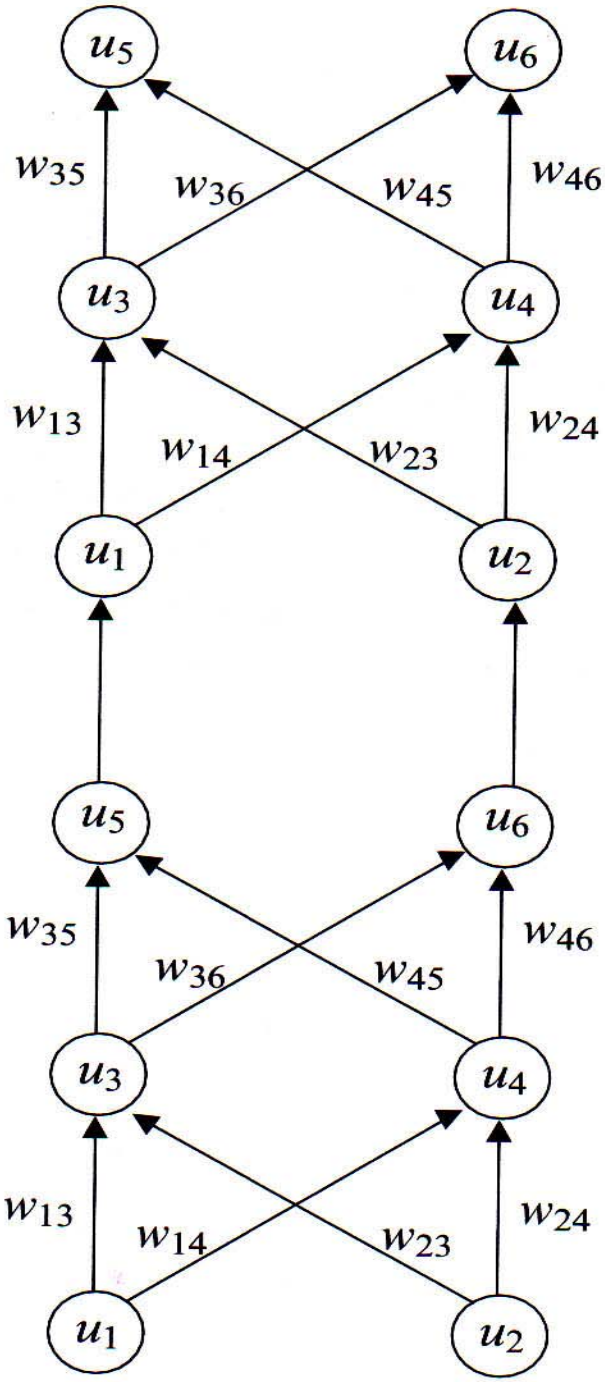
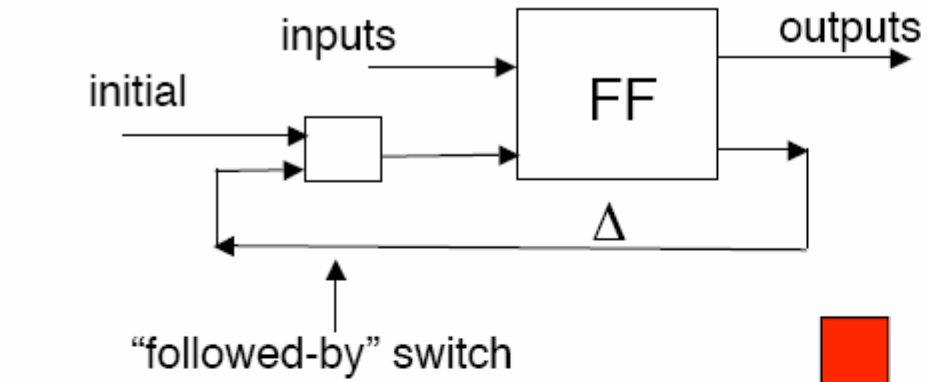


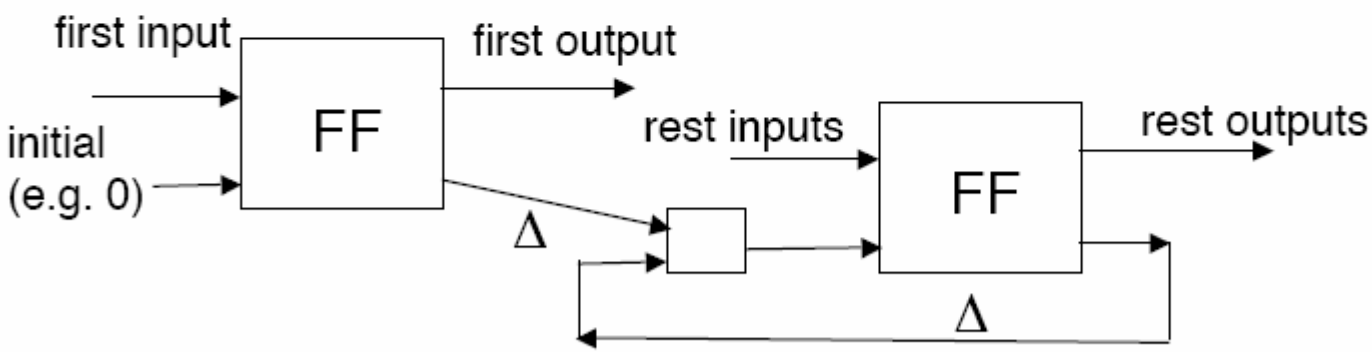
Figure 5.2 A recurrent backpropagation network.

An expanded version of the network in the previous figure



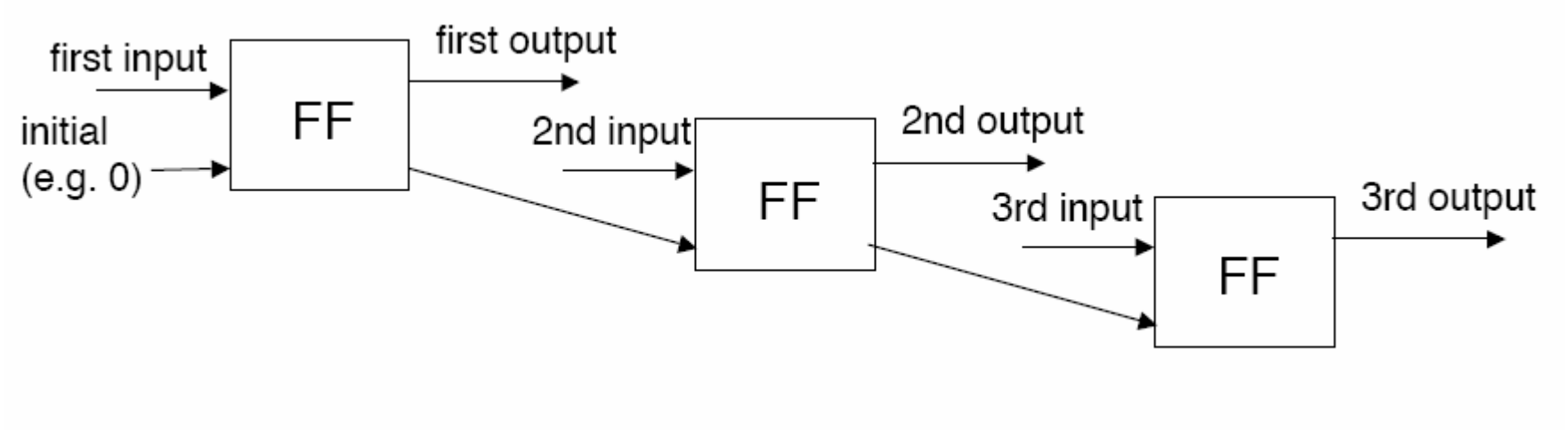
Unrolling idea



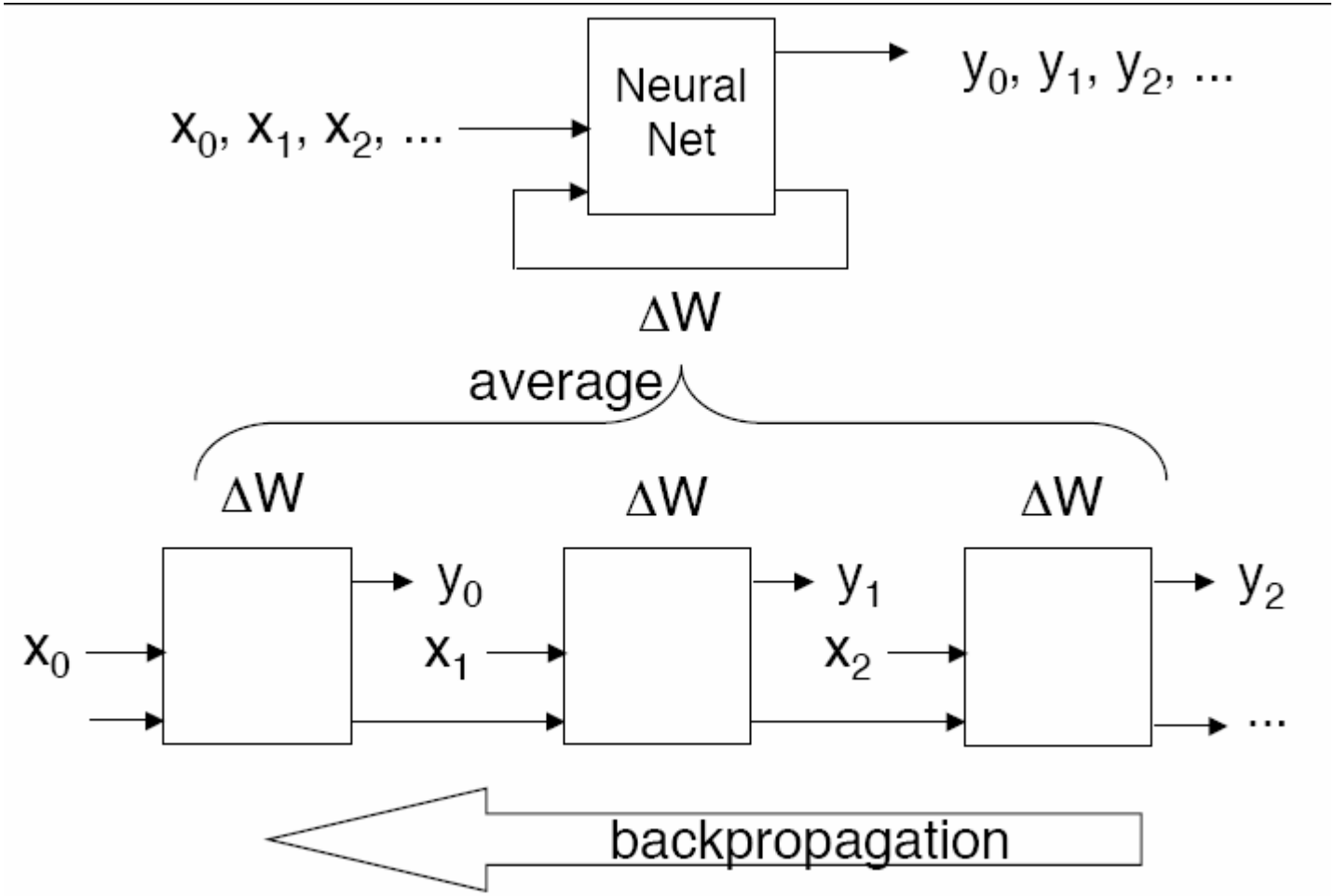


repeat the construction
on the right subsystem

Unrolling idea (2)



Error back propagation through time



Backprop Through Time (2)

- During training a pattern is presented to the network and a feedforward pass is made.
- Each network copy corresponds to a time step.
- The weight changes calculated for each network copy are summed before individual weights are adapted.
- The set of weights for each copy(time step) always remain the same.

Error back propagation through time (3)

Output unit k's target at time t is denoted by $d_k(t)$. Using mean squared error, k's error signal is

$$\vartheta_k(t) = f'_k(net_k(t))(d_k(t) - y^k(t)),$$

where

$$y^i(t) = f_i(net_i(t))$$

is the activation of a non-input unit i with differentiable activation function f_i,

$$net_i(t) = \sum_j w_{ij} y^j(t-1)$$

is unit i's current net input, and w_{ij} is the weight on the connection from unit j to i.

Some non-output unit j's backpropagated error signal is

$$\vartheta_j(t) = f'_j(net_j(t)) \sum_i w_{ij} \vartheta_i(t+1).$$

Real-Time Recurrent Learning (RTRL) (Williams&Zipser, 1989)

- Trains a network without unrolling by deriving a recurrence for weight updates.
- The algorithm is “real-time” in the sense that the weights can be updated while the input sequence is being applied.

Real-Time Recurrent Learning (2)

- $\Delta w_{ij}(t) = \eta \sum_k e_k(t) p^k_{ij}(t)$

  k varying over the fed-back inputs

where

- $p^k_{ij}(t+1) = f'_k(net_k(t)) \left[\sum_r w_{kr} p^r_{ij}(t) + \delta_{ij} z_j(t)\right]$

  r varying over the fed-back inputs

- $p^k_{ij}(0) = 0$
- $z_j(t)$ is the value of the output of neuron j at time t
- δ_{ij} is the Kronecker delta (= 1 if i=j, 0 otherwise)

Real-Time Recurrent Learning (3)

$$e_k(t) = \begin{cases} d_k(t) - y_k(t) & \text{if } k \in T(t) \\ 0 & \text{otherwise} \end{cases}$$

Difficulty of usage of gradient descend learning algorithms in RNN

- These algorithms often are sensitive to details of algorithm, in particular, to number of time steps unrolled in the case of back propagation through time

Sequence Completion Example

- Rumelhart et. al. (1986) performed experiments with recurrent connections.
- Train a network to perform sequence completion.
- A sequence consisted of 6 characters: two letters from the set {A,B,C,D,E} and 4 being numbers from the integer set {1,2,3}.
- For example, A can denote “12” and C can denote “13”.
- The complete sequence prefixed by AC would be “AC1213”.
- The complete sequence prefixed by AA would be “AA1212”.

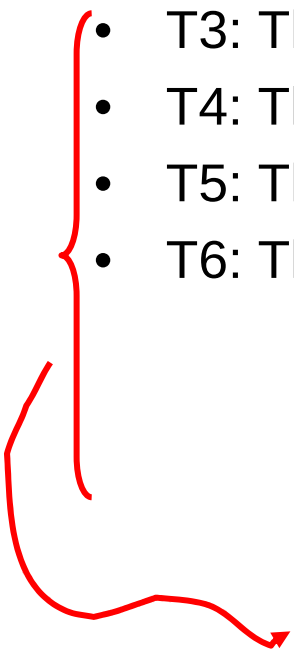
Rumelhart network

- The network has:
  - 5 input units, one for each letter.
  - 3 output units, one for each number.
  - 30 hidden units.

  - Each hidden unit is connected to itself and every other hidden unit.

Training Steps

- T1: Input unit corresponding to the first letter is 1 and other inputs are off.
- T2: Input unit corresponding to the second letter is 1 and other inputs are off.
- T3: The target output is the third character in the sequence.
- T4: The target output is the fourth character in the sequence.
- T5: The target output is the fifth character in the sequence.
- T6: The target output is the sixth character in the sequence.



Output generation at different time steps.

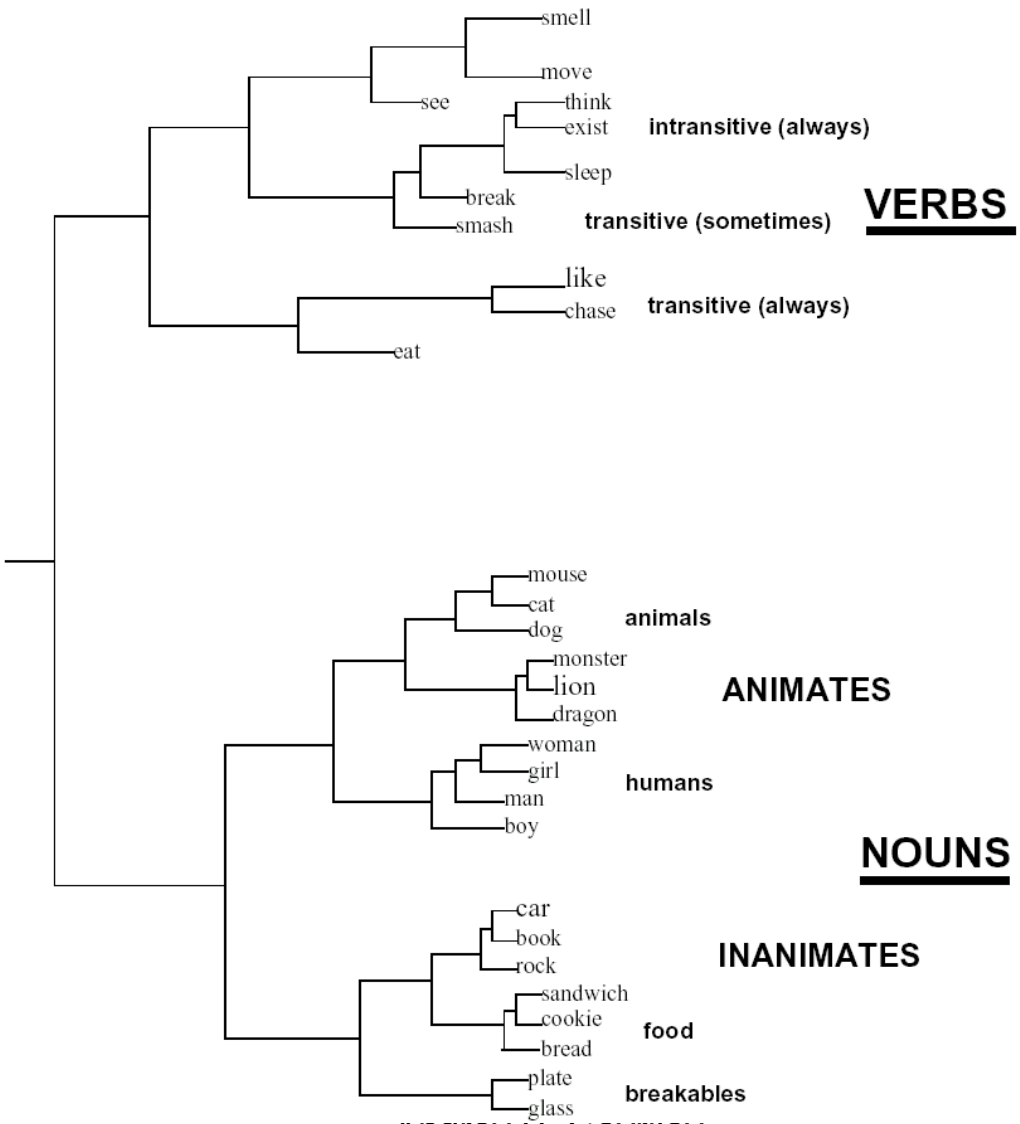
Rumelhart Network Results

- All hidden and output unit activations started at 0.2
- Network is trained on 20 sequences.
- Given first two characters, the rest of the characters were completed in five test cases.
- The completion could be achieved even if there were delays between the presentation of the first two letters.

Classic experiment on language acquisition and processing (Elman, 1990)

- **Task**
  - Elman net to predict successive words in sentences.
- **Data**
  - Suite of sentences, e.g.
    - "The boy catches the ball."
    - "The girl eats an apple."
  - Words are input one at a time
- **Representation**
  - Binary representation for each word, e.g.
    - 0-1-0-0-0 for "girl"
- **Training method**
  - Backpropagation

- Internal representation of words



Grammatical inference by RNN (S.Lawrence, 1996)

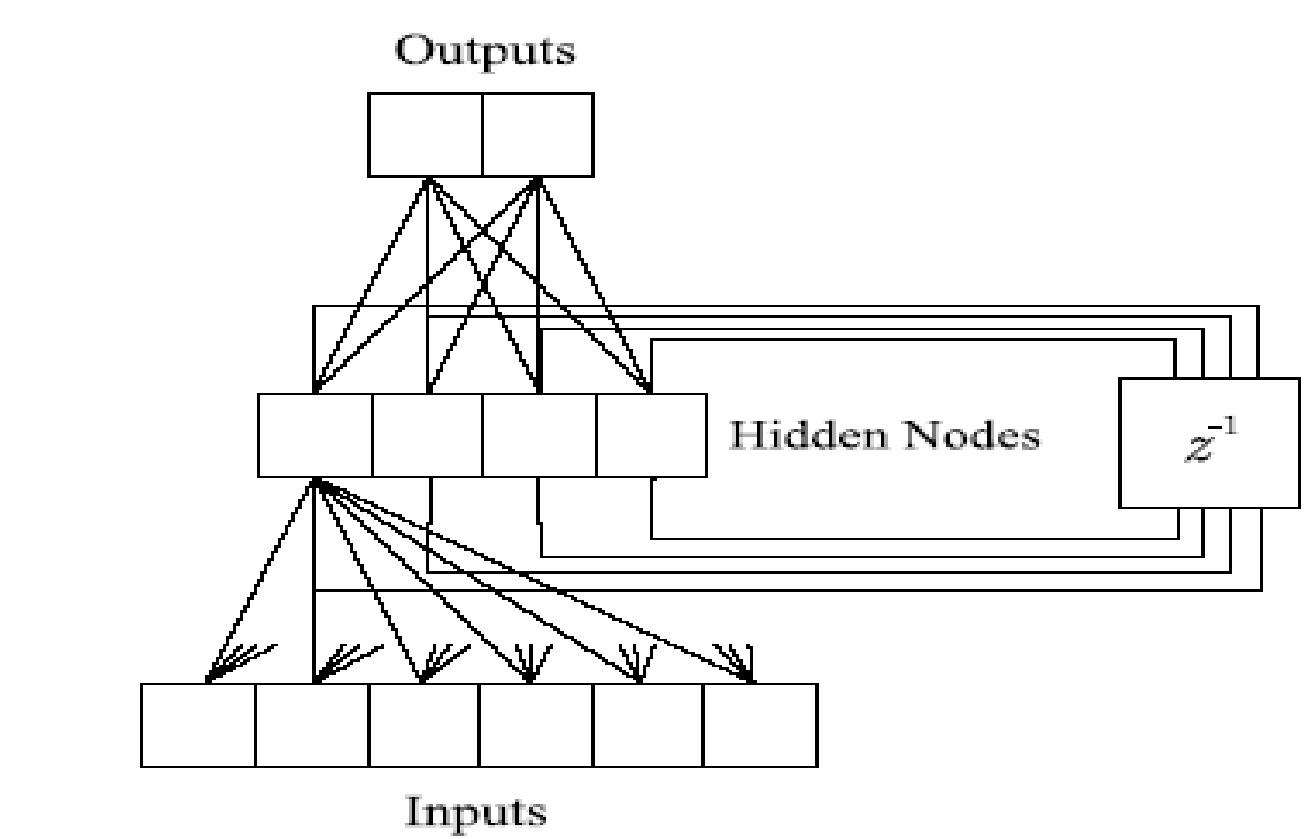
*I am happy John to be here	I believe John to be here
I am happy for John to be here	*I believe for John to be here
I am happy to be here	*I believe to be here

Asterisk marks the ungrammaticality.
Task is grammaticality judgment of sentence.

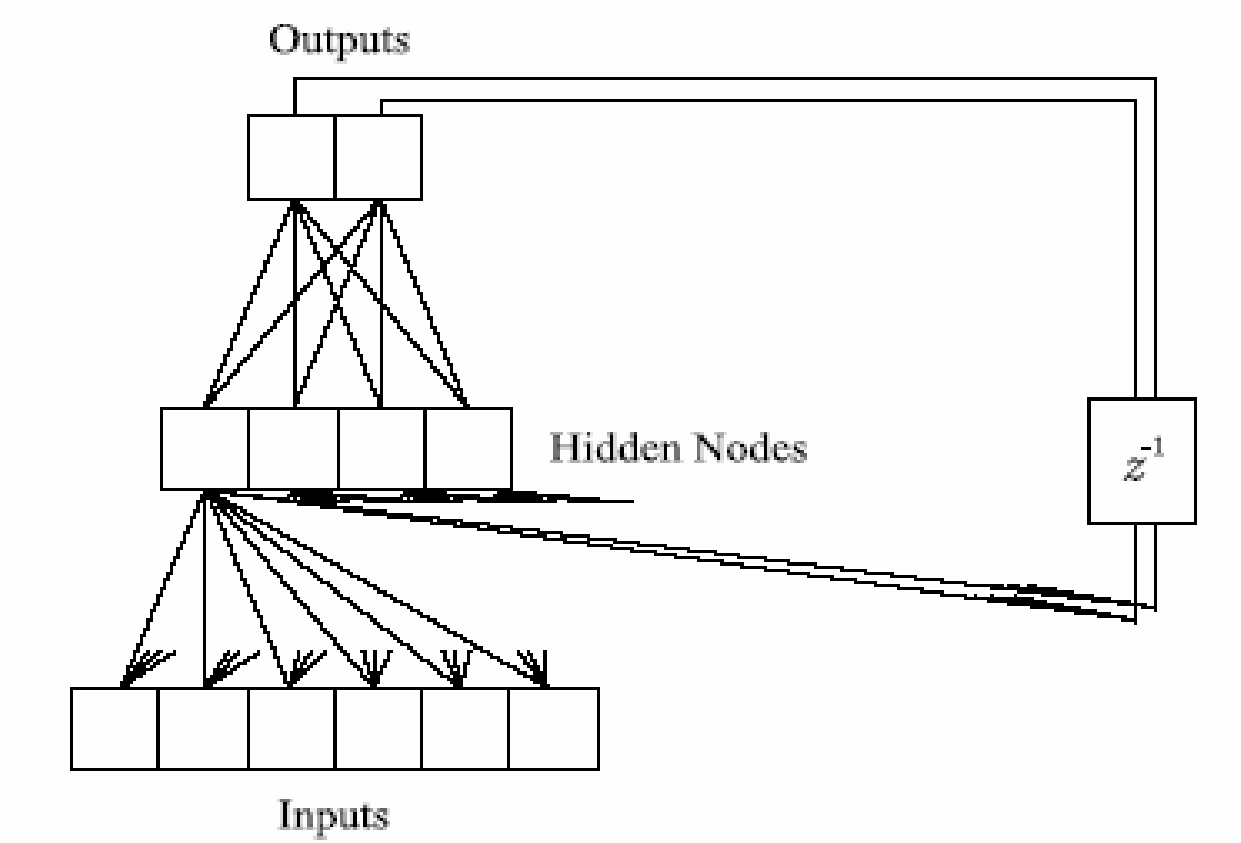
Different following RNNs was compared: `

1. *Frasconi-Gori-Soda locally recurrent networks [9].* A multi-layer perception augmented with local feedback around each hidden node. We have used the local-output version where the output of a node, $y(t) = f(wy(t-1) + \sum_{i=0}^{n} w_i x_i)$.
2. *Narendra and Parthasarathy [19].* A recurrent network with feedback connections from each output node to all hidden nodes.
3. *Elman [8].* A recurrent network with feedback from each hidden node to all hidden nodes.
4. *Williams and Zipser [29].* A recurrent network where all nodes are connected to all other nodes.

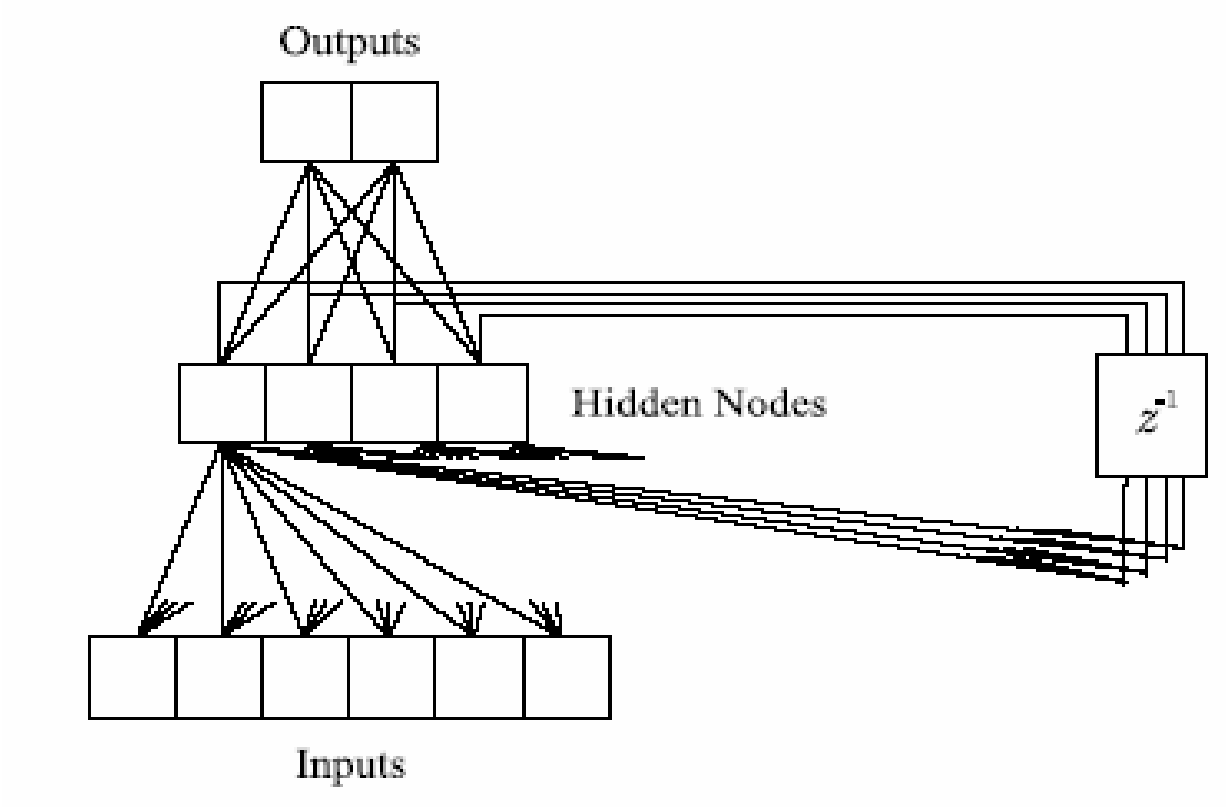
A Frasconi-Gori-Soda locally recurrent network. Not all connections are shown fully



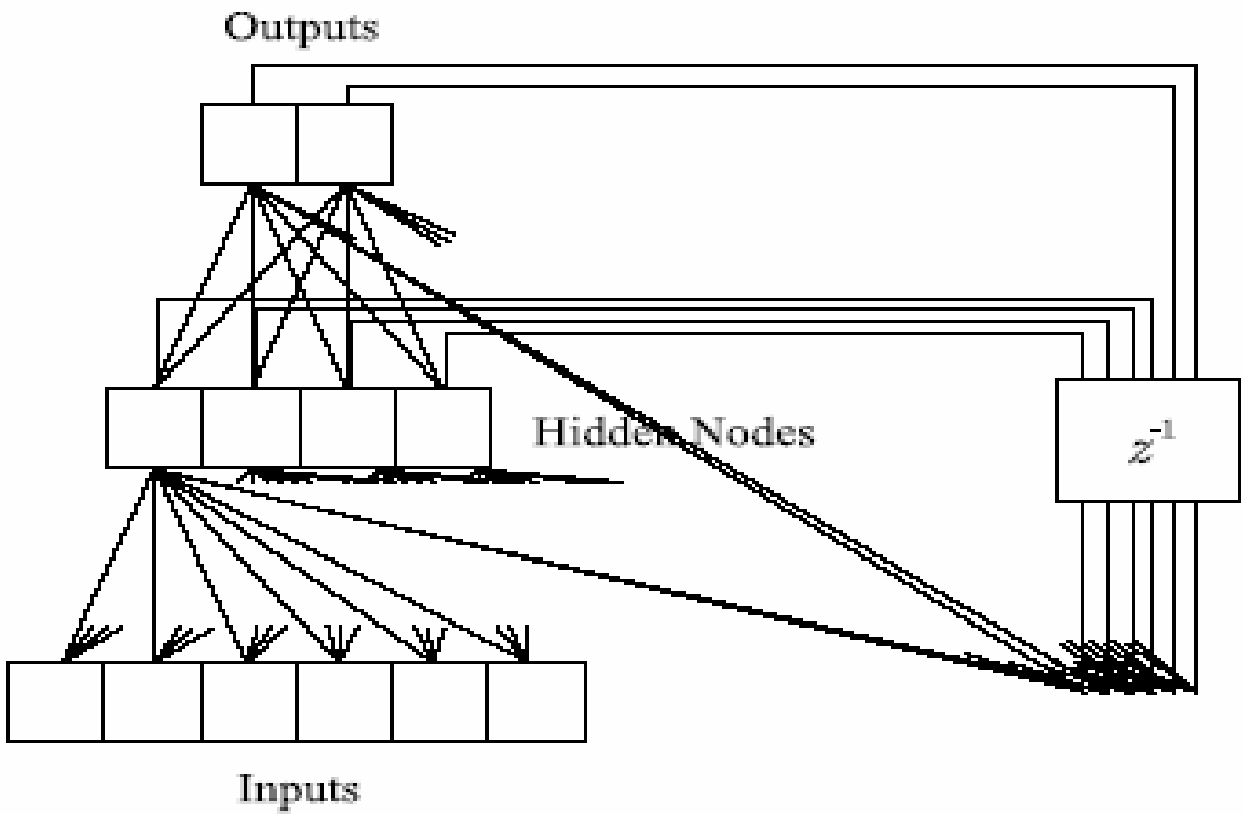
A Narendra & Parthasarathy recurrent network. Not all connections are shown fully



An Elman recurrent network. Not all connections are shown fully



A Williams & Zipser fully recurrent network. Not all connections are shown fully



Parts of speech used in experiments

Category	Examples
Nouns (N)	*John*, *book* and *destruction*
Verbs (V)	*hit*, *be* and *sleep*
Adjectives (A)	*eager*, *old* and *happy*
Prepositions (P)	*without* and *from*
Complementizer (C)	*that* or *for* as in *I thought* that . . . or *I am eager* for . . .
Determiner (D)	*the* or *each* as in the *man* or each *man*
Adverb (Adv)	*sincerely* or *why* as in *I* sincerely *believe* . . . or Why *did John want* . . .
Marker (Mrkr)	possessive *'s*, *of*, or *to* as in *John*'s *mother*, *the destruction* of . . . , or *I want* to *help* . . .

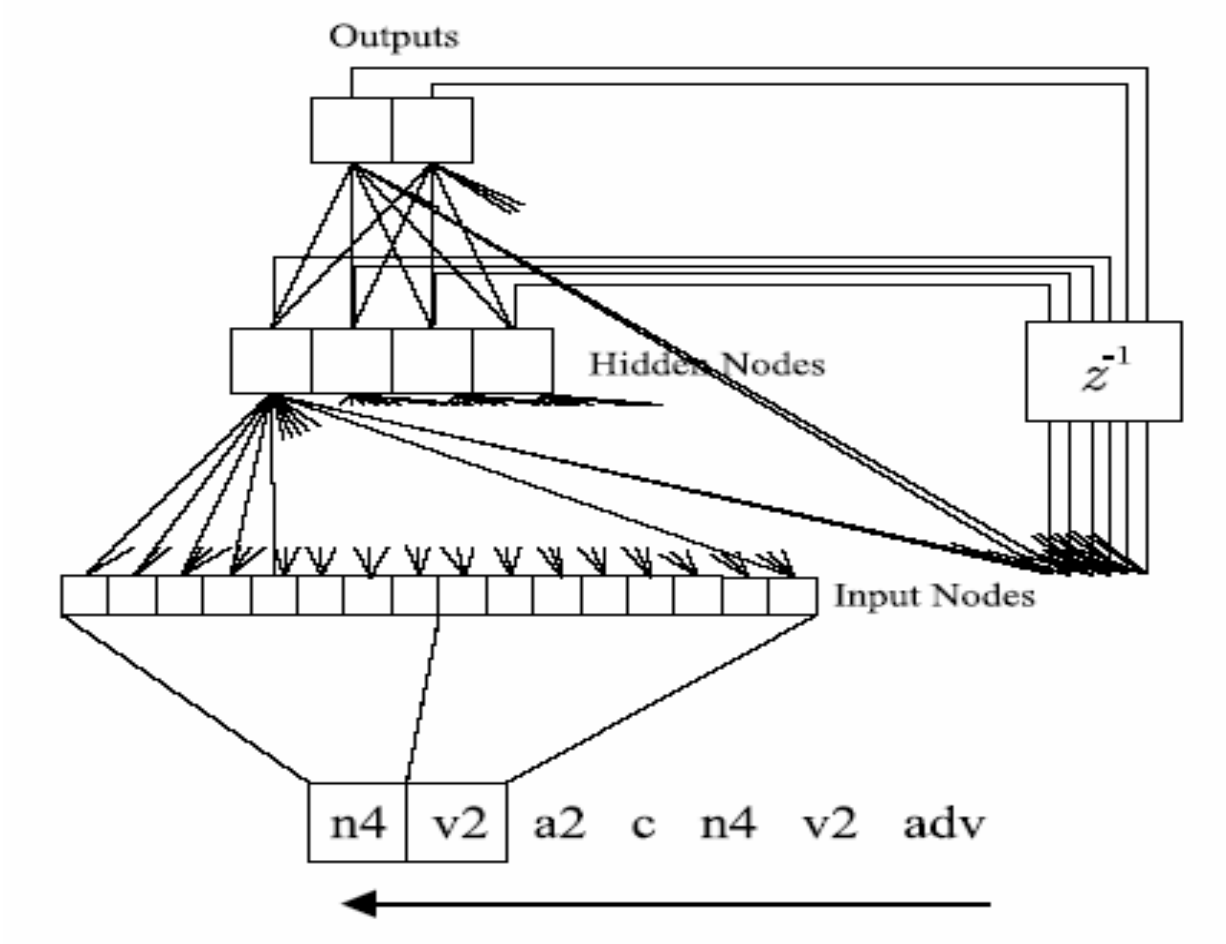
A simple grammar

S	→	NP VP "."
NP	→	PropN | N | N RC
VP	→	V (NP)
RC	→	who NP VP | who VP (NP)
N	→	boy | girl | cat...
PropN	→	John | Mary
V	→	chase | feed | see...

Examples of part-of-speech tagging

Sentence	Encoding	Grammatical Status
I am eager for John to be here	n4 v2 a2 c n4 v2 adv	1
	n4 v2 a2 c n4 p1 v2 adv	1
I am eager John to be here	n4 v2 a2 n4 v2 adv	0
	n4 v2 a2 n4 p1 v2 adv	0
I am eager to be here	n4 v2 a2 v2 adv	1
	n4 v2 a2 p1 v2 adv	1

Depiction of how the neural network inputs come from an input window on the sentence. The window moves from the beginning to the end of the sentence



Results of comparison

TRAIN	Classification	Std. dev.	Confidence
Elman	**99.6%**	0.84	78.8%
FGS	67.1%	1.22	17.6%
N&P	75.2%	1.41	32.2%
W&Z	**91.7%**	2.26	63.2%

ENGLISH TEST	Classification	Std. dev.	Confidence
Elman	**74.2%**	3.82	75.4%
FGS	59.0%	1.52	18.8%
N&P	60.6%	0.97	26.9%
W&Z	**71.3%**	0.75	65.1%

Std. dev. – the standart deviation value.
The confidence is the average confidence of the networks.

Results of comparison with other methods

Table 3. Percentage correct classification for the training data.

TRAIN	large window	small window
MLP	100	55
FGS	100	56
BT-FIR	100	56
Elman	100	**100**
W&Z	100	**92**

Table 4. Percentage correct classification for the test data.

TEST	large window	small window
Edit-distance	55	N/A
Euclidean	65	55
Decision trees	60	N/A
MLP	63	54
FGS	65	59
BT-FIR	64	54
Elman	65	**74**
W&Z	59	**71**

Nearest-neighbors

Conclusions about applications of RNN for learning of NL

- Nearest-neighbors, decision trees, feedforward networks do not learn parsimonious representations of the grammar – they work by finding statistically close matches in the training data. They are expected to require a much larger amount of data for similar performance
- RNN may be learned an appropriate grammar for discriminating between sharply grammatical/ungrammatical pairs. 100% correct classification of the training data is not possible using only a small temporal input window without forming internal states
- Generalization is limited by the amount of data available