Language Acquisition with Recurrent Neural Networks by Ryotaro Kamimura

Presented by Michael A. Dandini
4/21/2005

Outline

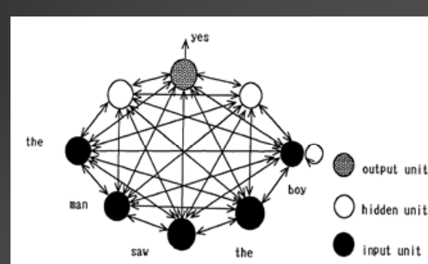
- Introduction
- Problem of Language Acquisition
- The Recurrent Neural Network
- Experiment
- Results
- Conclusion

Language Acquisition

- Grammatical Competence
  - E.g. Is a sentence grammatically correct?
  - Considered equivalent to learning a language (“observationally adequate”)
- “Creativity” of Language
  - Ability to produce and understand new sentences
  - Experiment “fudges” on this

Recurrent Neural Networks

- In class, we studied Feed-Forward networks…
- For this experiment, a <u>recurrent</u> network was used.



Why Recurrent Networks?

- Advantages
  - Recurrency acts as "memory"
  - No constraints on topology
  - Generally more flexible
- Disadvantages
  - Best architecture for a specific problem unknown
  - Older information is unreliable

Training RNNs

- Must consider all units at every step
- Network dynamics based on systems of differential equations
- Basic Idea:
  - Get systems to converge
  - Use solutions of systems to train weights

Dynamics of a RNN

$$\frac{dv_i}{dt} = -v_i + f\left(\sum_{j}^{K} w_{ij} v_j\right) + \theta_i$$

- Where
  - v_i = the output of unit i
  - w_{ij} = the weight from unit j to unit i
  - θ_I = the bias at unit I
  - K = the number of units
  - f() is <u>any</u> differentiable function

Solving the 1st Equation

- At equilibrium, we should get:

$$\frac{dv_i}{dt} = 0$$

- Using this, we can get the solution…
  - Where v_i* is the solution for unit i

$$v_i^* = f\left(\sum_{j}^{K} w_{ij} v_j^*\right) + \theta_i$$

Calculating Error

The error of the whole network is given by

$$E(v^*) = \frac{1}{2}\sum_{i}^{K} J_i^2$$

Where

$$J_i = \begin{cases} \tau_i - v_i^*, & \text{if } i \in \text{output nodes} \\ 0, & \text{otherwise} \end{cases}$$

Adjusting the Weights

- The Delta rule:
  - Uses gradient decent to minimize the error
  - η = the learning factor

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

- Working out the derivation, we get
  - Y = Number of output units

$$\Delta w_{ij} = \eta \sum_{k \in \text{output u.}}^{Y} J_k \frac{\partial v_k^*}{\partial w_{ij}}$$

Adjusting the Weights (cont.)

- Differentiating the term, $\frac{\partial v_k}{\partial w_{ij}}$, we get

$$\Delta w_{ij} = \eta f'(\sum_{j}^{K} w_{ij} v_j) z_i^* v_j^*$$

Where z_i^* is the solution to $\frac{dz_i}{dt} = -z_i + \sum_{j}^{K} \left[f'(\sum_{k}^{K} w_{jk} v_k) w_{ji} z_i \right] + J_i$

Adjusting the weights (cont.)

- In order to train the network, we must converge on v* and z*
  - v* approximated by **propagating**
  - z* approximated by **back-propagating**
  - Once convergence on both is reached, the weights are updated.

Experiment

- Network
  - “Fully Recurrent”
  - 100 input units
    - Sentence length limited to 10 words
    - 10 distinct units reserved for 10 distinct words
  - One output unit
  - “Several” hidden units

Training data

- Well-formed sentences created with:
  - Det Adj N V Det Adj N Prep Det N
  - Det N V Det Adj N Prep Det N
  - Det Adj N V Det N Prep Det N
  - Det N V Det N Prep Det N
  - Det N V Det Adj N
  - Det N V Det N
- Made ill-formed sentences by inverting order and random word choice.

Performance on New Sentences

- Tested “creativity”
- Two Parts
  - Sentences with same formulas as training set
  - Sentences with CFG:

    S ⇒ NP VP

    VP ⇒ V NP

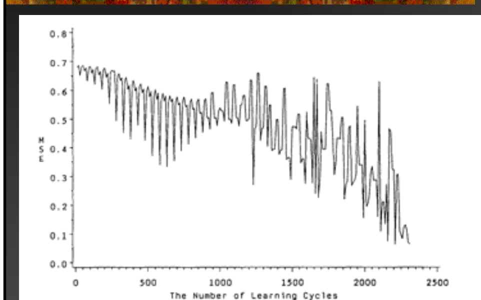
    VP ⇒ V NP Prep NP

    NP ⇒ Det Adj N

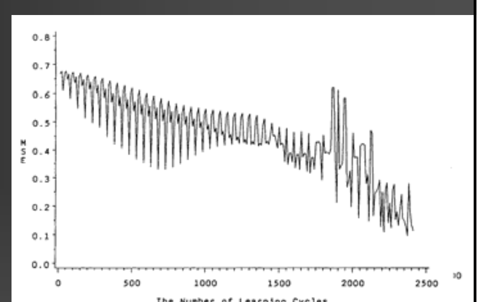
    NP ⇒ Det N

    That did <u>not</u> match previous formulas

Convergence of Network

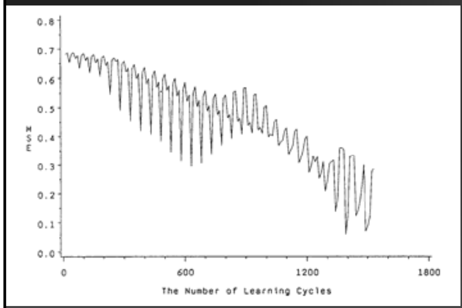


$\eta = 0.6$

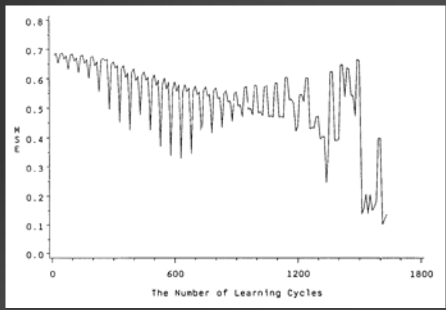


$\eta = 0.3$

Convergence (cont.)

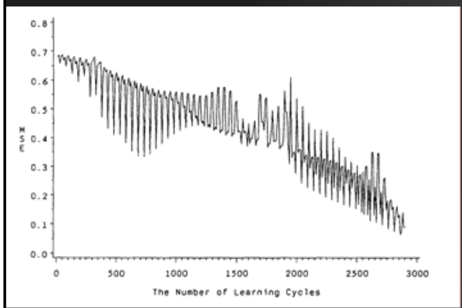


One propagation

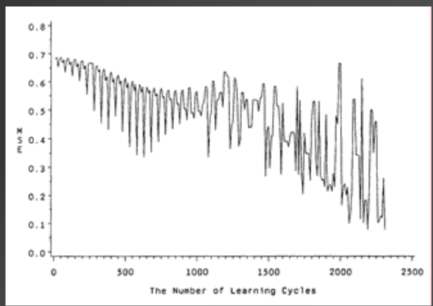


Two propagations

Convergence (cont.)

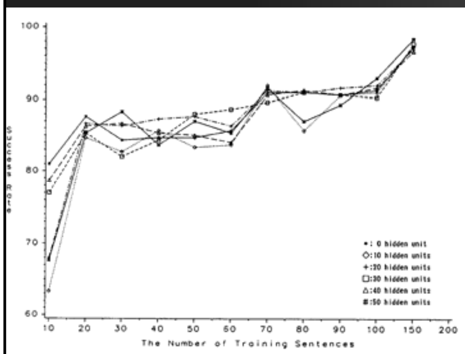


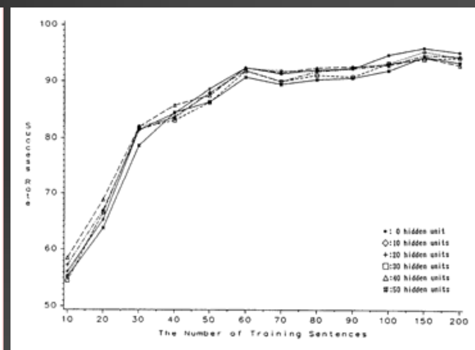
One back-propagation



Two back-propagations

New Sentences from Formulas

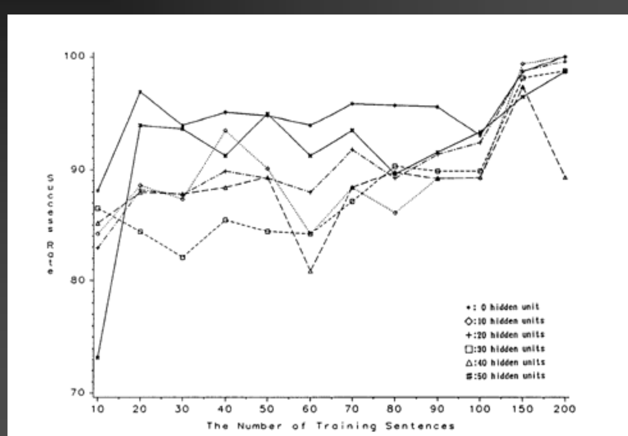




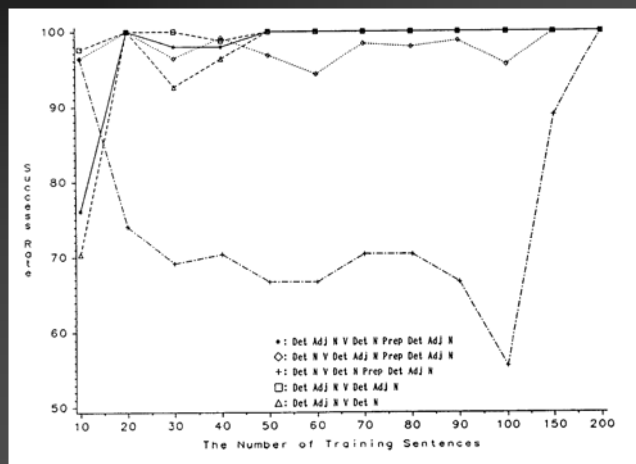
Well-Formed

Ill-formed

New Sentences from CFG



New Sentences from CFG (cont.)



Discussion

- Network more likely to detect:
  - Simple sentences
  - Pattern Adj+N
- Problems?
  - Maybe limited experiment too much?

Conclusion

- A recurrent neural network can detect the correctness of simple sentences
- Takes few propagations and back-propagations for the network to converge
- Performance of network is not related to the number of units

Resources

- Alpaydin, Ethem, 2004. *Introduction to Machine Learning*. Cambridge, MA: MIT Press.
- Kamimura, Ryotaro, 1991. Application of the Recurrent Neural Network to the Problem of Language Acquisition. *Proceedings of the Conference on Analysis of Neural Network Applications*, 14-28. Fairfax, VA: ACM.

Resources (cont.)

- Orr, Genevieve, 1999. "CS-449: Neural Networks." http://www.willamette.edu/~gorr/classes/cs449/intro.html.
- Pineda, Fernando J., 1987. Generalization of Back-Propagation to Recurrent Neural Networks. *Physical Review Letters* Vol. 59, No. 19, 2229-2232.