Computational Psycholinguistics

Lecture 11: Introduction to Connectionist Models

Afra Alishahi

January 26, 2009

(based on slides by Matthew Crocker and Marshall Mayberry)

Connectionist Modeling

- Connectionism was proposed as an alternative to the symbolic accounts of information processing
    - Motivation: design computers inspired by brain
    - Key ideas: distributed, implicit representations; dense connectivity; communication of ‘real values’ not ‘symbols’; single mechanism for rules and exceptions
- A functionalist assumption of language:
    - knowledge of language develops in the course of learning how to perform primary communicative tasks of comprehension and production

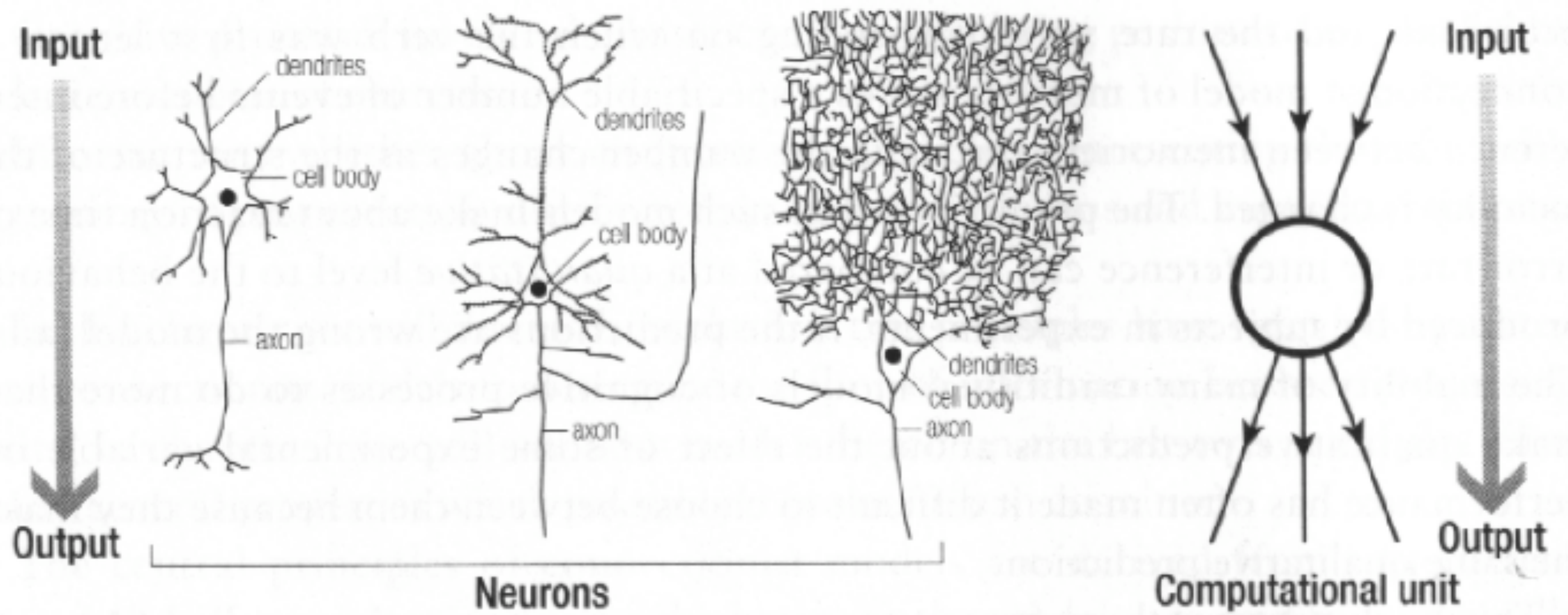
Connectionist Information Processing

- The idea of connectionist models is based on simple neuronal processing in the brain
  - Basic computational operation: one neuron receives input signals, processes them and passes the resulting information to other neurons
  - Learning: changing the strength of the connections between neurons
  - Cognitive processes: using large numbers of neurons to perform these basic computations in parallel
  - Information is distributed across many neurons and connections

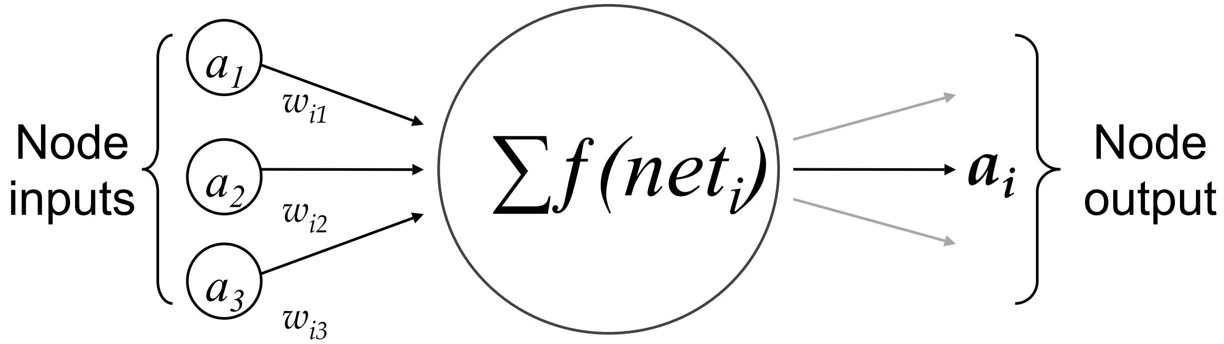
Assumptions about the brain ...

- Neurons integrate information: all neuron types sum inputs and compute an output
- Neurons encode the strength of their input in the output they pass to other neurons: firing rate
- Brain structure is layered: information passes through sequences of independent structures
- Influence of one neuron upon another depends on connection strength
- Learning is accomplished through changing connection strengths

Neurons versus Nodes



Basic Structure of Nodes



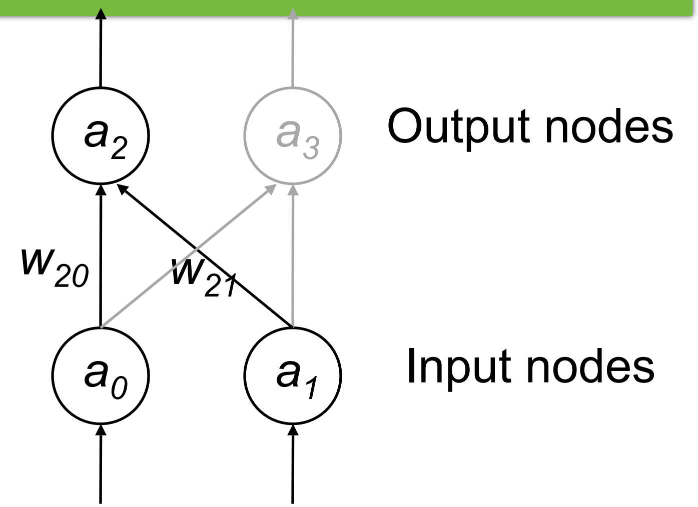
- Input connections represent the flow of activation from other nodes or some external source
- Each input connection has a *weight*, which determines its influence on the node
- A node *i* has an output activation $a_i = f(net_i)$ which is a function of the weighted sum of its input activations, net_i

$$net_i = \sum_j w_{ij} a_j$$

An example

- A one-layer network:

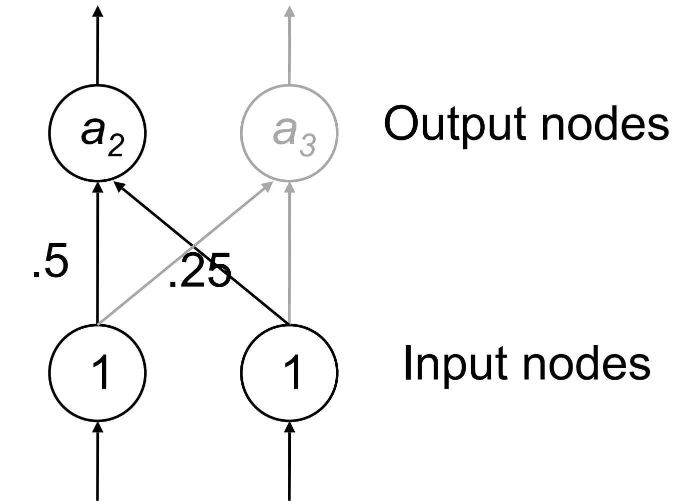
$$net_i = \sum_j w_{ij} a_j$$



- So the net input for a_2 is:

$$\text{net input } a_2 = w_{20} \cdot a_0 + w_{21} \cdot a_1$$

- Consider this network:



- The net input for node a_2 is:

$$1 \text{ x } .5 + 1 \text{ x } .25 = 0.75$$

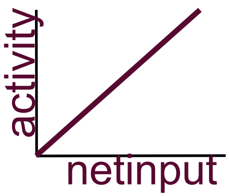
About weights

- Node j influences node i by passing information about its activity level
- The degree of influence it has is determined by the weight connecting node j to node i.
- Weights can be either positive or negative
    - Positive weights contribute activation to the net input
    - Negative weights lead to a reduction of the net input activation

Calculating the Activation

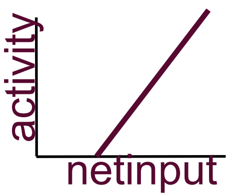
- Linear activation

$$f(net_i) = net_i$$
$$f(1.25) = 1.25$$



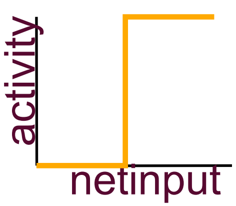
- Linear threshold
  (T=0.5)

$$\text{IF } net_i > T \text{ then } f(net_i) = net_i - T$$
$$\text{ELSE } f(net_i) = 0$$
$$f(1.25) = 1.25 - 0.5 = 0.75$$



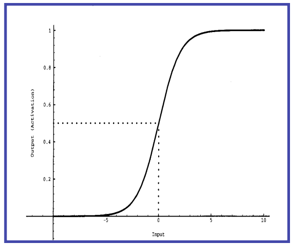
- Binary threshold
  (T=0.5)

$$\text{IF } net_i > T \text{ then } f(net_i) = 1$$
$$\text{ELSE } f(net_i) = 0$$
$$f(1.25) = 1$$



- Nonlinear activation
  (Sigmoid or “logistic”)

$$f(net_i) = \frac{1}{1 + e^{-net_i}}$$
$$f(1.25) = 0.777$$



About activation functions

- The activation function defines the relationship between the net input to a node, and its activation level (which is also its output)
- Most common in connectionist modeling: sigmoid/logistic
  - Activation ranges between 0 and 1
  - Rate of activation change is highest for net inputs around 0
  - Models neurons by implementing thresholding, a maximum activity, and smooth transition between states.

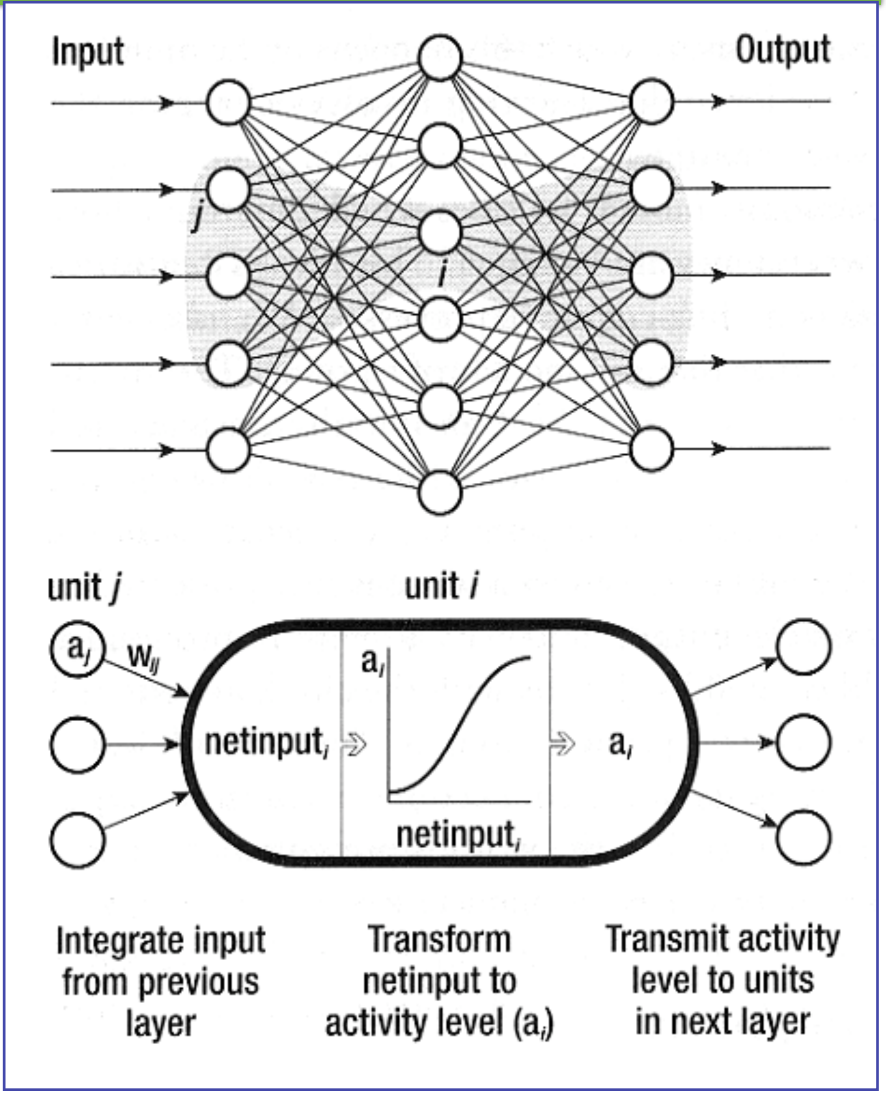
Summary of network architecture

- The activation of a unit i is represented by the symbol a_i
- The extent to which unit j influences unit i is determined by the weight w_{ij}
- The input from unit j to unit i is the product: $a_j * w_{ij}$
- For a node i in the network:

$$net_i = \sum_j w_{ij} a_j$$

- The output activation of node i is determined by the activation function, e.g. the logistic:

$$a_i = f(net_i) = \frac{1}{1 + e^{-net_i}}$$



Learning in Neural Networks

- Supervised learning in connectionist networks:
  - Adjusting connection weights to reduce the discrepancy between the actual output activation and the target output activation
- Procedure:
  - An input is presented to the network
  - Activations are propagated through the network
  - Outputs are compared to ‘correct’ outputs
  - Weights are adjusted to reduce error

The Delta Rule

- The Delta Rule: $$\Delta w_{ij} = (t_i - a_i)a_j\varepsilon$$
  - $(t_i - a_i)$ is the difference between the target output activation and the actual activation produced by the network
  - a_j is the activity of the contributing unit j
  - ε is the learning rate parameter.
    - How rapidly do we want to make changes?

Training the Network

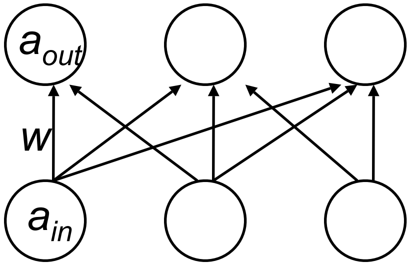
- Consider the AND function
  - Present stimulus: 0 0
  - Compute output activation
  - Compared with desired output (0)
  - Use Delta rule to change weights
  - Present next stimulus: 0 1
  - ...

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

- Key terms:
  - Epoch: a single presentation of all training examples
  - Sweep: a presentation of a single training example

Perceptrons *(Rosenblatt, 1958)*

- Perceptron: a simple, one-layer network:



$$\text{net}_{out} = \sum_{in} w \cdot a_{in}$$

- Binary threshold activation function:

$$\begin{aligned} a_{out} &= 1 \text{ if net}_{out} > \theta \\ &= 0 \text{ otherwise} \end{aligned}$$

- Learning: the perceptron convergence rule
  - Two parameters can be adjusted:
    - The threshold
    - The weights

$$\text{The error, } \delta = (t_{out} - a_{out})$$

$$\Delta\theta = -\varepsilon\delta$$

$$\Delta w = \varepsilon\delta a_{in}$$

Global Error

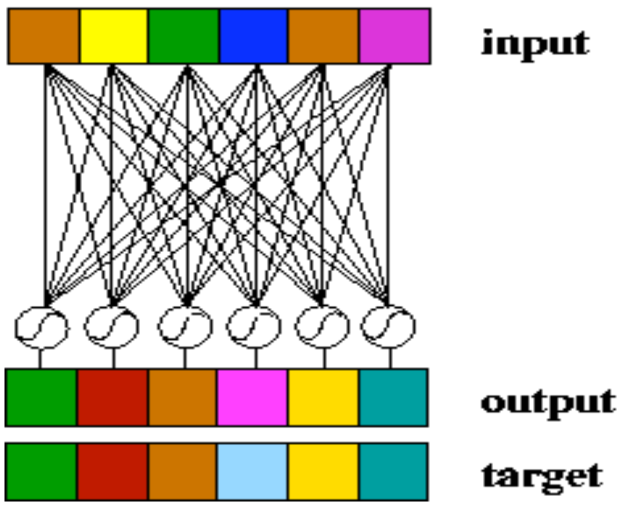
- We can define the global error of the network, as the average error across all input patterns, k:
  - One common measure is the square root of mean error or Root Mean Square (RMS)

$$\text{rms error} = \sqrt{\frac{\sum_k (\vec{t}_k - \vec{o}_k)^2}{k}}$$

  - Squaring avoids positive and negative errors canceling each other out

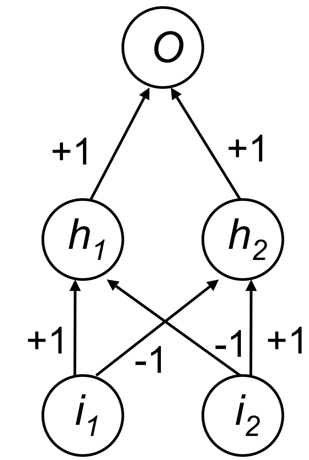
Learning in a nutshell

- Patterns are vectors on [0,1]
- Input pattern is passed through a weight matrix
- Net values are summed and squashed to [0,1]
- Output pattern is compared to target pattern
- Error between output and target is propagated back through weight matrix
- Weights are changed to minimize error



Hidden Units

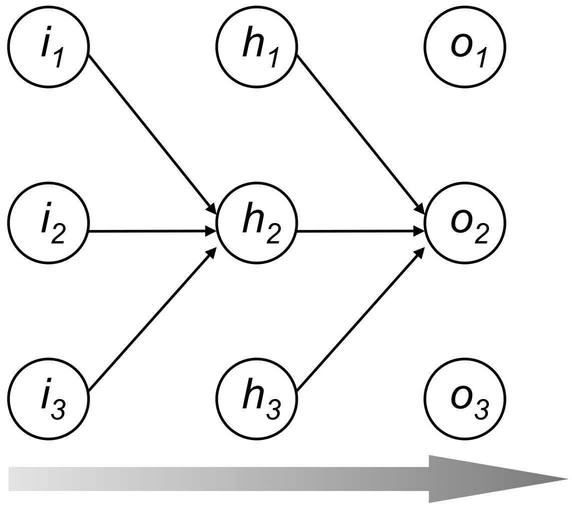
- One-layer networks can only simulate simple problems, whereas multi-layer networks can learn any mapping function
- Consider the following network:
  - two-layer, feedforward
  - 2 units in a 'hidden' layer



- Current learning rule can't be used for hidden units:
  - We don't know what the 'error' is at these nodes
  - Delta rule requires that we know the desired activation

$$\boxed{\Delta w = 2\varepsilon\delta F^{*} a_{in}}$$

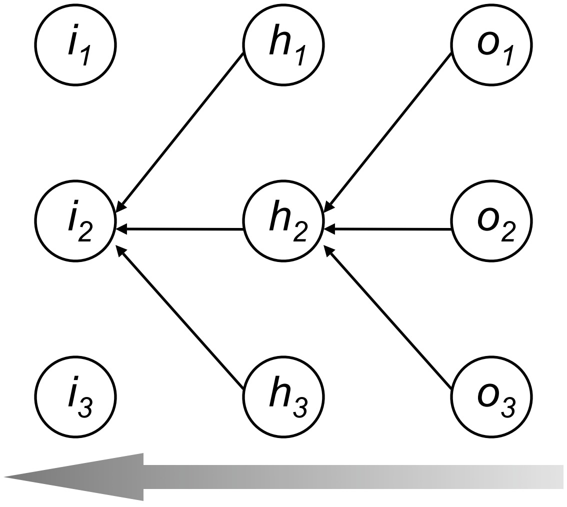
Backpropagation of Error



(a) Forward propagation of activity :

$$\text{net}_{out} = \sum w_{oh} \cdot a_{hidden}$$

$$a_{out} = f(\text{net}_{out})$$



(b) Backward propagation of error :

$$\text{err}_{hidden} = \sum w_{oh} \cdot \delta_{out}$$

$$\delta_{hidden} = f'(\text{net}_{hidden}) \cdot \text{err}_{hidden}$$

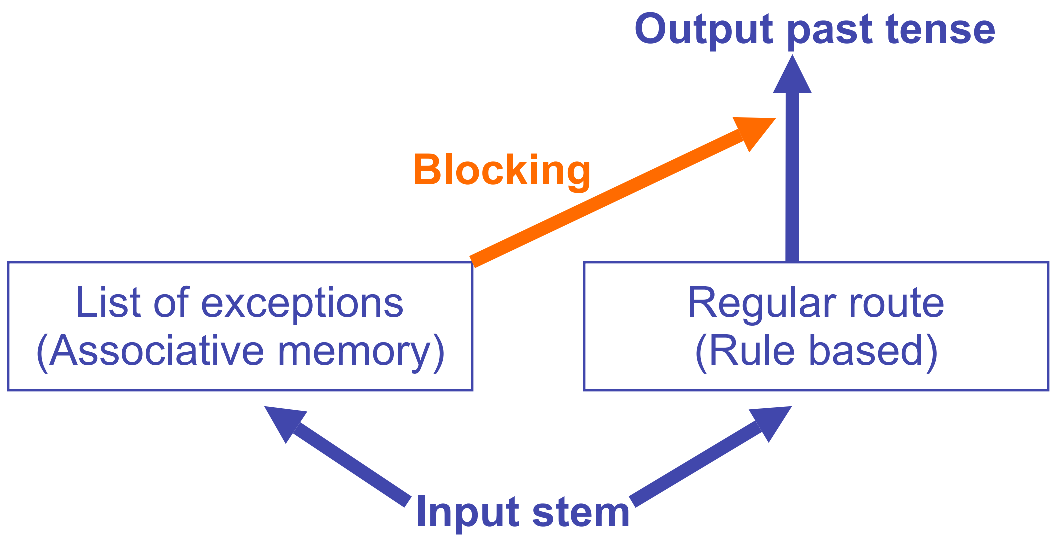
Example: Learning the Past Tense

- The problem of English past tense formation:
  - Regular formation: *stem* + 'ed'
  - Irregulars do show some patterns:
    - No-change: hit » hit (all end in a 't' or 'd')
    - Vowel-change: ring » rang, sing » sang
    - Arbitrary: go » went
- Over-regularizations are common: *"goed"*
  - These errors often occur after the child has already produced the correct irregular form: *"went"*
- The U-shaped learning curve has to be explained

A Symbolic Account: Dual-Route Model

- General pattern of behaviour:
  - At first, children learn past tenses by rote learning (i.e. memorizing each form)
  - Later they recognize ‘the rule’, and form a general device to add the ‘ed’ suffix to each verb form
  - Forms do not need to be memorized anymore, but this leads to overgeneralization
  - Finally, they distinguish which forms can be generated by the rule, and which must be stored as exceptions

A Symbolic Account: Dual-Route Model



- Errors result from the transition from rote learning to rule-governed
- Recovery occurs after sufficient exposure to irregulars
  - More frequency results in increased ‘strength’
- Prediction: faster recovery for frequent irregulars

Learning the Rule

- This model requires two qualitatively different types of mechanisms
- It accounts for the U-shaped curve and the observed dissociation
  - Children make mistakes on irregular forms only
- No explicit account of how the rule is learned
- Perhaps the notion of inflection is innately specified, and need not itself be learned:
  - The inflectional mechanism is triggered by the environment or maturation
  - The language specific manifestation must be learned

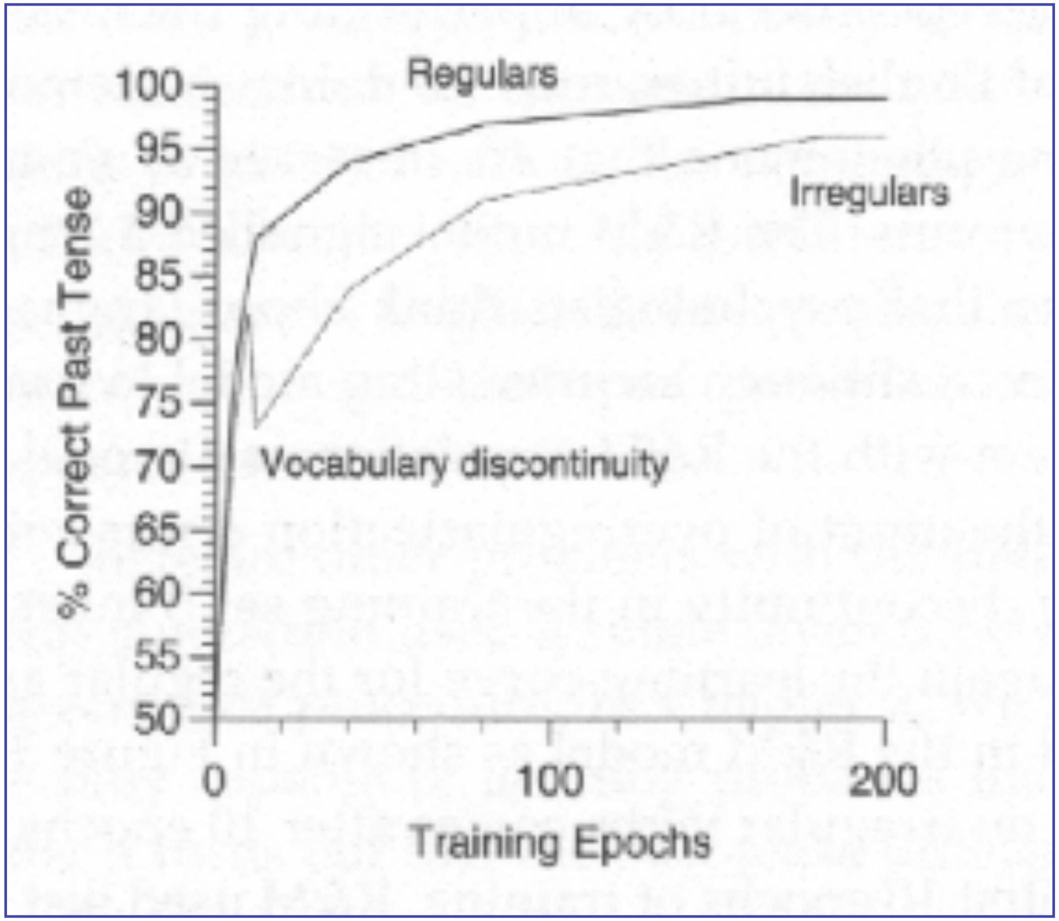
Criticisms

- Early learning tends to be focussed on irregular verbs
- Irregular sub-classes (hit, sing, ring) might lead to incorrect rule learning
  - These do occur, but typically late in learning
  - How are ‘good’ rules distinguished and selected?
- English is unusual in possessing a large class of regular verbs (only 180 irregulars)
  - Only 20% of plurals in Arabic are regular
  - Norwegian has 2 regular forms for verbs: 3-route model ?

Rummelhart and McClelland (1986)

- A single-layer feed-forward network (perceptron)
  - Input: a phonological representation of the stem
  - Output: a phonological representation of the past tense
- Training:
  - First trained on 10 high frequency verbs, then on 420 (medium frequency) verbs (80% regular)
  - Early in training, shows tendency to overgeneralize
  - End of training, exhibits near perfect performance
  - Generalized reasonably well to 86 low frequency verbs

Rummelhart and McClelland (1986)

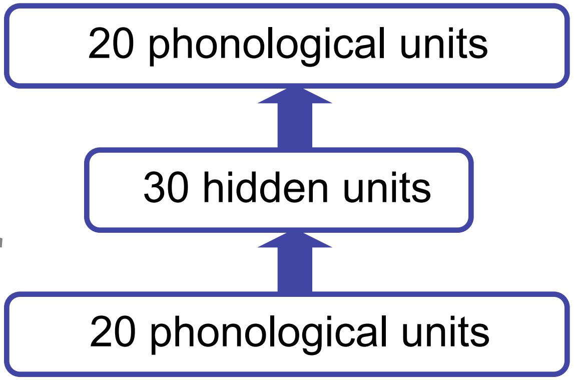


Performance of R&M (1986)

- Criticisms:
  - U-shape performance depends on sudden changes from 10-420 in the training regime
  - Most of the 410 new verbs are regular, overwhelming the network and leading to overgeneralization
- Justification: children do exhibit vocabulary spurt at end of year 2
  - But errors typically occur at end of year 3
  - Vocabulary spurt is mostly due to nouns

Plunkett and Marchman (1993)

- A standard feedforward network with one hidden layer



- Initially, the model is trained to learn the past tense of 10 regular and 10 irregular verbs
- Training proceeds using the standard backprop algorithm, in response to error between actual and desired output
  - Is this plausible?

Properties of P & M

- Highly sensitive to training environment:
  - Onset of overgeneralization is closely bound to a 'critical mass' of regular verbs learned by the child
  - Requires more training on arbitrary irregulars (go/went), which are highly frequent in the language
  - More robust for no-change verbs (hit, put) which are more numerous (type) and less frequent (token)
- Models the frequency × regularity interaction:
  - Faster reaction time for high frequency irregulars than low frequency ones
  - No advantage for regulars

Criticism

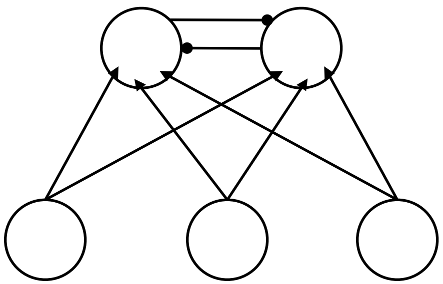
- Pinker & Prasada argue that the (idiosyncratic) statistical properties of English help the model:
  - Regulars have low token frequency but high type frequency: facilitates generalization
  - Irregulars have low type frequency but high token frequency: facilitates rote learning mechanism
- They argue no connectionist model can accommodate default generalization for a class which has both low type and token frequency
  - Default inflection of plural nouns in German appear to have this property

Competitive Networks: Overview

- Operation:
  - Given a particular input, output units compete with each other for activation
  - The winning output unit is the one with the greatest response activation
- During training:
  - Connections to the winning unit from the active input units are strengthened
  - Connections from inactive units are weakened
- Training is unsupervised
  - The network will categorize inputs based on similarity
  - Learns to capture statistical properties of input space

Architecture of Competitive Networks

- A simple network:
  - Inputs are fully connected to outputs by feed-forword connections
  - Outputs may be connected to each other by inhibitory connections

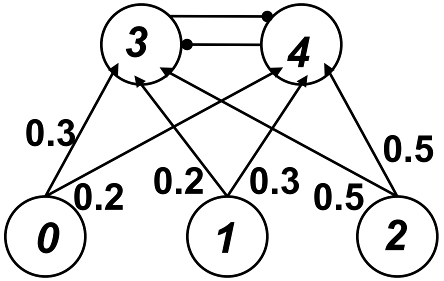


$$\text{netinput}_i = \sum_j a_j w_{ij}$$

- Outputs compete until only one remains active
  - Or, simply the unit with highest activation wins
  - Active units force other units to become inactive

An example

Consider the following network:

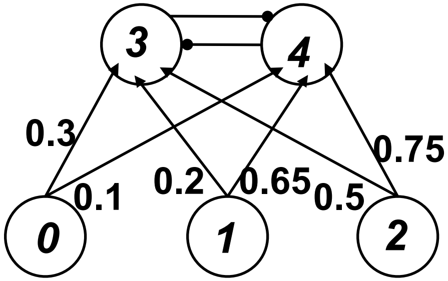


- Input pattern: (0 1 1)

  $\text{netinput}_3 = (0 \times 0.3 + 1 \times 0.2 + 1 \times 0.5)$
  $= 0.7$

  $\text{netinput}_4 = (0 \times 0.2 + 1 \times 0.3 + 1 \times 0.5)$
  $= 0.8$

- Since unit_4 wins, no changes in connections to unit_3
- For connections to unit_4:
  - $\Delta w_{ij} = \varepsilon\, (a_j - w_{ij})$
  - $\Delta w_{ij} = 0.5\ (0.0{-}0.2\ \ 1.0{-}0.3\ \ 1.0{-}0.5)$
  - $\Delta w_{ij} = 0.5\ ({-}0.2\ \ 0.7\ \ 0.5)$
  - $\Delta w_{ij} = ({-}0.1\ \ 0.35\ \ 0.25)$



Overall Behaviour

- Net input to an output unit is greatest when its weight vector is most similar to the input vector
- Training makes the weight vector for a particular winning unit more similar to the input pattern
- The weight vector for a particular output unit learns to respond to similar input patterns
  - The learned weights will be an average of the patterns, based on the frequency of presentation during training
- The competitive network can therefore learn to categorize similar inputs without any ‘teacher’

Summary

- Connectionism is inspired by information processing in the brain
- An input stimulus causes a pattern of activation on the first layer
  - Activations are then propagated through the network
  - Weights determine the influence of unit on each other
  - The output is the pattern of activation on final layer
- Learning aims to reduce the discrepancy between actual and desired output patterns of activation
  - Delta rule changes the weights of successive epochs
  - Training is complete when error is sufficiently reduced