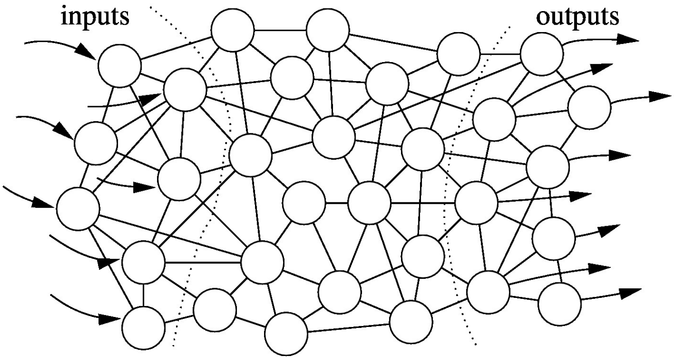
Constraint satisfaction

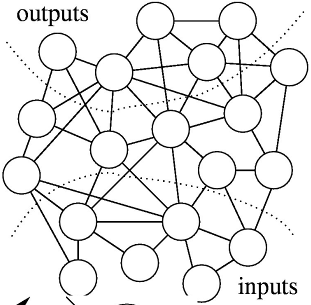


- Units represent **hypotheses** about parts of a problem
- Weights code **constraints** on how hypotheses can combine (i.e., the degree to which they are consistent or inconsistent)
- Possible **solutions** correspond to particular patterns of active units
- External input introduces **bias** to favor one possible solution over others

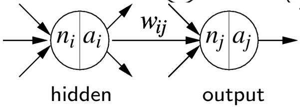
Temporal processing: Recurrent networksGeneralized Delta rule ("back-propagation")

Learning in networks with unrestricted connectivity:
Back-propagation through time

- Repeatedly update unit activations synchronously (first n_j, then a_j)
- Store entire activation history of each unit
- Attribute error to sending activations computed earlier in time



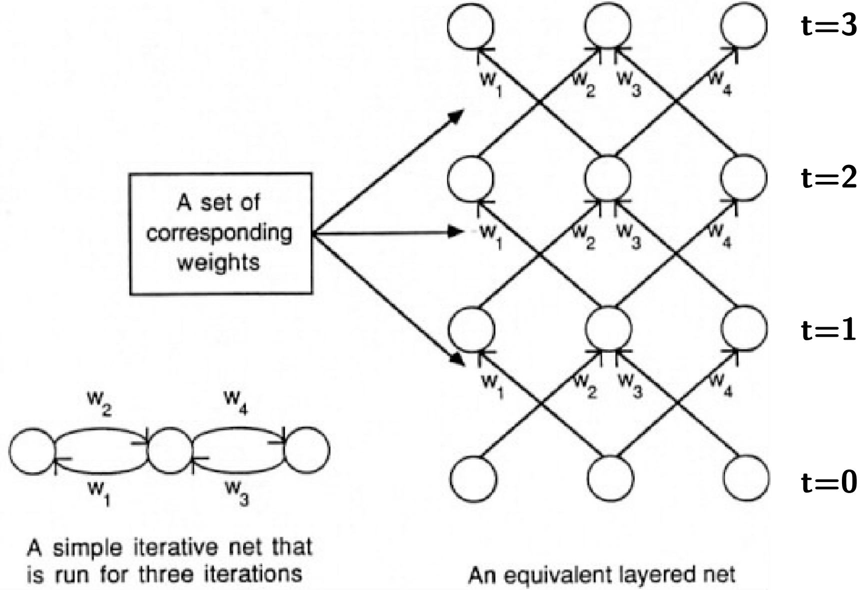
$$
\begin{aligned}
n_j &= \sum_i a_i \, w_{ij} \\
a_j &= \frac{1}{1+\exp\left(-n_j\right)} \\
\text{Error} \quad E &= \frac{1}{2}\sum_j \left(t_j - a_j\right)^2
\end{aligned}
$$



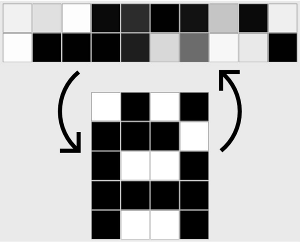
$$
\begin{array}{ccccccccc}
 & & w_{ij} & \searrow & & & t_j & \searrow & \\
n_i & \rightarrow & a_i & \rightarrow & n_j & \rightarrow & a_j & \rightarrow & E
\end{array}
$$

∂E

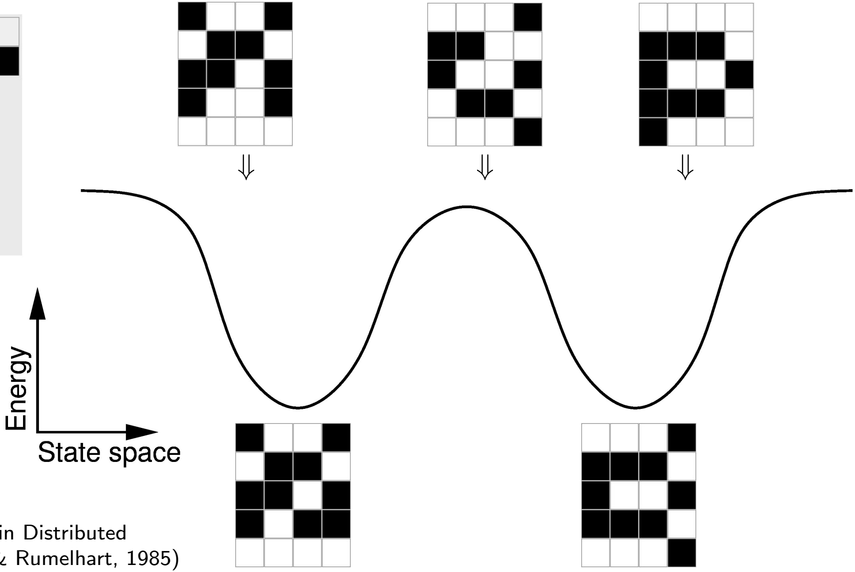
"Unfolding" a recurrent network into a feedforward one



Recurrent processing, pattern completion, and attractors



Recurrence can clean up noisy patterns into correct (clean) ones as long as input falls within correct basin of attraction

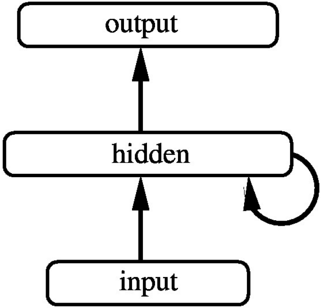


(like noisy dogs, cats, bagels in Distributed Memory Model; McClelland & Rumelhart, 1985)

Temporal processing: Simple recurrent networks (Elman, 1990)

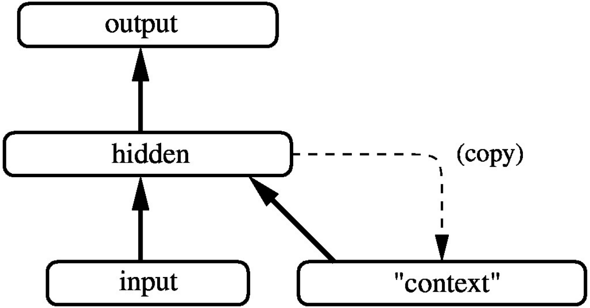
Fully recurrent network

- Computationally intensive to simulate
- Must update unit activities multiple times per input



Simple recurrent network (SRN)

- Adapt feedforward network to learn temporal tasks
- Computationally efficient but functionally limited compared to fully recurrent network



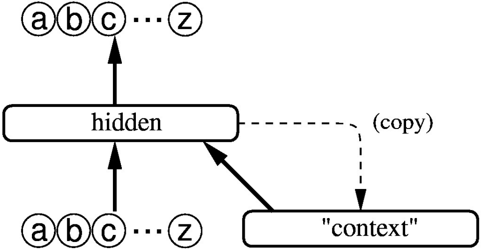
Sequential prediction tasks

- Input is a sequence of discrete elements (e.g., letters, words)
- Target is next item in the sequence
  - *Self-supervised* learning: environment provides both inputs and targets
- Network is **guessing**; cannot be completely correct but can perform better than chance if the input sequence is **structured**
- Given a particular sequence of past elements and current input, total error is minimized by generating the **probabilities** of next elements (i.e., their proportion of occurrence in this context across examples)
  - Across examples, each next element "votes for" its targets; result is average
  - Example: Train on ABAC, ABCA and ACBB

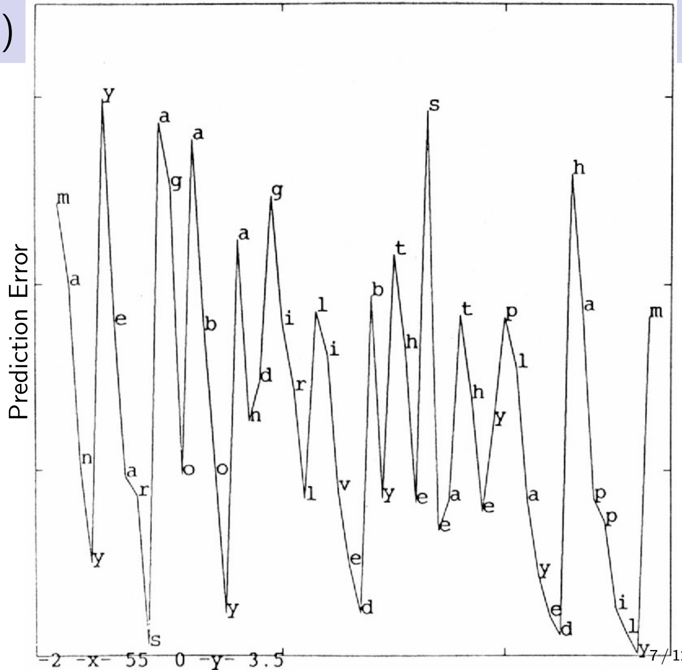
		Activations		
Time	Input	A	B	C
0	A	0.0	0.67	0.33
1	B	0.5	0.0	0.5
2	C	1.0	0.0	0.0

Letter prediction (Elman, 1990)

Network is trained to predict the next letter in text (without spaces)



Network **discovers** word boundaries as peaks in letter prediction error



Word prediction (Elman, 1990)

- Network is presented with sequence of words (localist representation for each)
- Sequence is constructed from 2-word or 3-word sentences
- **No punctuation or sentence boundaries**
- Trained to predict the next word within and across sentences [analogous to predicting letters within/across words]

Categories of Lexical Items Used in Sentence Simulation

Category	Examples
NOUN-HUM	man, woman
NOUN-ANIM	cat, mouse
NOUN-INANIM	book, rock
NOUN-AGRESS	dragon, monster
NOUN-FRAG	glass, plate
NOUN-FOOD	cookie, break
VERB-INTRAN	think, sleep
VERB-TRAN	see, chase
VERB-AGPAT	move, break
VERB-PERCEPT	smell, see
VERB-DESTROY	break, smash
VERB-EAT	eat

Sentences

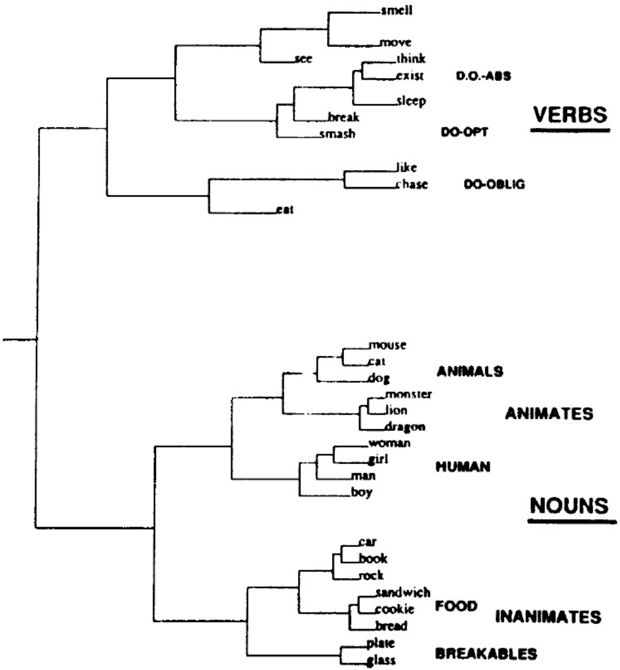
Templates for Sentence Generator

WORD 1	WORD 2	WORD 3
NOUN-HUM	VERB-EAT	NOUN-FOOD
NOUN-HUM	VERB-PERCEPT	NOUN-INANIM
NOUN-HUM	VERB-DESTROY	NOUN-FRAG
NOUN-HUM	VERB-INTRAN	
NOUN-HUM	VERB-TRAN	NOUN-HUM
NOUN-HUM	VERB-AGPAT	NOUN-INANIM
NOUN-HUM	VERB-AGPAT	
NOUN-ANIM	VERB-EAT	NOUN-FOOD
NOUN-ANIM	VERB-TRAN	NOUN-ANIM
NOUN-ANIM	VERB-AGPAT	NOUN-INANIM
NOUN-ANIM	VERB-AGPAT	
NOUN-INANIM	VERB-AGPAT	
NOUN-AGRESS	VERB-DESTROY	NOUN-FRAG
NOUN-AGRESS	VERB-EAT	NOUN-HUM
NOUN-AGRESS	VERB-EAT	NOUN-ANIM
NOUN-AGRESS	VERB-EAT	NOUN-FOOD

Learned word representations

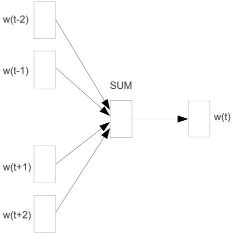
Hierarchical clustering of hidden representations for words after training

- Network has learned parts of speech and (rudimentary) semantic similarity

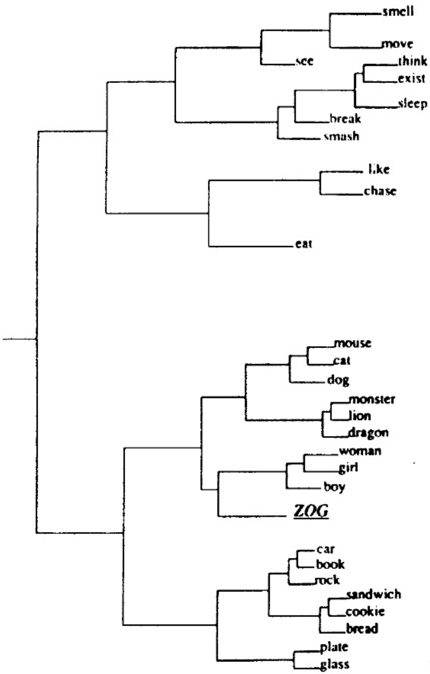


Generalization to ZOG (man)

- Test network on novel input (ZOG) with no overlap with existing words, where ZOG occurs everwhere that MAN did
  - **No additional training**
  - Equivalent to having **no input** and only using context (representation of preceding text) to make prediction
  - Like a version of **word2vec**



- Network produces hidden representation for ZOG that is highly similar to that of MAN
  - The contexts in which MAN occurs are sufficient to predict its (likely) occurrence



Generalization to ZOG (man)

- Test network on novel input (ZOG) with no overlap with existing words, where ZOG occurs everwhere that MAN did
  - **No additional training**
  - Equivalent to having **no input** and only using context (representation of preceding text) to make prediction

- Network produces hidden representation for ZOG that is highly similar to that of MAN
  - The contexts in which MAN occurs are sufficient to predict its (likely) occurrence
  - Like a version of **word2vec**