Natural Language Processing

Pushpak Bhattacharyya
CSE Dept,
IIT Patna and Bombay

Recurrent Neural Network

NLP-ML marriage

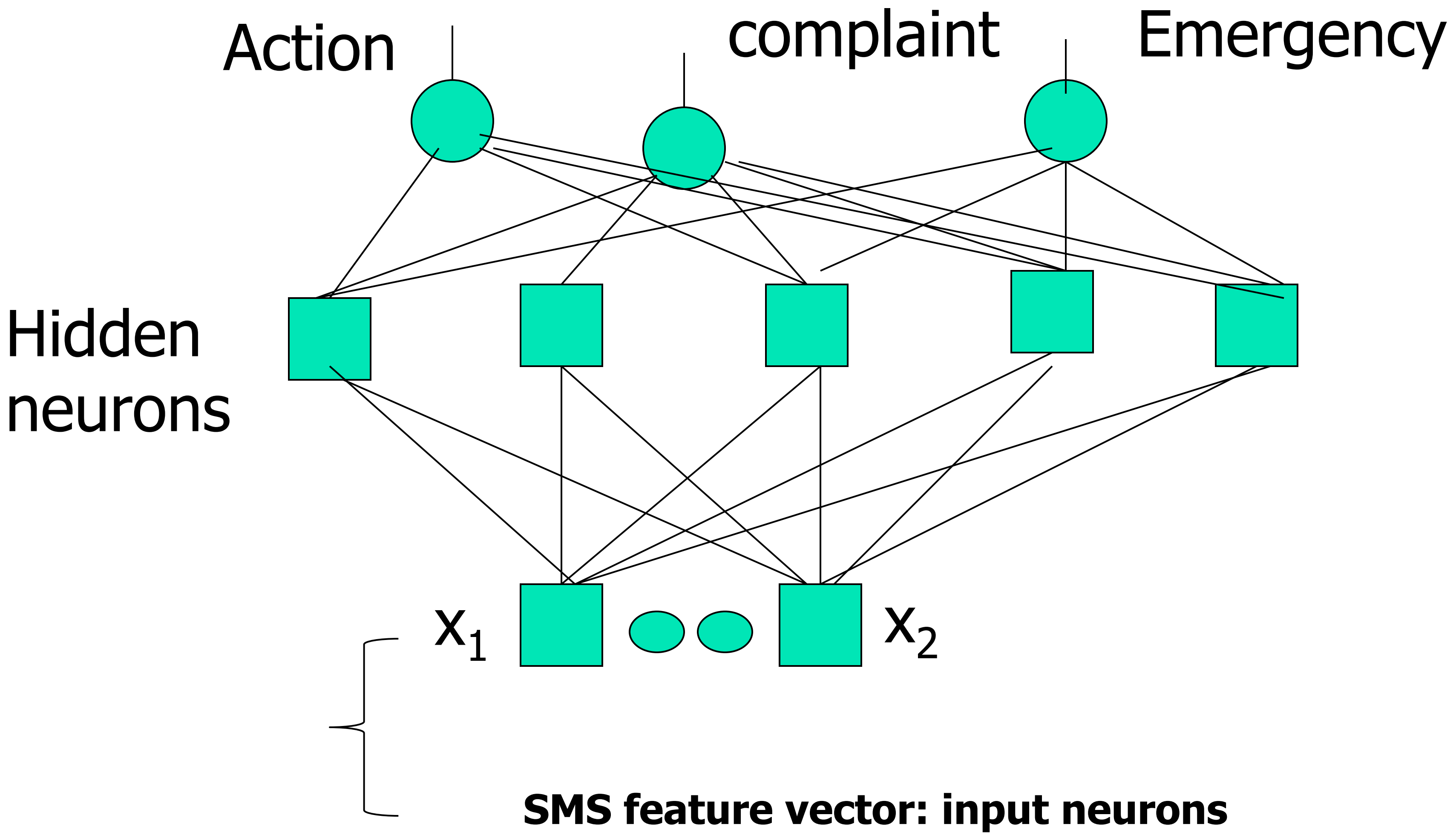
An example SMS complaint

- I have purchased a 80 litre Videocon fridge about 4 months ago when the freeze go to sleep that time compressor give a sound (khat khat khat khat ....) what is possible fault over it is normal I can't understand please help me give me a suitable answer.

Significant words (in red): after stop word removal

- I have purchased a 80 litre Videocon fridge about 4 months ago when the freeze go to sleep that time compressor give a sound (khat khat khat khat ....) what is possible fault over it is normal I can't understand please help me give me a suitable answer.

SMS classification



Feedforward Network and Backpropagation

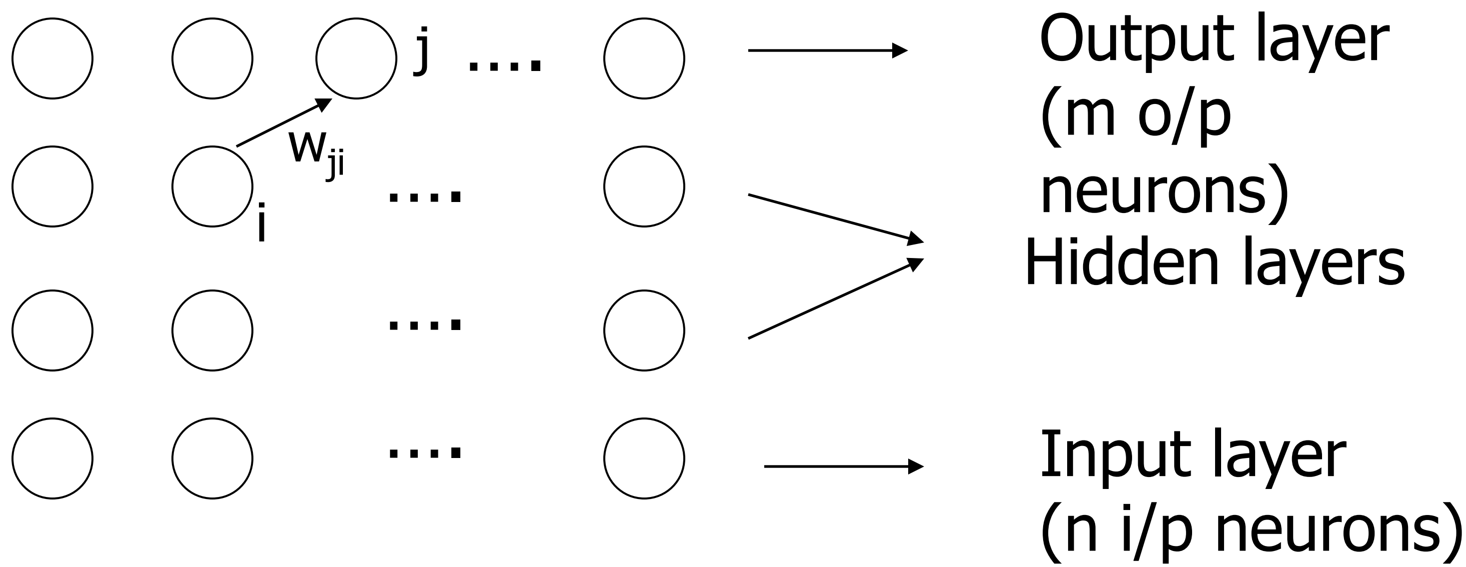
Gradient Descent Technique

- Let E be the error at the output layer

$$E = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{n}(t_i - o_i)_j^2$$

- t_i = target output; o_i = observed output
- i is the index going over n neurons in the outermost layer
- j is the index going over the p patterns (1 to p)
- Ex: XOR:– p=4 and n=1

Backpropagation algorithm



- Fully connected feed forward network
- Pure FF network (no jumping of connections over layers)

General Backpropagation Rule

- General weight updating rule:

$$\Delta w_{ji} = \eta \delta j o_i$$

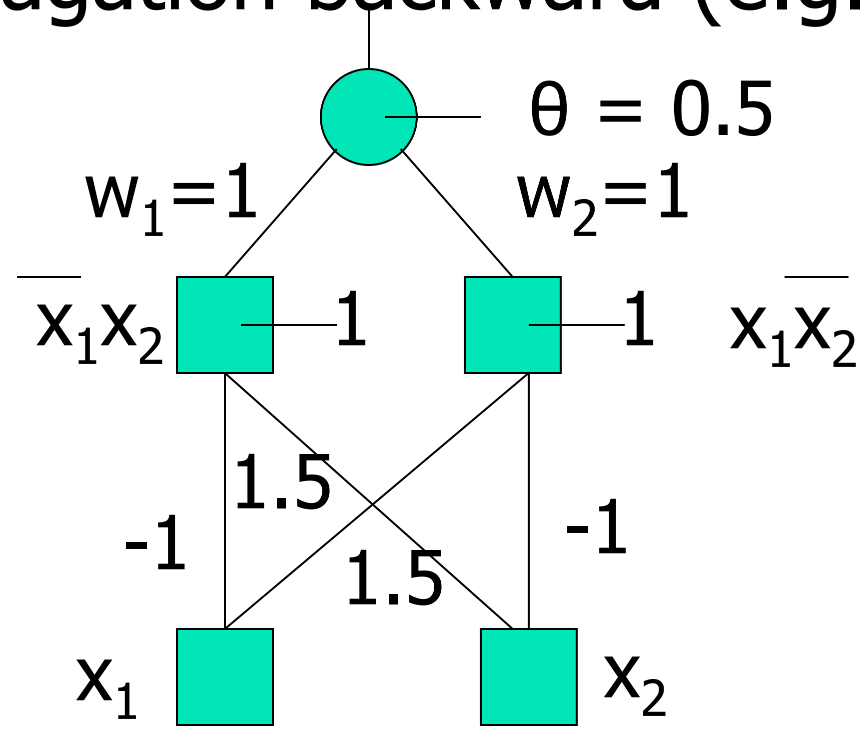
- Where

$$\delta_j = (t_j - o_j) o_j (1 - o_j) \quad \text{for outermost layer}$$

$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j) o_i \ \text{for hidden layers}$$

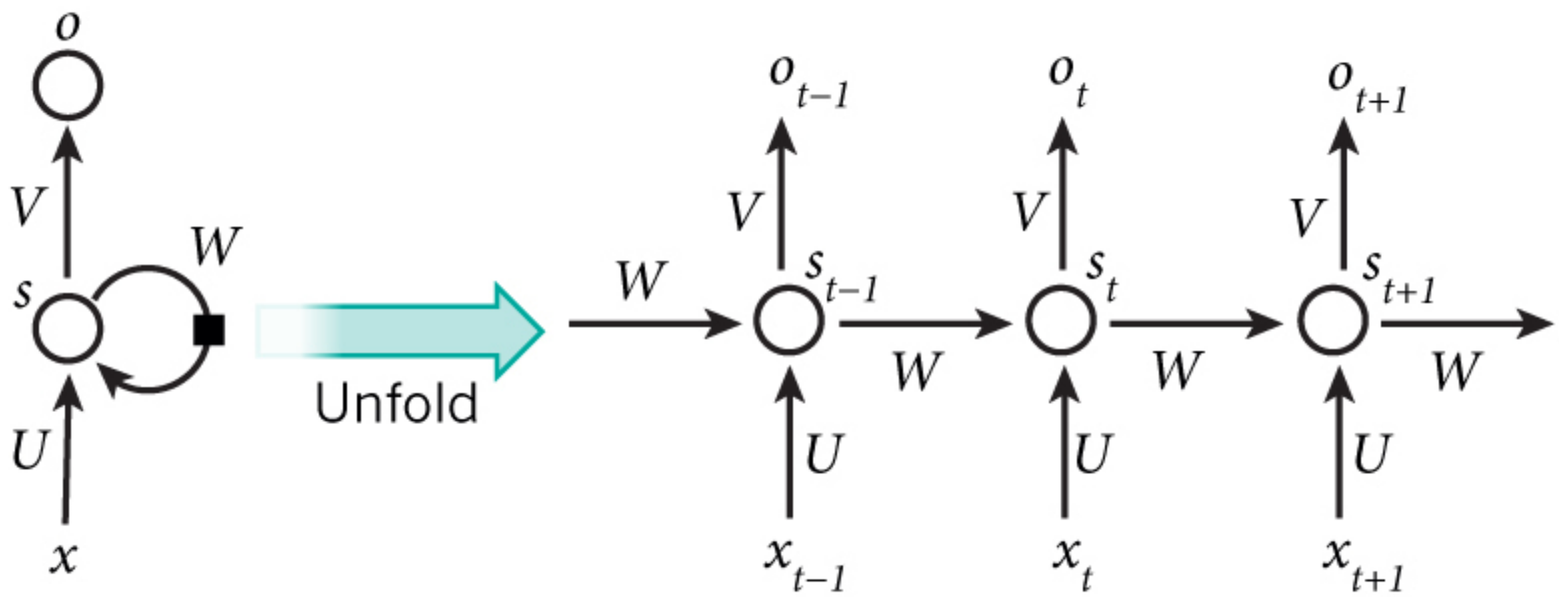
How does it work?

- Input propagation forward and error propagation backward (e.g. XOR)

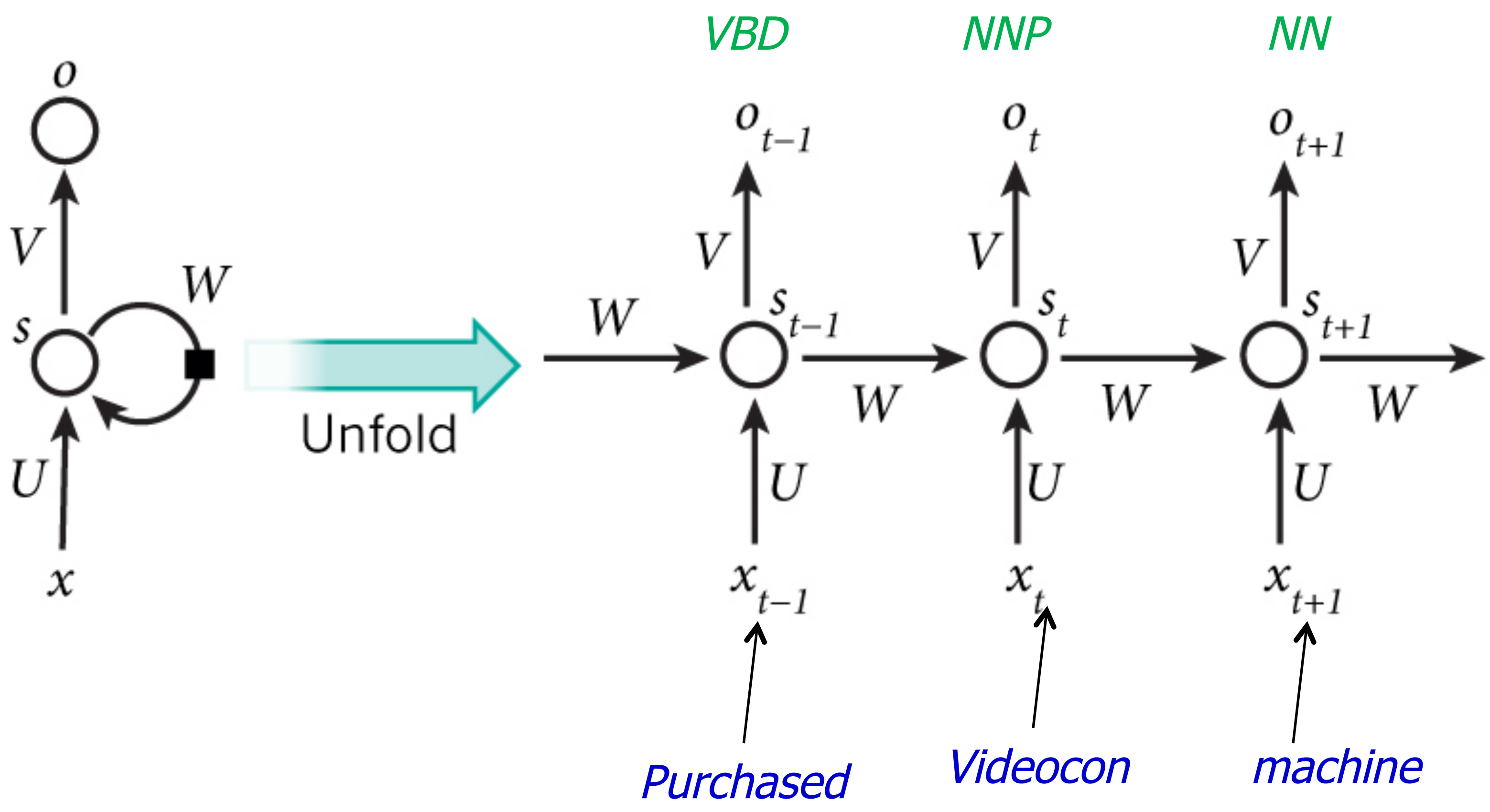


Recurrent Neural Network (1 min recap)

Sequence processing m/c

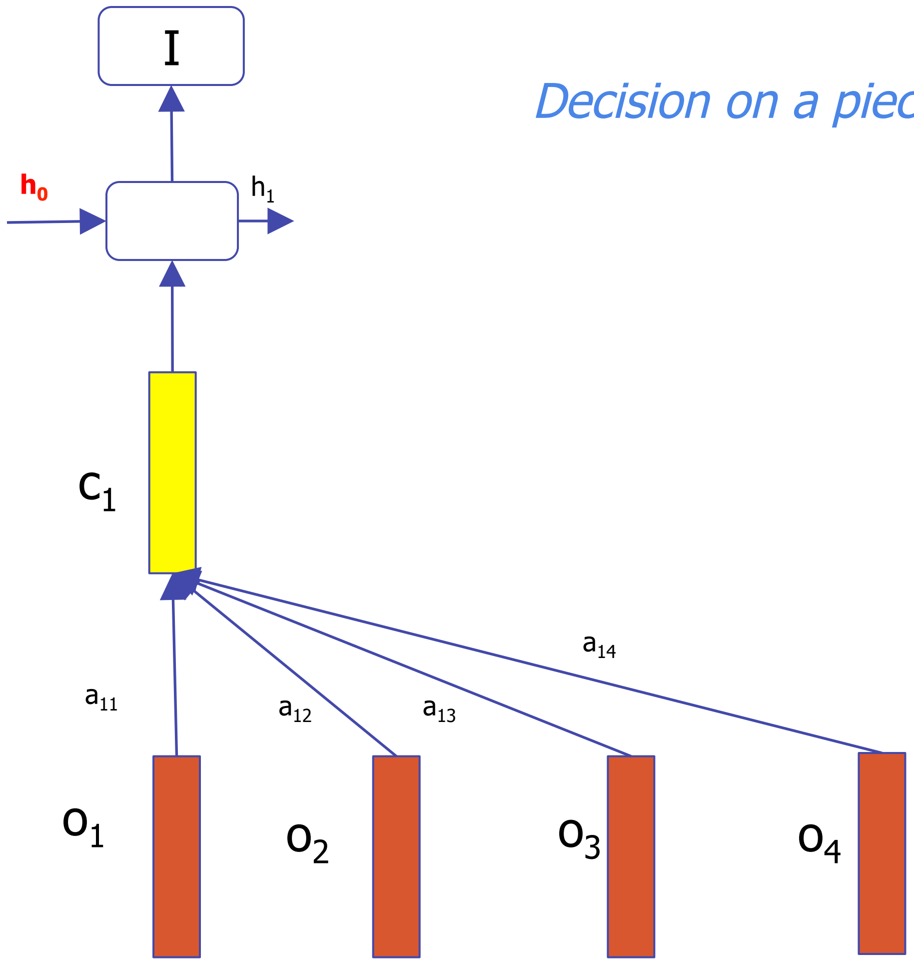


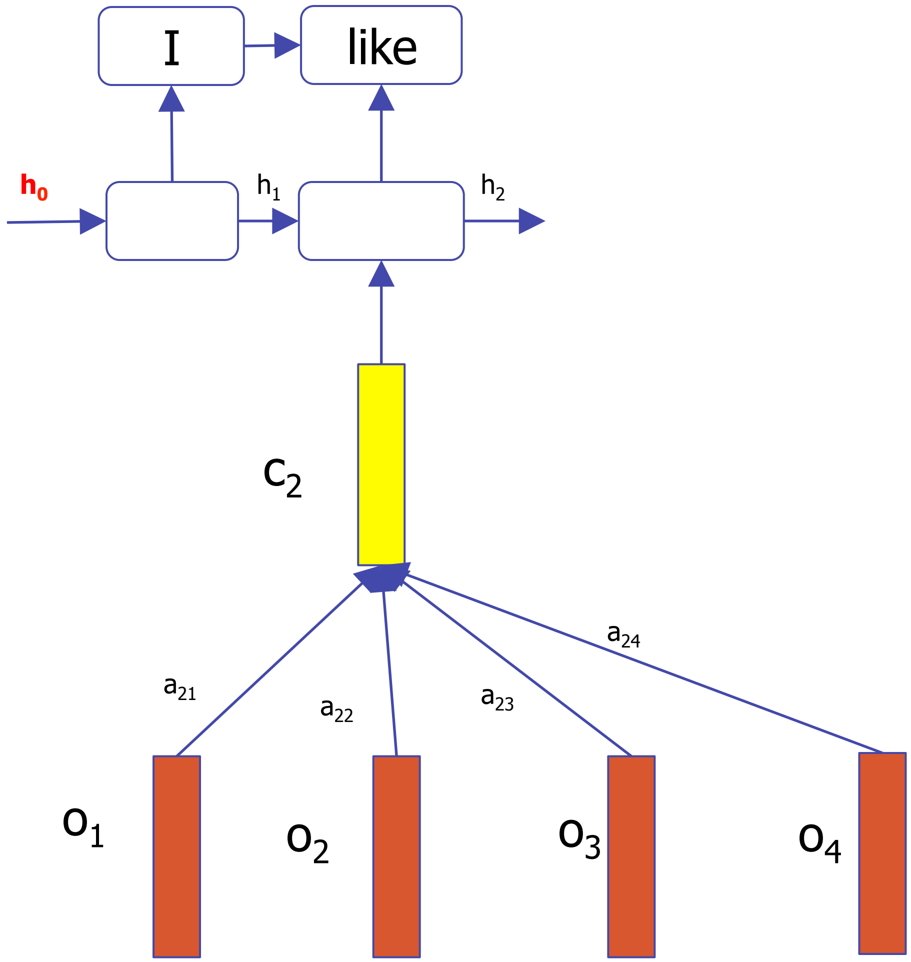
E.g. POS Tagging

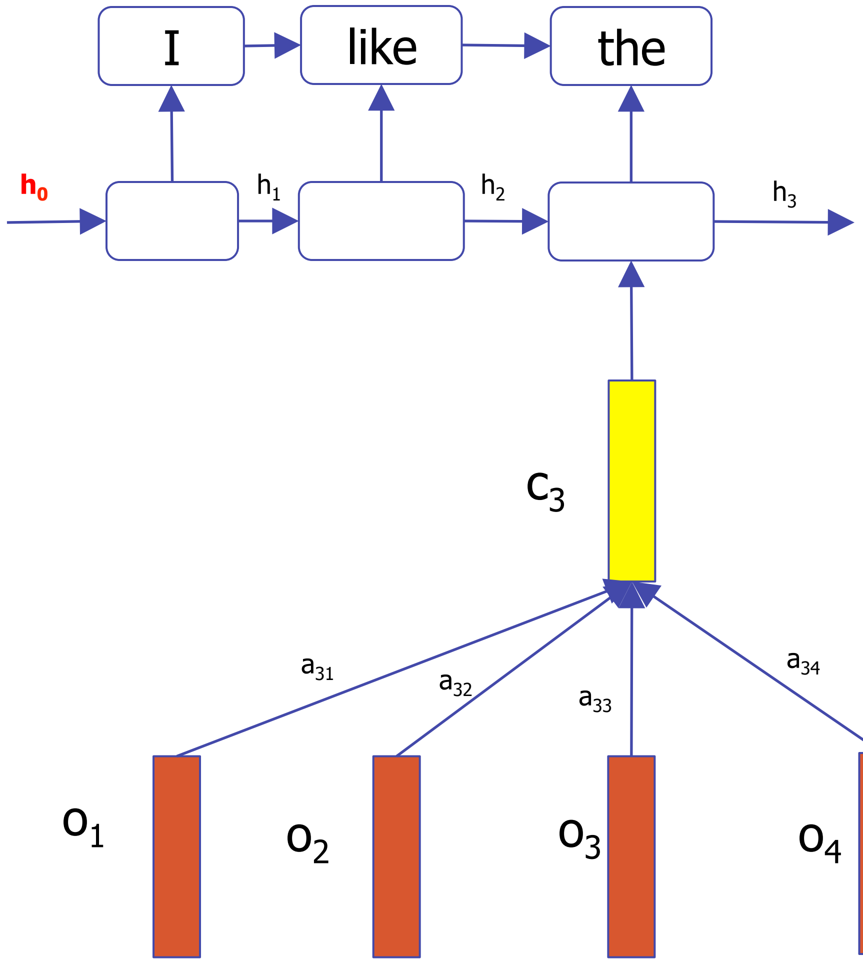


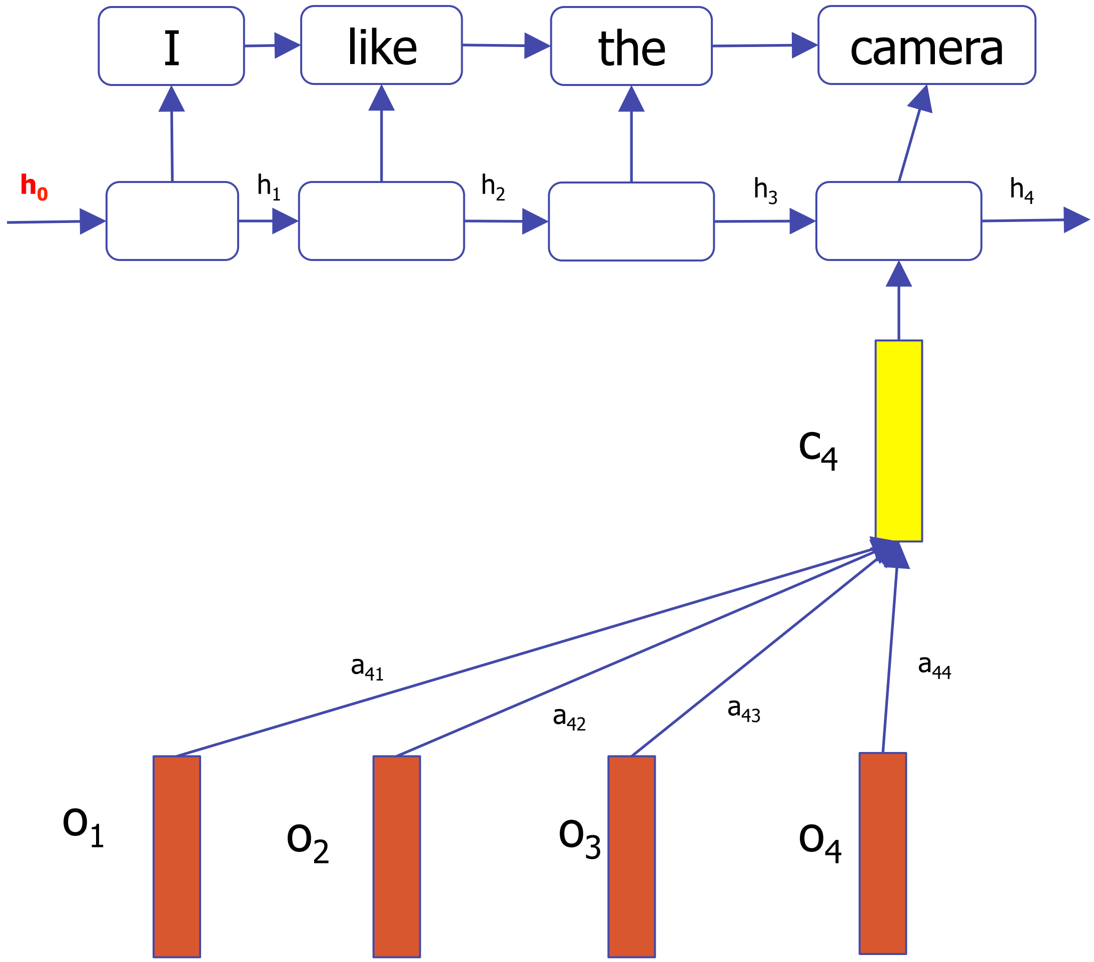
E.g. Sentiment Analysis

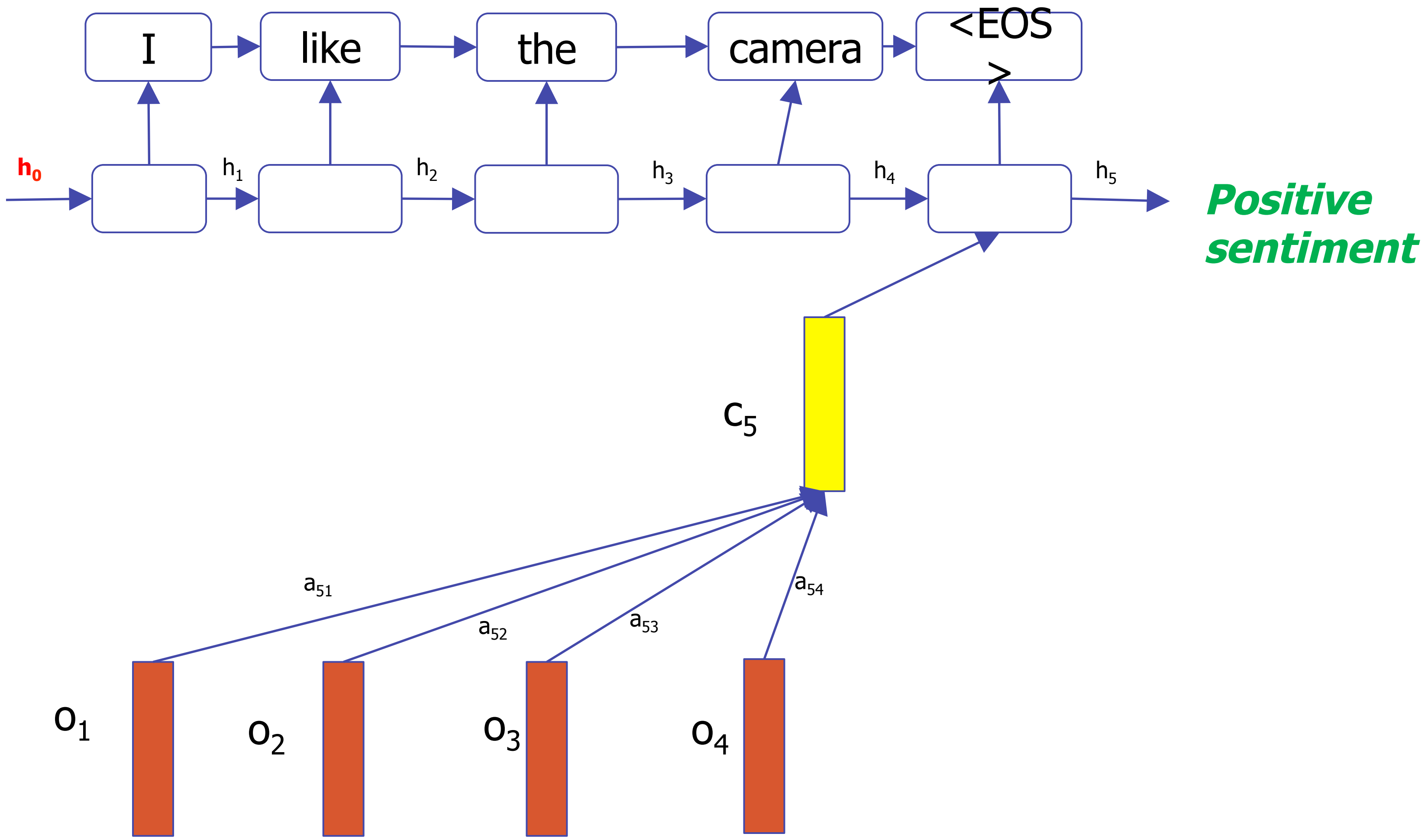
Decision on a piece of text




I
like
h0
h1
h2
c2
a21
a22
a23
a24
o1
o2
o3
o4


I
like
the
h_0
h_1
h_2
h_3
c_3
a_{31}
a_{32}
a_{33}
a_{34}
o_1
o_2
o_3
o_4


I
like
the
camera
h0
h1
h2
h3
h4
c4
a41
a42
a43
a44
o1
o2
o3
o4


I
like
the
camera
<EOS
>
h_0
h_1
h_2
h_3
h_4
h_5
Positive sentiment
c_5
a_{51}
a_{52}
a_{53}
a_{54}
o_1
o_2
o_3
o_4

Most famous “Ancient” RNN

Hopfield Net

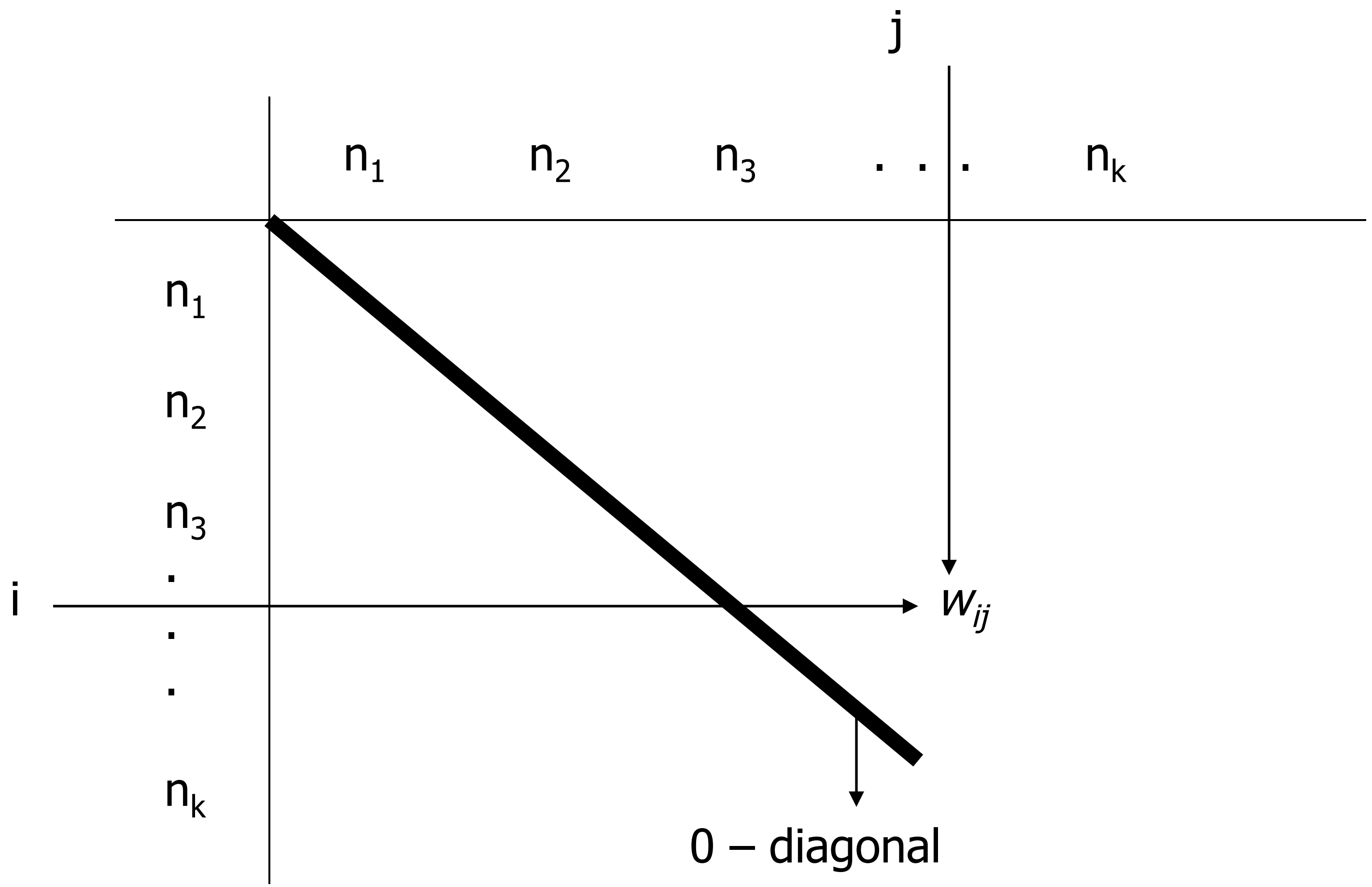
Hopfield net

- Inspired by associative memory which means memory retrieval is not by address, but by part of the data; Same idea can be used for **sentence completion**
- Consists of

  N neurons fully connected with

  symmetric weight strength $w_{ij} = w_{ji}$
- No self connection. So the weight matrix is 0-diagonal and symmetric.
- Each computing element or neuron is a linear threshold element with threshold = 0.

Connection matrix of the network, 0-diagonal and symmetric



Example

$w_{12} = w_{21} = 5$

$w_{13} = w_{31} = 3$

$w_{23} = w_{32} = 2$

<u>At time *t=0*</u>

$s_1(t) = 1$

$s_2(t) = -1$

$s_3(t) = 1$

Unstable state: Neuron 1 will flip.

A stable pattern is called an attractor for the net.

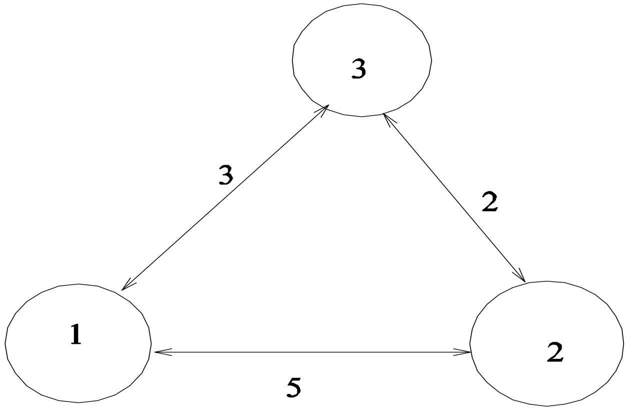
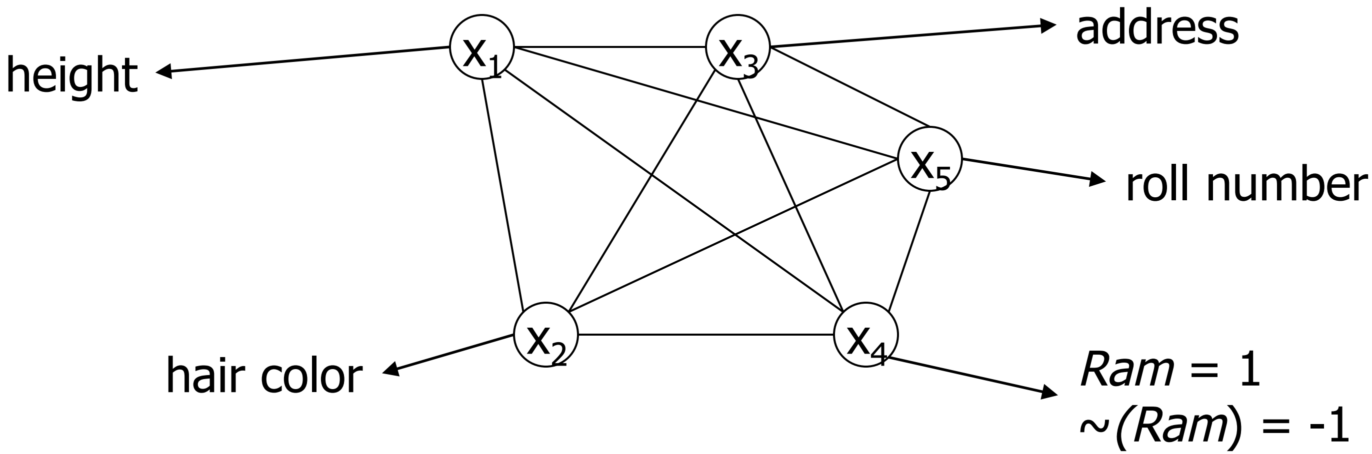


Figure: An example Hopfield Net

State Vector

- Binary valued vector: value is either 1 or -1

  $X = \langle x_n \, x_{n-1} \quad . \quad . \quad . \quad . \quad x_3 \quad x_2 \quad x_1 \rangle$
- *e.g.* Various attributes of a student can be represented by a state vector

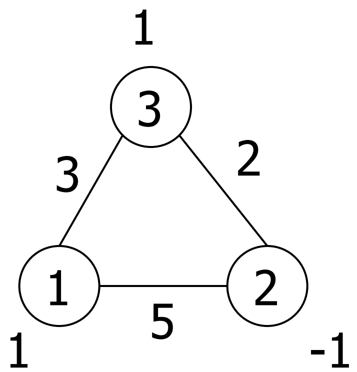


Concept of Energy

- Energy at state *s* is given by the equation:

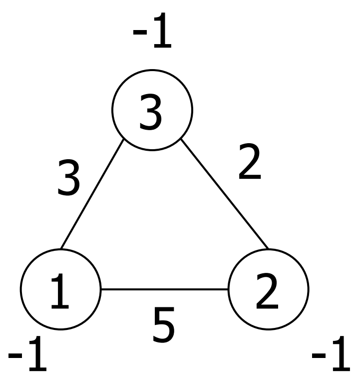
$$
\begin{aligned}
E(s) = -\Big[& w_{12}x_1x_2 + w_{13}x_1x_3 + \ldots + w_{1n}x_1x_n \\
& + w_{23}x_2x_3 + \ldots + w_{2n}x_2x_n + \\
& \vdots \\
& + w_{(n-1)n}x_{(n-1)}x_n \Big]
\end{aligned}
$$

Energy Consideration



At time $t = 0$, state of the neural network is:
$s(0) = <1, -1, 1>$

- E(0) = -[(5*1*-1)+(3*1* 1)+(2*-1*1)] = 4



The state of the neural network under stability is $<-1, -1, -1>$

E(stable state) = - -[(5*-1*-1)+(3*-1* -1)+(2*-1*-1)] = -10

The Hopfield net has to "converge" in the asynchronous mode of operation

- As the energy *E* goes on decreasing, it has to hit the bottom, since the weight and the state vector have finite values.
- That is, the Hopfield Net has to converge to an energy minimum.
- Hence the Hopfield Net reaches stability.

Hopfield Net: an *o(n²) algorithm*

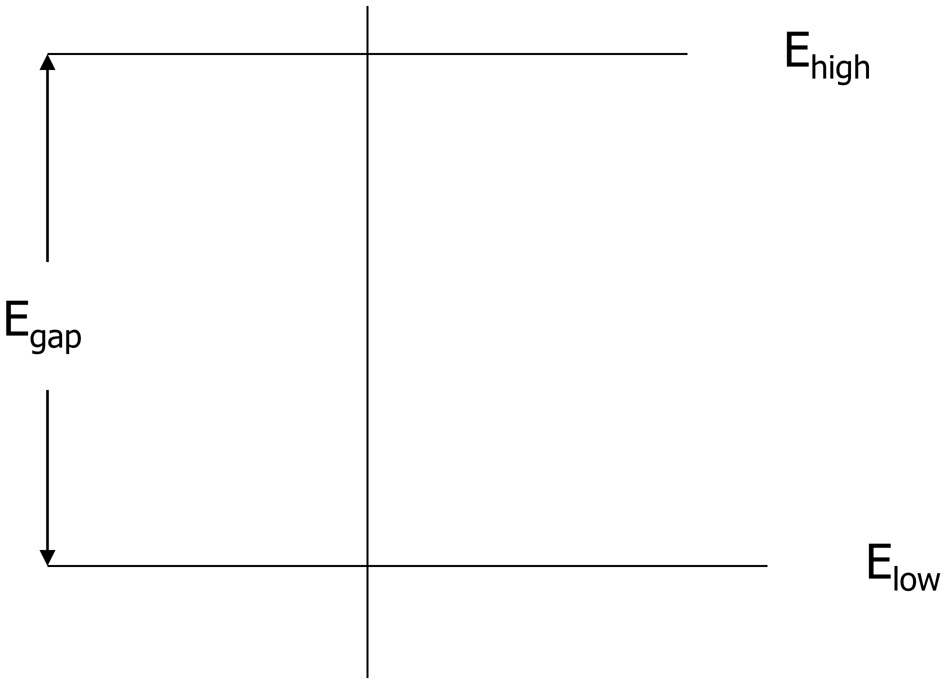
- Consider the energy expression

$$E = -[w_{12}x_1x_2 + w_{13}x_1x_3 + \dots + w_{1n}x_1x_n$$
$$+w_{23}x_2x_3 + w_{24}x_2x_4 + \dots + w_{2n}x_2x_n$$
$$\vdots$$
$$+w_{(n-1)n}x_{(n-1)}x_n]$$

- *E* has *[n(n-1)]/2* terms
- Nature of each term
  - w_{ij} is a real number
  - x_i and x_j are each +1 or -1

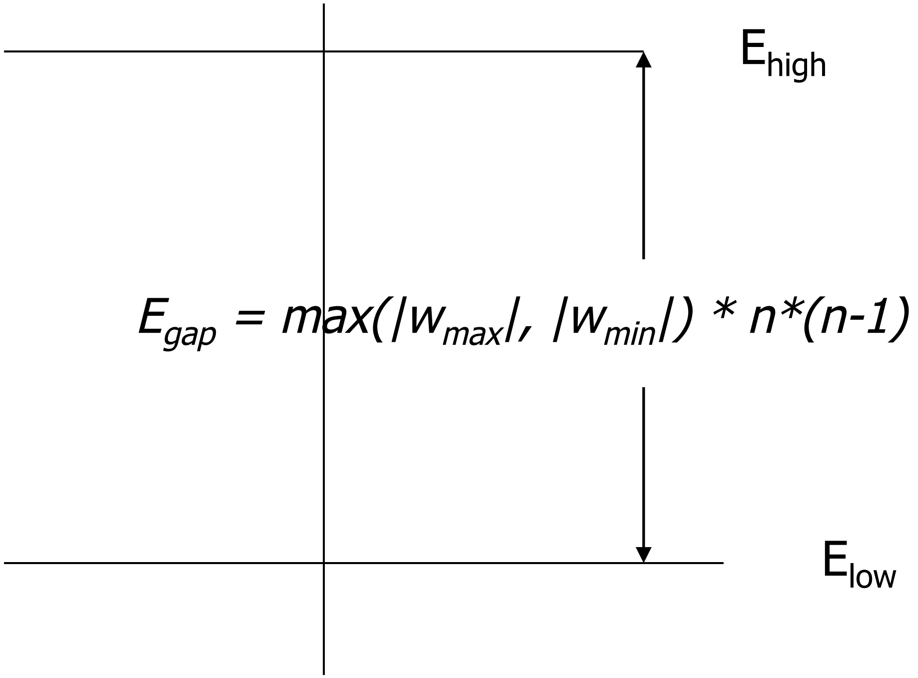
No. of steps taken to reach stability

- $E_{gap} = E_{high} - E_{low}$



The energy gap

- In general,



$$E_{gap} = max(|w_{max}|, |w_{min}|) * n*(n-1)$$

#steps to stable state: $o(n^2)$

- It is <u>possible</u> to reach the minimum <u>independent</u> of *n*.
- Hence in the worst case, the number of steps taken to cover the energy gap is less than or equal to

  $[max(|w_{max}|,|w_{min}|) * n * (n-1)] / constant$
- Thus stability has to be attained in $O(n^2)$ steps

Training of Hopfield Net

- Early Training Rule proposed by Hopfield
- Rule inspired by the concept of electron spin
- Hebb's rule of learning
  - If two neurons *i* and *j* have activation x_i and x_j respectively, then the weight w_{ij} between the two neurons is directly proportional to the product $x_i \cdot x_j$ *i.e.*

$$w_{ij} \propto x_i \cdot x_j$$

Hopfield Rule

- Training by Hopfield Rule
  - Train the Hopfield net for a specific memory behavior
  - Store memory elements
  - *How to store patterns?*

Hopfield Rule

- To store a pattern

  $<x_n, x_{n-1}, \ldots, x_3, x_2, x_1>$

  make

$$w_{ij} = \frac{1}{(n-1)} \cdot x_i \cdot x_j$$

- Storing pattern is equivalent to '*Making that pattern the stable state*'

Hopfield Net for Optimization

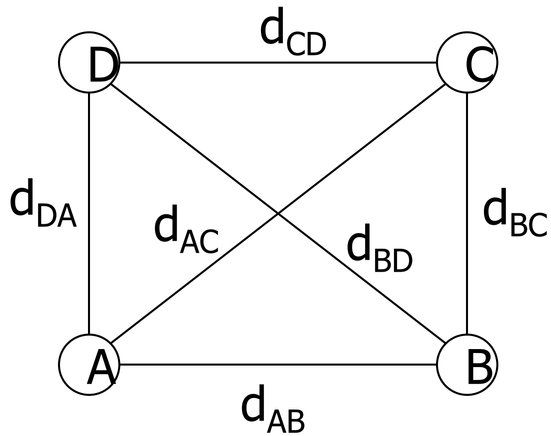
- Optimization problem
  - Maximizes or minimizes a quantity
- Hopfield net used for optimization
  - Hopfield net and Traveling Salesman Problem
  - Hopfield net and Job Scheduling Problem

The essential idea of the correspondence

- In optimization problems, we have to *minimize* a quantity.
- Hopfield net minimizes the energy
- THIS IS THE CORRESPONDENCE

Hopfield net and Traveling Salesman problem

- We consider the problem for *n=4* cities
- In the given figure, nodes represent cities and edges represent the paths between the cities with associated distance.



Traveling Salesman Problem

- Goal
  - Come back to the city A, visiting $j = 2$ to n (n is number of cities) exactly once and minimize the total distance.
- To solve by Hopfield net we need to decide the *architecture*:
  - How many neurons?
  - What are the weights?

Constraints decide the *parameters*

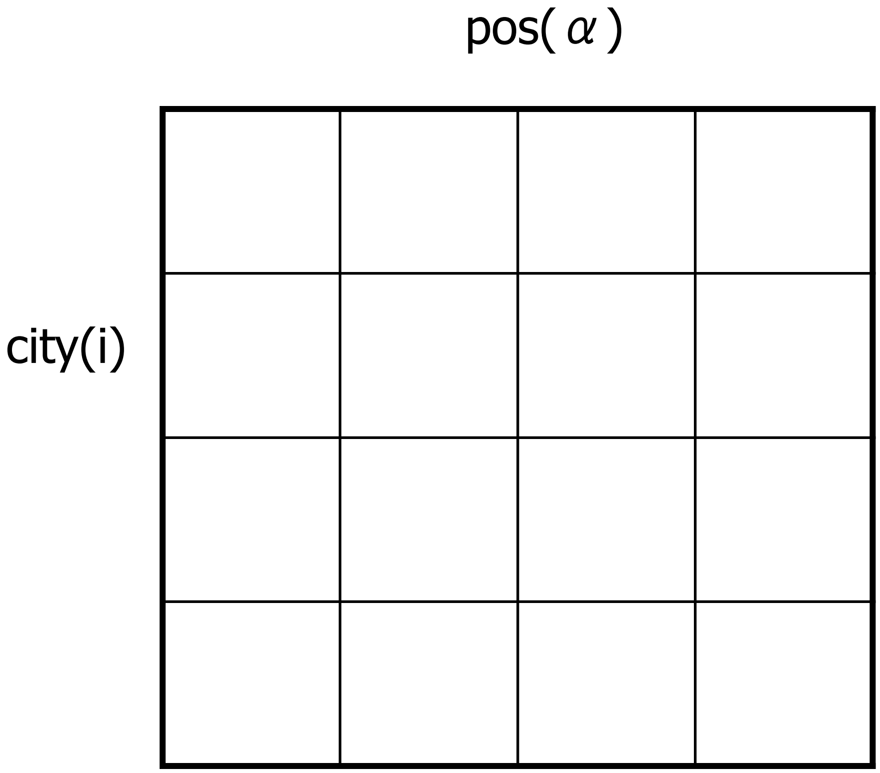
1. For *n* cities and *n* positions, establish city to position correspondence, *i.e.*

   Number of neurons = *n* cities * *n* positions

2. Each position can take <u>one and only one</u> city
3. Each city can be in exactly one position
4. Total distance should be minimum

Architecture

- $n * n$ matrix where rows denote cities and columns denote positions
- $cell(i, j) = 1$ if and only if i^{th} city is in j^{th} position
- Each cell is a neuron
- n^2 neurons, $O(n^4)$ connections



Expressions corresponding to constraints

- We equate constraint energy:

$$E_{Problem} = E_{network} \qquad (*)$$

Where, $E_{problem} = E_1 + E_2 + E_3 + E_4$

and $E_{network}$ *is the well known energy expression for the Hopfield net*

- Find the weights from (*).

Finding weights for Hopfield Net applied to TSP

- $E_{problem} = E_1 + E_2$

where

E_1 is the equation for n cities, each city in one position and each position with one city.

E_2 is the equation for distance

Expressions for E_1 and E_2

$$E_1 = \frac{A}{2}[\sum_{\alpha=1}^{n}(\sum_{i=1}^{n} x_{i\alpha} - 1)^2 + \sum_{i=1}^{n}(\sum_{\alpha=1}^{n} x_{i\alpha} - 1)^2]$$

$$E_2 = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\alpha=1}^{n} d_{ij} \cdot x_{i\alpha} \cdot (x_{j,\alpha+1} + x_{j,\alpha-1})$$

Explanatory example

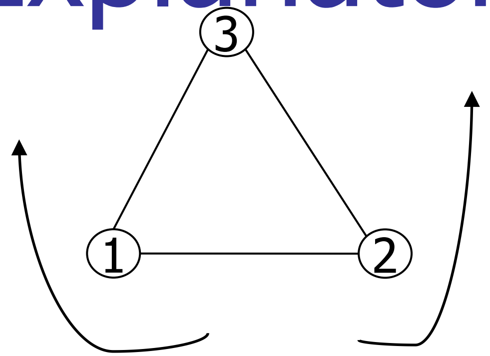


Fig. 1

Fig. 1 shows two possible directions in which tour can take place

pos →

city ↓

	1	2	3
1	x_{11}	x_{12}	x_{13}
2	x_{21}	x_{22}	x_{23}
3	x_{31}	x_{32}	x_{33}

For the matrix alongside, $x_{i\alpha} = 1$, if and only if, i^{th} city is in position α

Kinds of weights

- Row weights:

$w_{11,12}$	$w_{11,13}$	$w_{12,13}$
$w_{21,22}$	$w_{21,23}$	$w_{22,23}$
$w_{31,32}$	$w_{31,33}$	$w_{32,33}$

- Column weights

$w_{11,21}$	$w_{11,31}$	$w_{21,31}$
$w_{12,22}$	$w_{12,32}$	$w_{22,32}$
$w_{13,23}$	$w_{13,33}$	$w_{23,33}$

Cross weights

$W_{11,22}$	$W_{11,23}$	$W_{11,32}$	$W_{11,33}$
$W_{12,21}$	$W_{12,23}$	$W_{12,31}$	$W_{12,33}$
$W_{13,21}$	$W_{13,22}$	$W_{13,31}$	$W_{13,32}$
$W_{21,32}$	$W_{21,33}$	$W_{22,31}$	$W_{23,33}$
$W_{23,31}$	$W_{23,32}$		

Expressions

$$E_{problem} = E_1 + E_2$$

$$\begin{aligned}
E_1 = \frac{A}{2}[&(x_{11} + x_{12} + x_{13} - 1)^2 \\
&+ (x_{21} + x_{22} + x_{23} - 1)^2 \\
&+ (x_{31} + x_{32} + x_{33} - 1)^2 \\
&+ (x_{11} + x_{21} + x_{31} - 1)^2 \\
&+ (x_{12} + x_{22} + x_{32} - 1)^2 \\
&+ (x_{13} + x_{23} + x_{33} - 1)^2]
\end{aligned}$$

Expressions (contd.)

$$
\begin{aligned}
E_2 = \frac{1}{2}[& d_{12}x_{11}(x_{22} + x_{23}) + \\
& d_{12}x_{12}(x_{23} + x_{21}) + \\
& d_{12}x_{13}(x_{21} + x_{22}) + \\
& d_{13}x_{11}(x_{32} + x_{33}) + \\
& d_{13}x_{12}(x_{33} + x_{31}) + \\
& d_{13}x_{13}(x_{31} + x_{32}) \ldots]
\end{aligned}
$$

$E_{network}$

$$E_{network} = -[w_{11,12}x_{11}x_{12} + w_{11,13}x_{11}x_{13} + w_{12,13}x_{12}x_{13} + w_{11,21}x_{11}x_{21} + w_{11,22}x_{11}x_{22} + w_{11,23}x_{11}x_{23} + w_{11,31}x_{11}x_{31} + w_{11,32}x_{11}x_{32} + w_{11,33}x_{11}x_{33} \ldots]$$

Find row weight

- To find, $w_{11,12}$
  = -(co-efficient of $x_{11}x_{12}$) in $E_{network}$
- Search $x_{11}x_{12}$ in $E_{problem}$

$w_{11,12}$ = -A ...from E_1. E_2 cannot contribute

Find column weight

- To find, $w_{11,21}$ = -(co-efficient of $x_{11}x_{21}$) in $E_{network}$
- Search co-efficient of $x_{11}x_{21}$ in $E_{problem}$

$w_{11,21}$ = -A ...from E_1. E_2 cannot contribute

Find Cross weights

- To find, $w_{11,22}$
  $= -(\text{co-efficient of } x_{11}x_{22})$
- Search $x_{11}x_{22}$ from $E_{problem.}$ E_1 cannot contribute
- Co-eff. of $x_{11}x_{22}$ in E_2
  $(d_{12} + d_{21}) / 2$

Therefore, $w_{11,22} = -((d_{12} + d_{21}) / 2)$

Find Cross weights

- To find, $w_{11,33}$
  $= -(\text{co-efficient of } x_{11}x_{33})$
- Search for $x_{11}x_{33}$ in $E_{problem}$
  $$w_{11,33} = -((d_{13} + d_{31}) / 2)$$

Summary

- Row weights = $-A$
- Column weights = $-A$
- Cross weights = $-((d_{ij} + d_{ji}) / 2), j = i \pm 1$

Recurrent Neural Network

Acknowledgement:

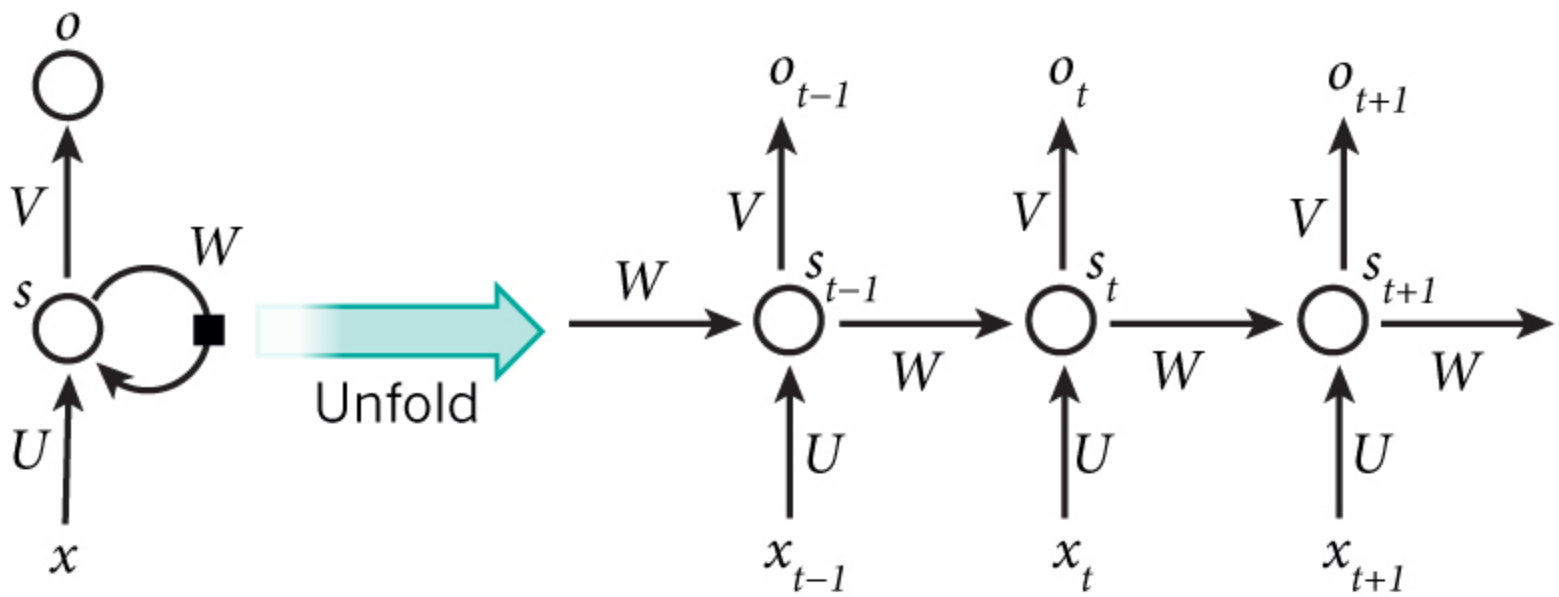
1. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

By Denny Britz

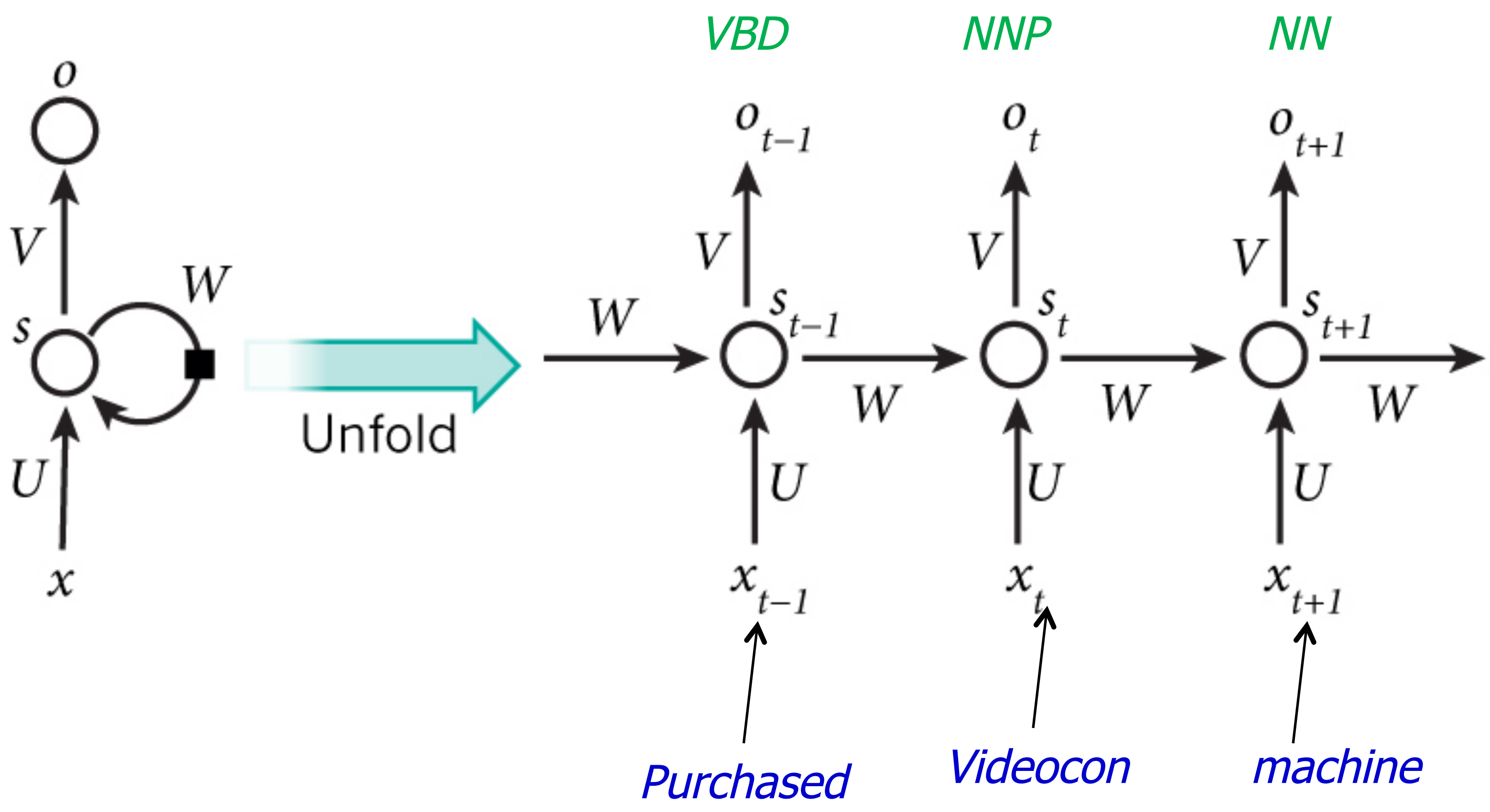
2. Introduction to RNN by Jeffrey Hinton

http://www.cs.toronto.edu/~hinton/csc2535/lectures.html

Sequence processing m/c

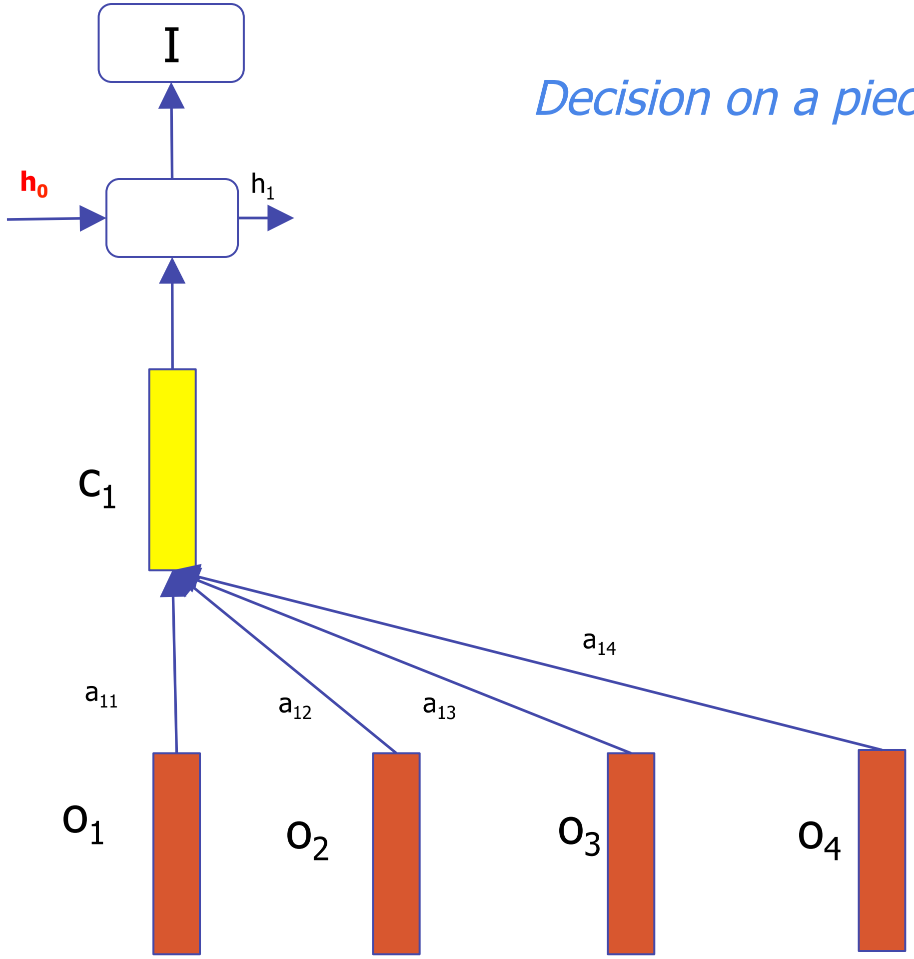


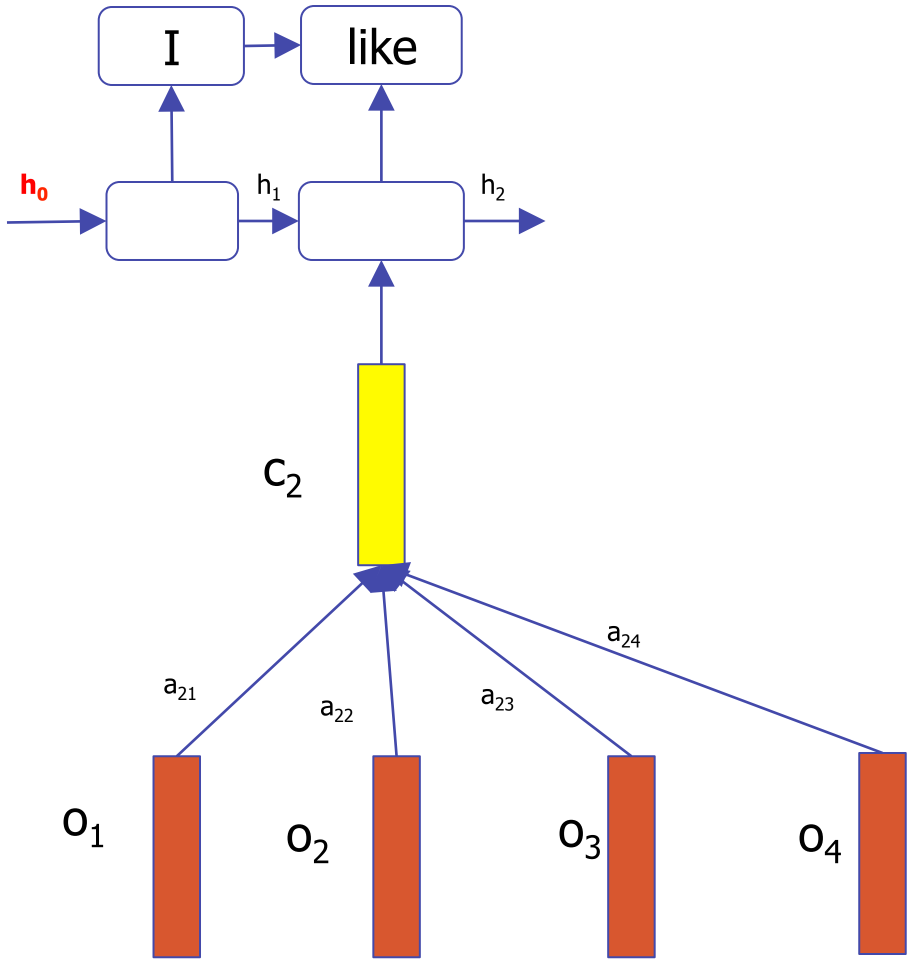
E.g. POS Tagging



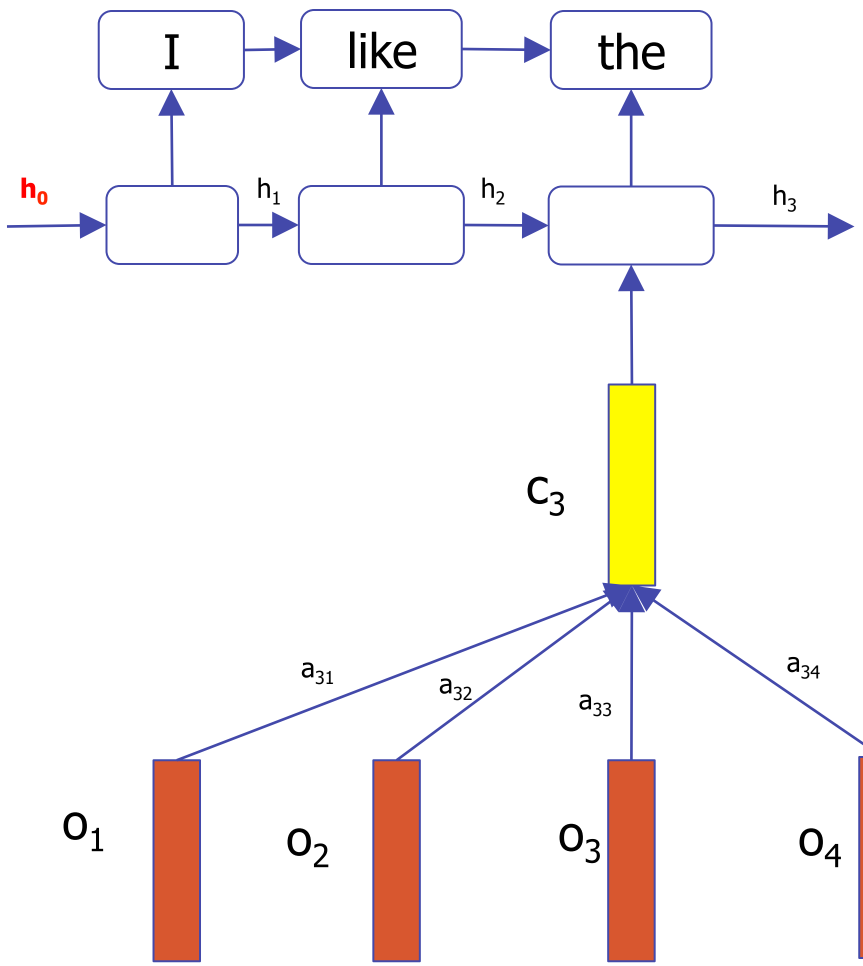
E.g. Sentiment Analysis

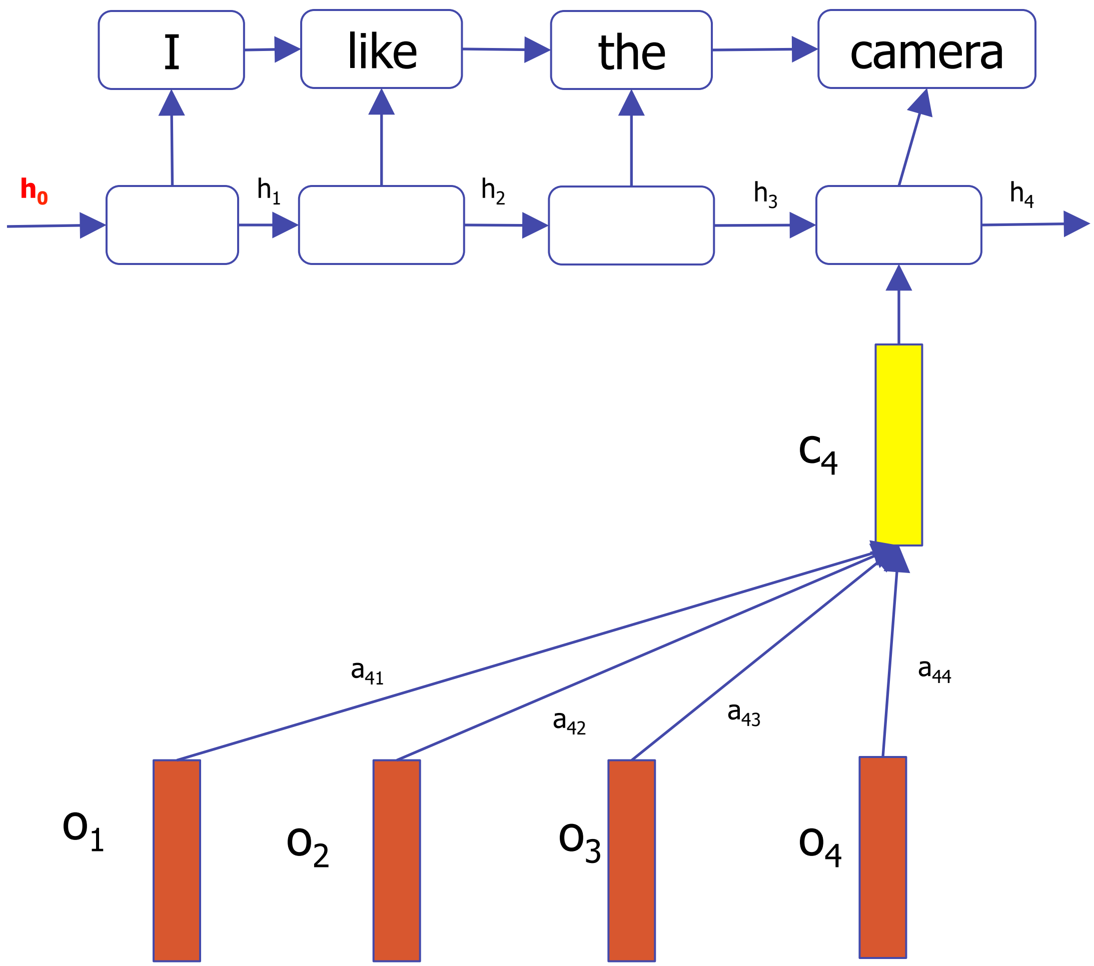
Decision on a piece of text

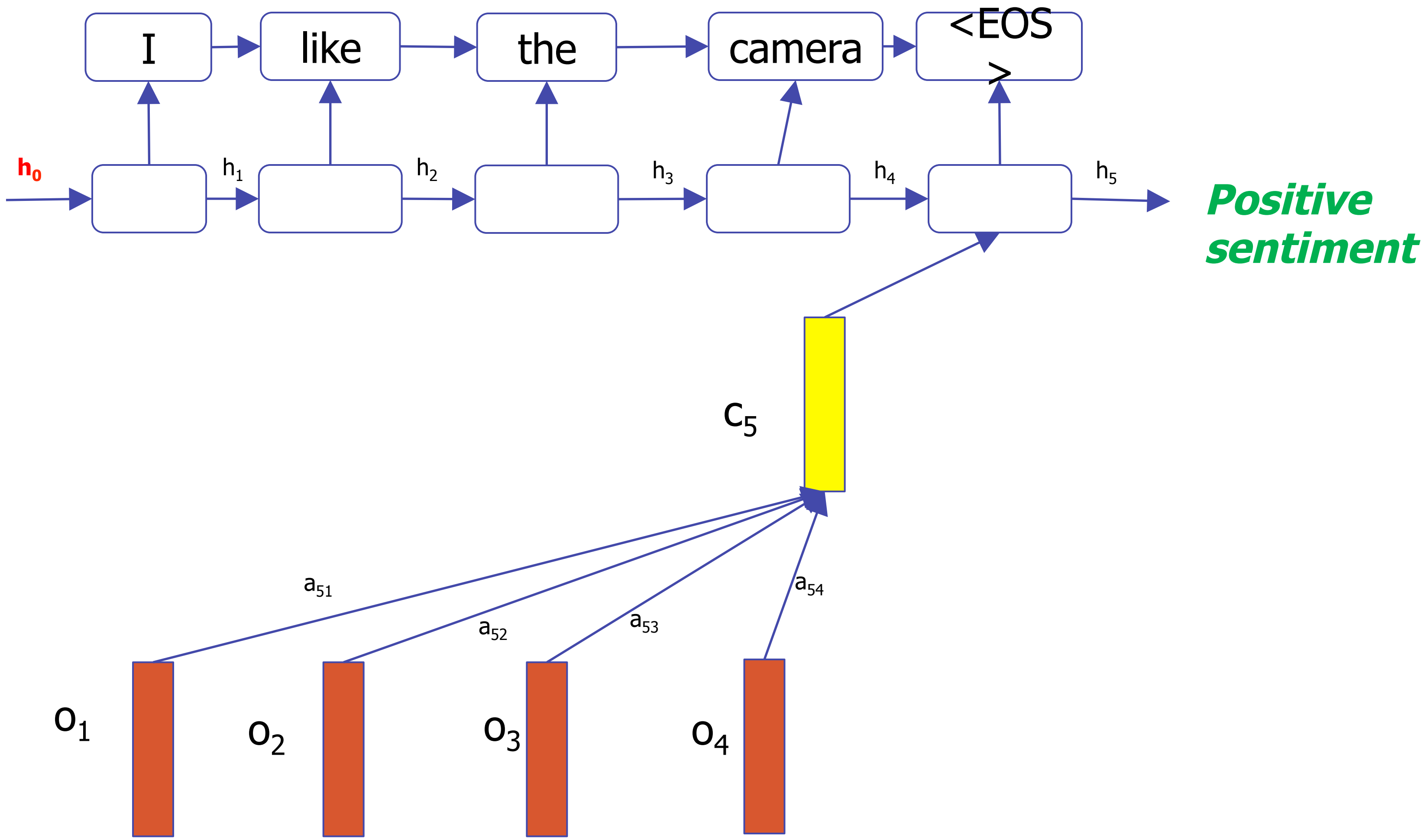




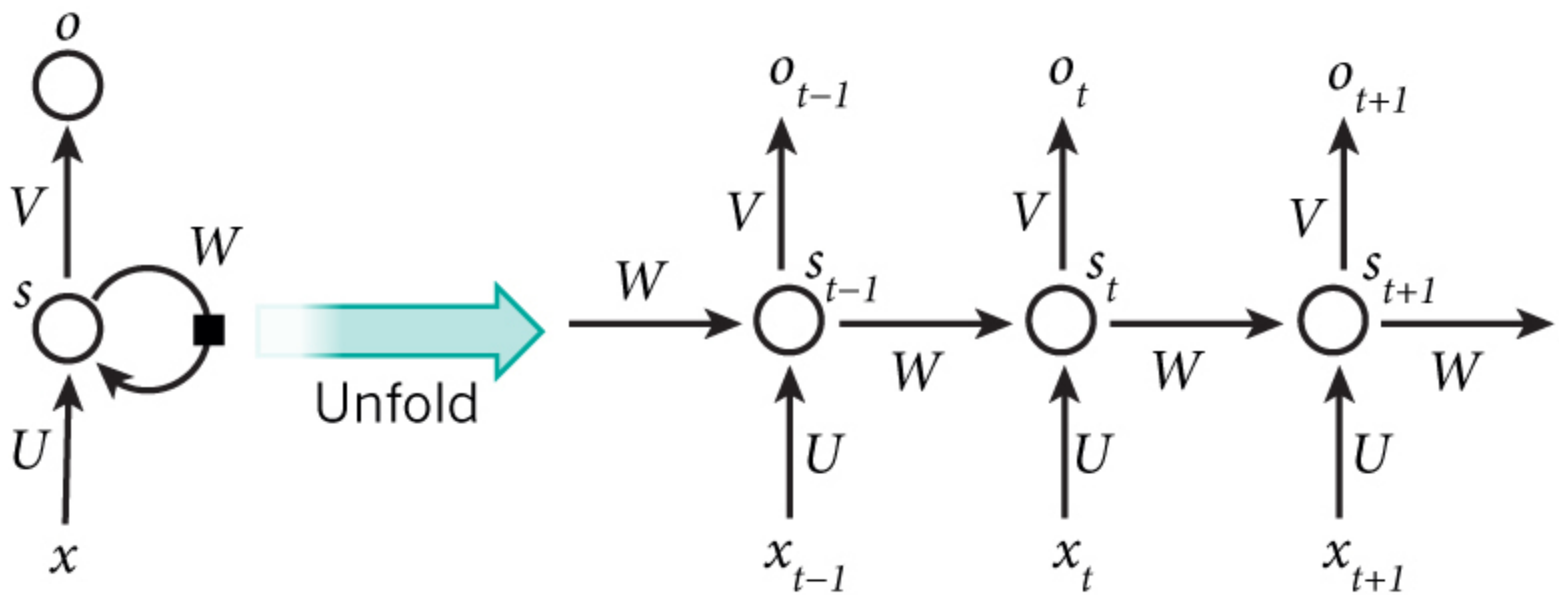
I
like
h0
h1
h2
c2
a21
a22
a23
a24
o1
o2
o3
o4


I
like
the
h_0
h_1
h_2
h_3
c_3
a_{31}
a_{32}
a_{33}
a_{34}
o_1
o_2
o_3
o_4


I
like
the
camera
h0
h1
h2
h3
h4
c4
a41
a42
a43
a44
o1
o2
o3
o4


I
like
the
camera
<EOS>
h0
h1
h2
h3
h4
h5
Positive sentiment
c5
a51
a52
a53
a54
o1
o2
o3
o4

Back to RNN model



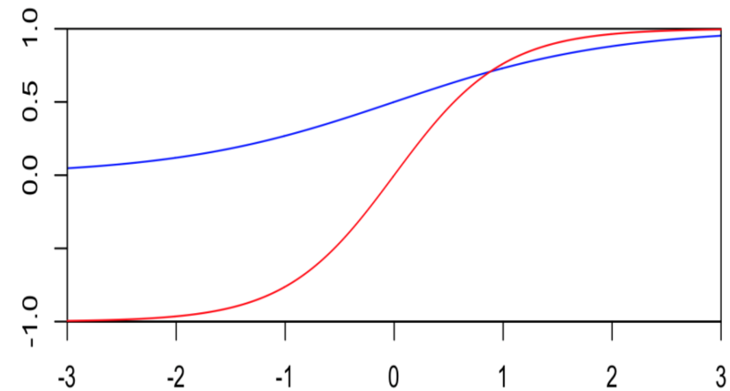
Notation: input and state

- x_t is the input at time step t. For example, could be a one-hot vector corresponding to the second word of a sentence.
- s_t is the hidden state at time step t. It is the "memory" of the network.
- $s_t = f(U.x_t + Ws_{t-1})$ ***U* and *W* matrices are learnt**
- f is a function of the input and the previous state
- Usually *tanh* or *ReLU* (approximated by *softplus*)

Tanh, *ReLU* (rectifier linear unit) and Softplus

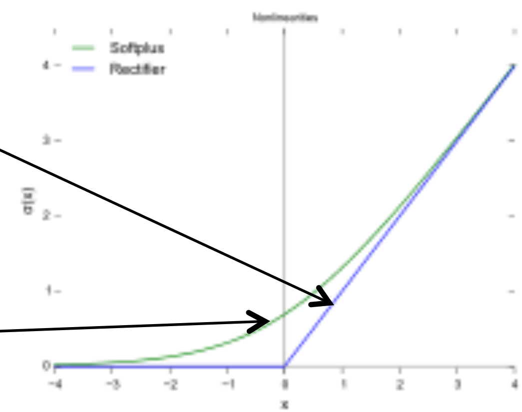
$$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

tanh =



$$f(x) = \max(0, x)$$

$$g(x) = \ln(1 + e^{x})$$



Notation: output

- o_t is the output at step t

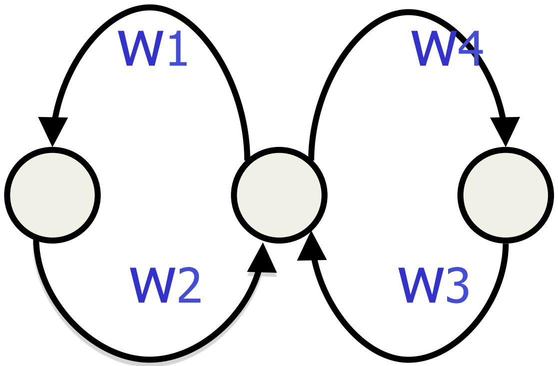
- For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary

- $o_t = softmax(V.s_t)$

Operation of RNN

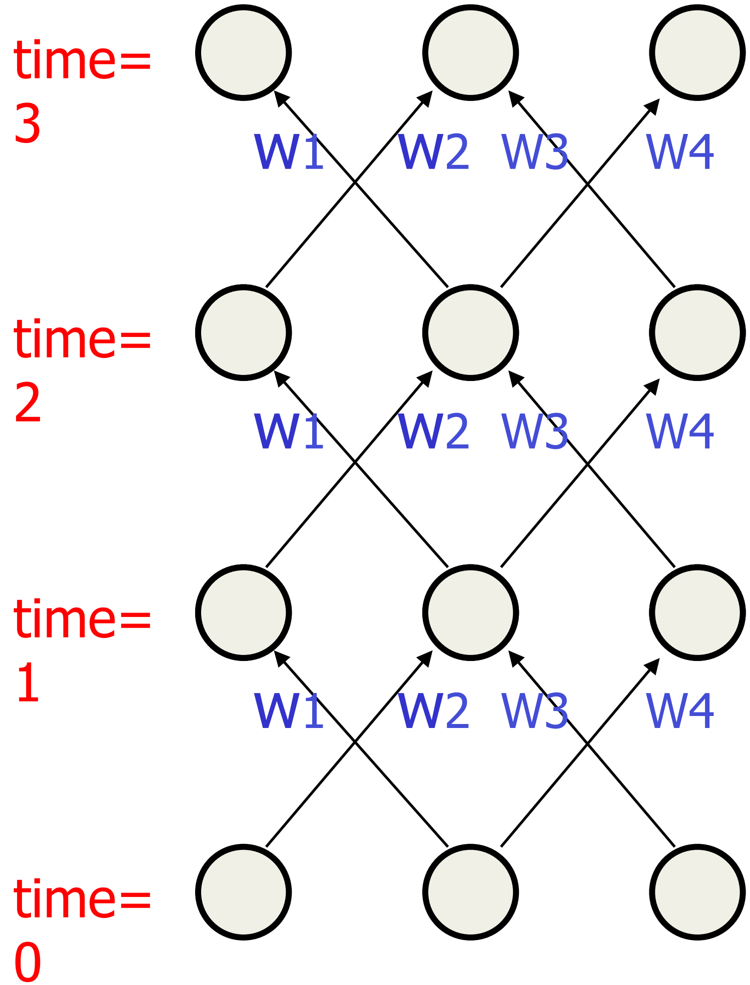
- RNN shares the same parameters (*U*, *V*, *W*) across all steps
- Only the input changes
- Sometimes the output at each time step is not needed: e.g., in sentiment analysis
- Main point: the hidden states !!

The equivalence between feedforward nets and recurrent nets



Assume that there is a time delay of 1 in using each connection.

The recurrent net is just a layered net that keeps reusing the same weights.



Reminder: Backpropagation with weight constraints

- **Linear constraints between the weights.**
- **Compute the gradients as usual**
- **Then modify the gradients so that they satisfy the constraints.**
- **So if the weights started off satisfying the constraints, they will continue to satisfy them.**

Example

$$To\ \ constrain: \quad w_1 = w_2$$

$$we\ \ need: \quad \Delta w_1 = \Delta w_2$$

$$compute: \quad \frac{\partial E}{\partial w_1} \quad and \quad \frac{\partial E}{\partial w_2}$$

$$use \quad \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2} \quad for\ w_1\ and\ w_2$$

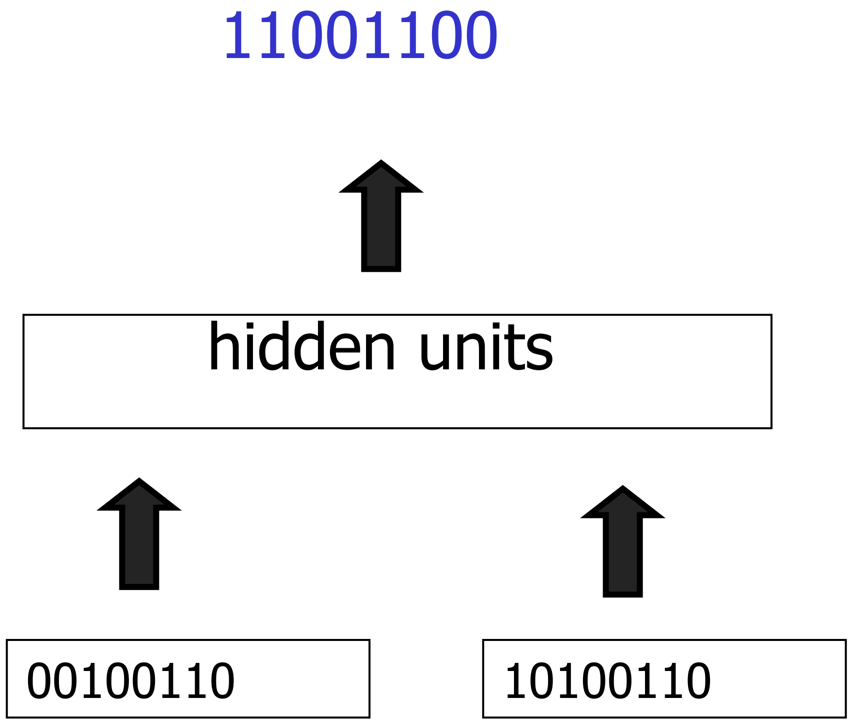
Backpropagation through time (BPTT algorithm)

- The forward pass at each time step.
-
- The backward pass computes the error derivatives at each time step.
- After the backward pass we add together the derivatives at all the different times for each weight.

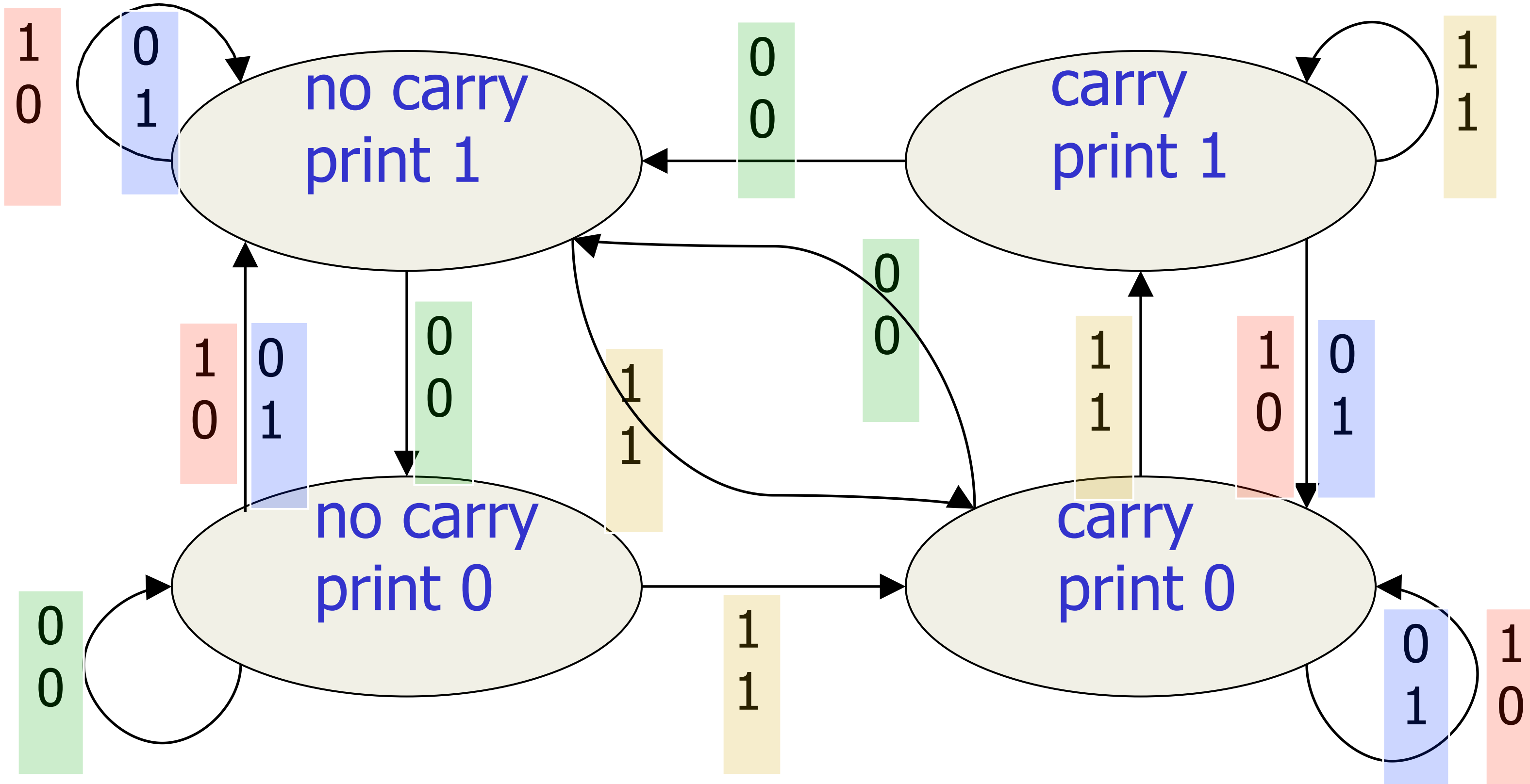
Binary addition using recurrent network (Jeffrey Hinton's lecture)

- Feed forward n/w

- But problem of variable length input



The algorithm for binary addition



This is a finite state automaton. It decides what transition to make by looking at the next column. It prints after making the transition. It moves from right to left over the two input numbers.

A recurrent net for binary addition

- Two input units and one output unit.
- Given two input digits at each time step.
- The desired output at each time step is the output for the column that was provided as input two time steps ago.
  - It takes one time step to update the hidden units based on the two input digits.
  - It takes another time step for the hidden units to cause the output.

0 0 1 1 0 1 0 0

0 1 0 0 1 1 0 1

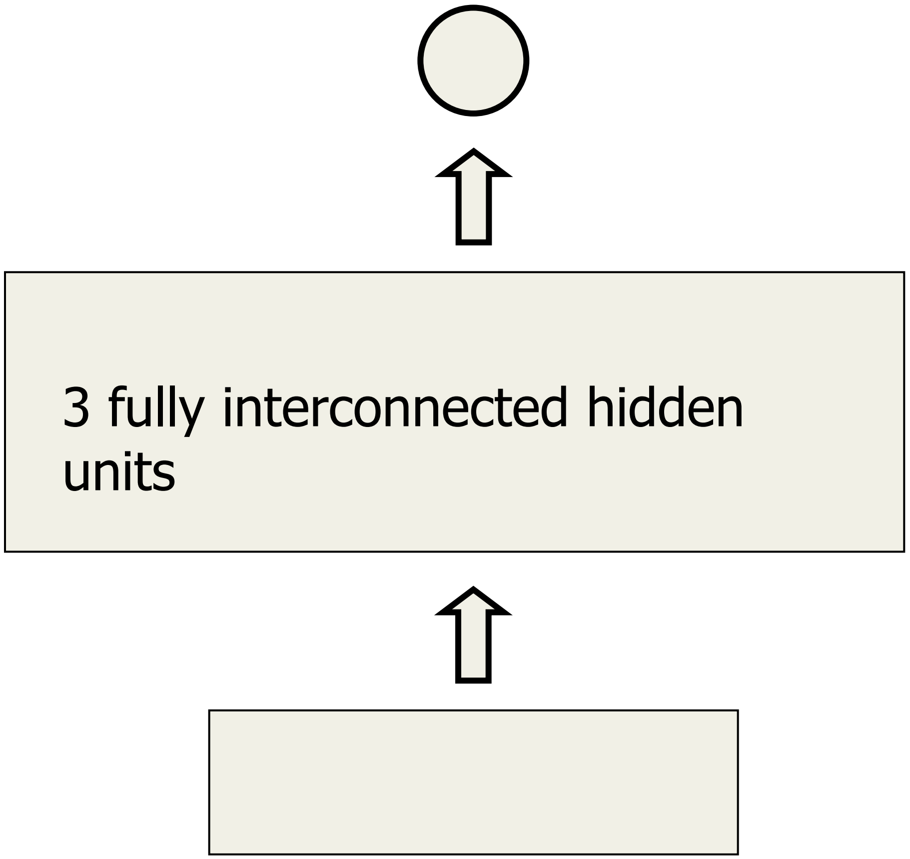
1 0 0 0 0 0 0 1

time

The connectivity of the network

- The input units have feed forward connections

- Allow them to vote for the next hidden activity pattern.



What the network learns

- Learns four distinct patterns of activity for the 3 hidden units.
- Patterns correspond to the nodes in the finite state automaton
- Nodes in FSM are like activity vectors
- The automaton is restricted to be in exactly one state at each time
- The hidden units are restricted to have exactly one vector of activity at each time.

Recall: Backpropagation Rule

- General weight updating rule:

$$\Delta w_{ji} = \eta \delta j o_i$$

- Where

$$\delta_j = (t_j - o_j) o_j (1 - o_j) \quad \text{for outermost layer}$$

$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j) o_i \ \text{for hidden layers}$$

The problem of exploding or vanishing gradients (1/2)

- If the weights are small, the gradients shrink exponentially

- If the weights are big the gradients grow exponentially.

- Typical feed-forward neural nets can cope with these exponential effects because they only have a few hidden layers.

The problem of exploding or vanishing gradients (2/2)

- In an RNN trained on long sequences (*e.g.* sentence with 20 words) the gradients can easily explode or vanish.
  - We can avoid this by initializing the weights very carefully.
- Even with good initial weights, its very hard to detect that the current target output depends on an input from many time-steps ago.
  - So RNNs have difficulty dealing with long-range dependencies.

Vanishing/Exploding gradient: solution

- LSTM
- Error becomes “trapped” in the memory portion of the block
- This is referred to as an "error carousel‟
- Continuously feeds error back to each of the gates until they become trained to cut off the value
- (to be expanded)

Application of RNN

Language Modeling

Definitions etc.

- What is Language Modeling:

  conditional probability of a word given the previous words in the sequence

- What is its mathematical expression

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

- How do we compute P (w_n | w^{n-1}_{n-N+1}):

  Count (w_n w_{n-1} .. W_{n-N+1})/ Count (w_{n-1} .. $W_{n-N+1)}$

An Example

$$P(w_n|w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

- <s> I am Sam </s>
- <s> Sam I am </s>
- <s> I do not like green eggs </s>

$$P(\texttt{I}\,|\,\texttt{<s>}) = \tfrac{2}{3} = .67 \qquad P(\texttt{Sam}\,|\,\texttt{<s>}) = \tfrac{1}{3} = .33 \qquad P(\texttt{am}\,|\,\texttt{I}) = \tfrac{2}{3} = .67$$

$$P(\texttt{</s>}\,|\,\texttt{Sam}) = \tfrac{1}{2} = 0.5 \qquad P(\texttt{Sam}\,|\,\texttt{am}) = \tfrac{1}{2} = .5 \qquad P(\texttt{do}\,|\,\texttt{I}) = \tfrac{1}{3} = .33$$

- Suppose we add another sentence <s> I do like salad </s>
- Now, what is the probability P(I| <s>), P(do| I)

Estimating Sentence Probabilities

- P(<s> I want english food </s>) =
  P(i|<s>)*
  P(want|I)*
  P(english|want)*
  P(food|english)*
  P(</s>|food)*
  =.000031

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1})$$
$$= \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

$$P(w_n \mid w_1^{n-1}) \approx P(w_n \mid w_{n-N+1}^{n-1})$$

Application of Sentence Probability

- Evaluate various possible translations in Statistical Translation Systems

    he briefed to reporters on the chief contents of the statement
    he briefed reporters on the chief contents of the statement
    he briefed to reporters on the main contents of the statement
    he briefed reporters on the main contents of the statement

- Similarly in Speech Recognition System

    A friend in deed is a friend indeed
    A fred in need is a friend indeed
    Afraid in need is a friend indeed
    A friend in need is a friend indeed

Another Application: Transliteration

- Mapping a written word from one language- script pair to another language-script pair:
  - युधिष्ठिर – *Yudhisthir*, Car – कार
  - This looks like a straight-forward problem
  - Define a simple mapping
  - Or is it *Yudhisthira* ? On web search:
    - 15,900 hits for *Yudhisthir*, vs 50,200 for *Yudhisthira*
    - 184,000 hits for *Qatil* vs. 7,300,000 for *Katil*
      - BTW, *Qatil* is the correct spelling
  - Generate all possible candidates and then somehow rank them

LM with RNN

The prediction task

$$x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$$

$$\begin{aligned} h_t &= \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right) \\ \hat{y}_t &= \operatorname{softmax}\left(W^{(S)} h_t\right) \end{aligned}$$

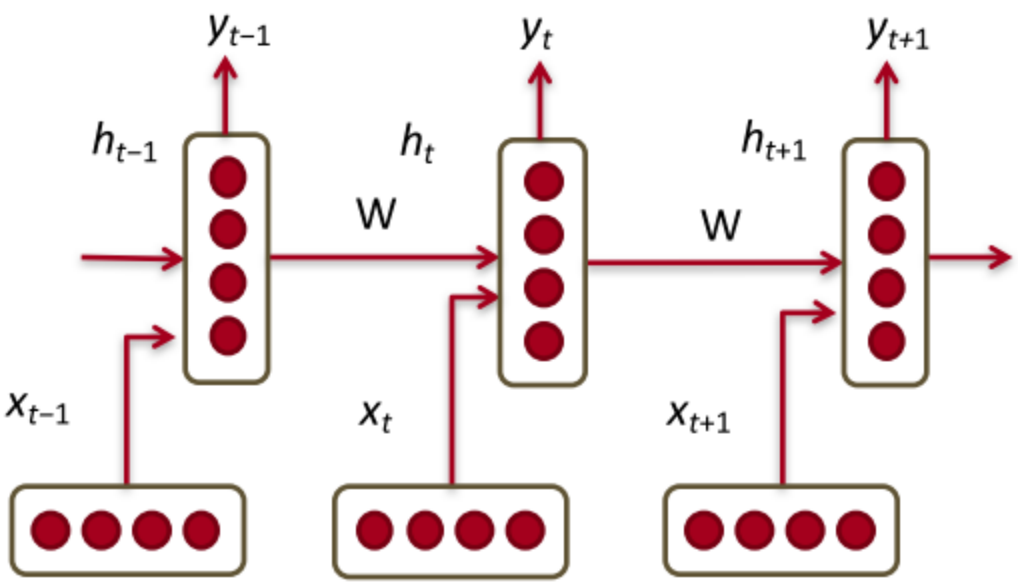
$$\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$$

Goal: minimize cross entropy

$$J = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_{t,j}\log \hat{y}_{t,j}$$

Perplexity: 2^J

Schematic



Ack: Socher lecture on RNN

Vanishing gradient problem hits again!

- General weight updating rule:

$$\Delta w_{ji} = \eta \delta j o_i$$

- Where

$$\delta_j = (t_j - o_j) o_j (1 - o_j) \quad \text{for outermost layer}$$

$$= \sum_{k \in \text{next layer}} (w_{kj} \delta_k) o_j (1 - o_j) o_i \text{ for hidden layers}$$

Trick for solving exploding/ vanishing gradient

- Clipping! (Pascanu et al, 2013)

- Do not let the gradient rise *above* (case of *exploding*) or fall *below* (case of *vanishing*) a threshold

- Facilitated by using ReLUs

Opinion Mining with Deep Recurrent Nets

(Irsoy and Cardie, 2014)

Goal

- Classify each word as direct subjective expressions (DSEs) and expressive subjective expressions (ESEs)
- DSE: Explicit mentions of private states or speech events Expressing private states
- ESE: Expressions that indicate sentiment, emotion, etc without explicitly conveying them

Annotation (1/2)

- In BIO notation (tags either begin-of-entity (B_X) or continuation-of-entity (I_X)):

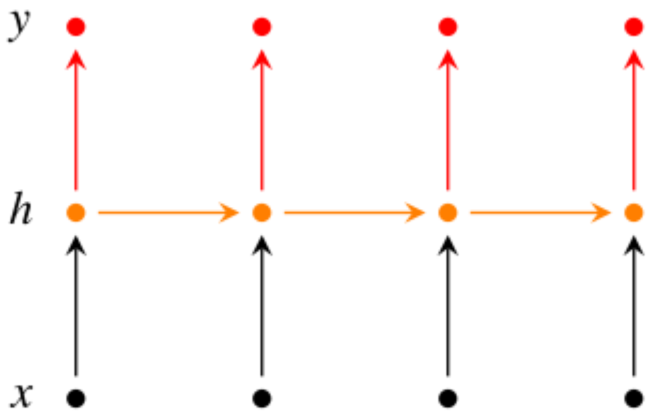
The committee, [as usual]$_{ESE}$, [has refused to make any Statements]$_{DSE}$

Annotation (2/2)

The	committee	,	as	usual	,	has
O	O	O	B_ESE	I_ESE	O	B_DSE

refused	to	make	any	statements	.
I_DSE	I_DSE	I_DSE	I_DSE	I_DSE	O

Unidirectional RNN

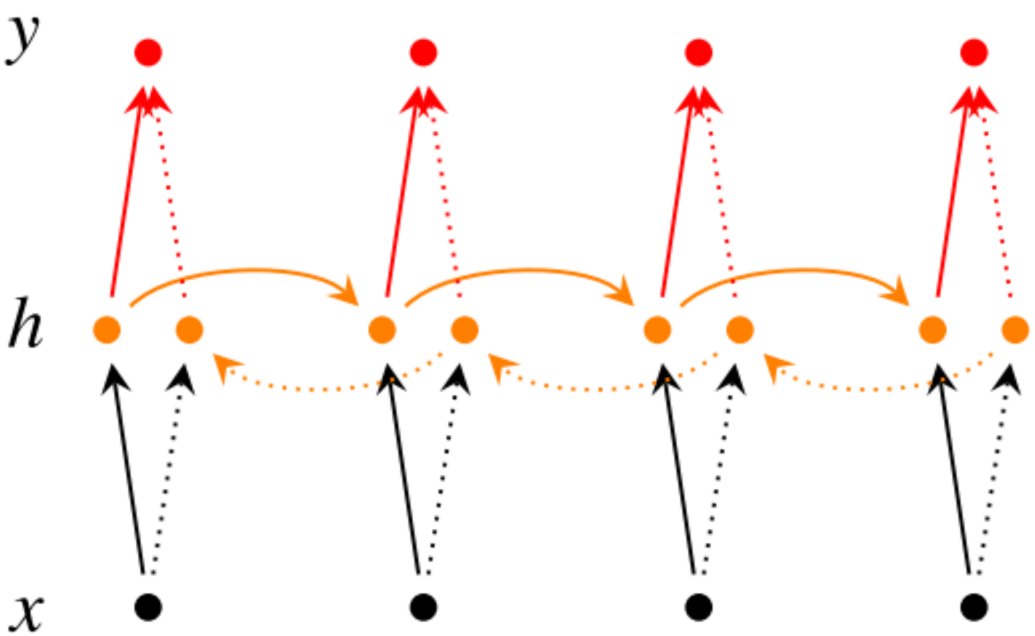


$$h_t = f(Wx_t + Vh_{t-1} + b)$$
$$y_t = g(Uh_t + c)$$

Notation from Irsoy and Cardie

- *x* represents a token (word) as a vector.
- *y* represents the output label (B, I or O)
- *h* is the memory, computed from the past memory and current word. It summarizes the sentence up to that time.

Bidirectional RNN

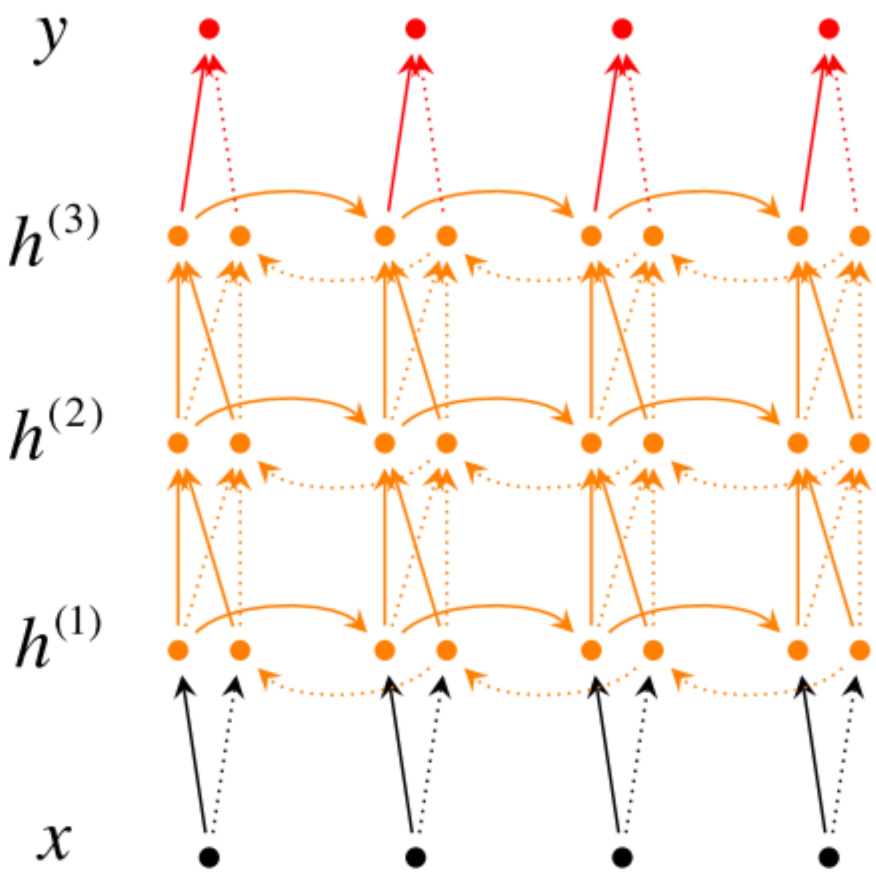


$$\overrightarrow{h}_t = f(\overrightarrow{W} x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

$$\overleftarrow{h}_t = f(\overleftarrow{W} x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\overrightarrow{h}_t;\overleftarrow{h}_t] + c)$$

Deep Bidirectional RNN



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Data

- MPQA 1.2 corpus (Wiebe et al., 2005)
- consists of 535 news articles (11,111 sentences)
- manually labeled with DSE and ESEs at the phrase level

F1 score