COMP 40260

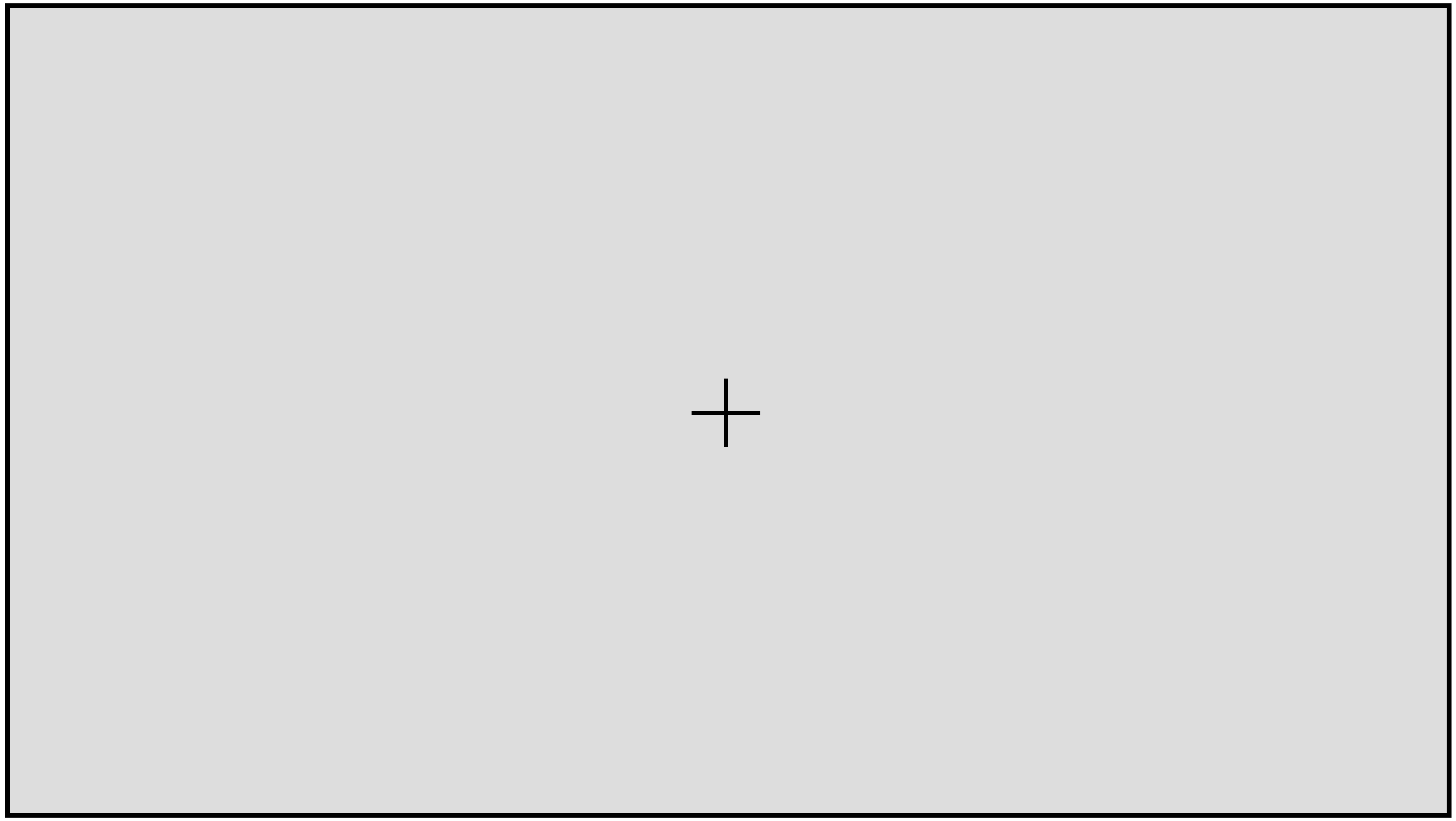
Connectionism (Artificial Neural Networks) and Dynamical Systems

Part 2

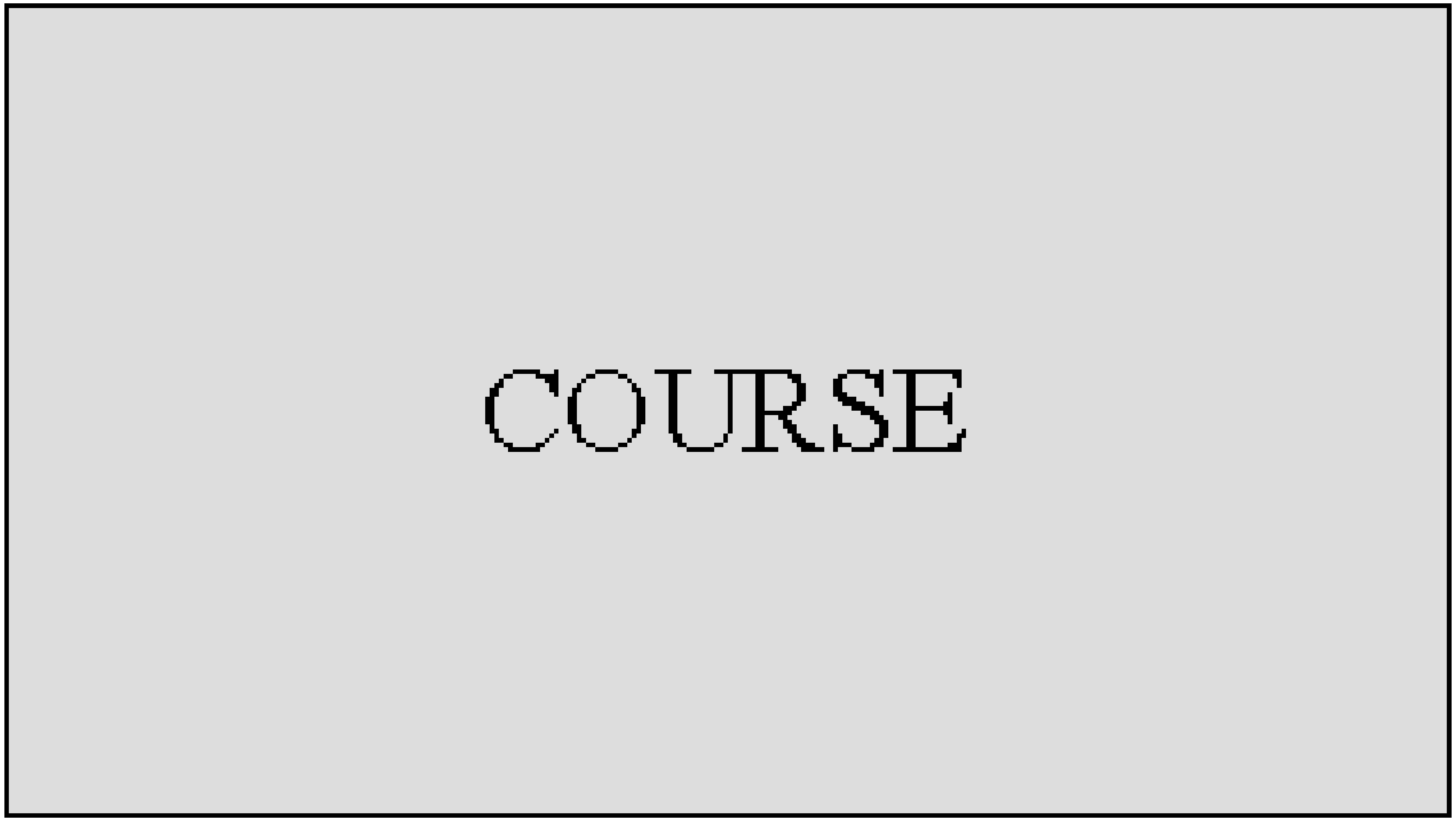
Read Rethinking Innateness, Chapters 1 & 2

Let’s start with an old neural network, created before training from data was possible.

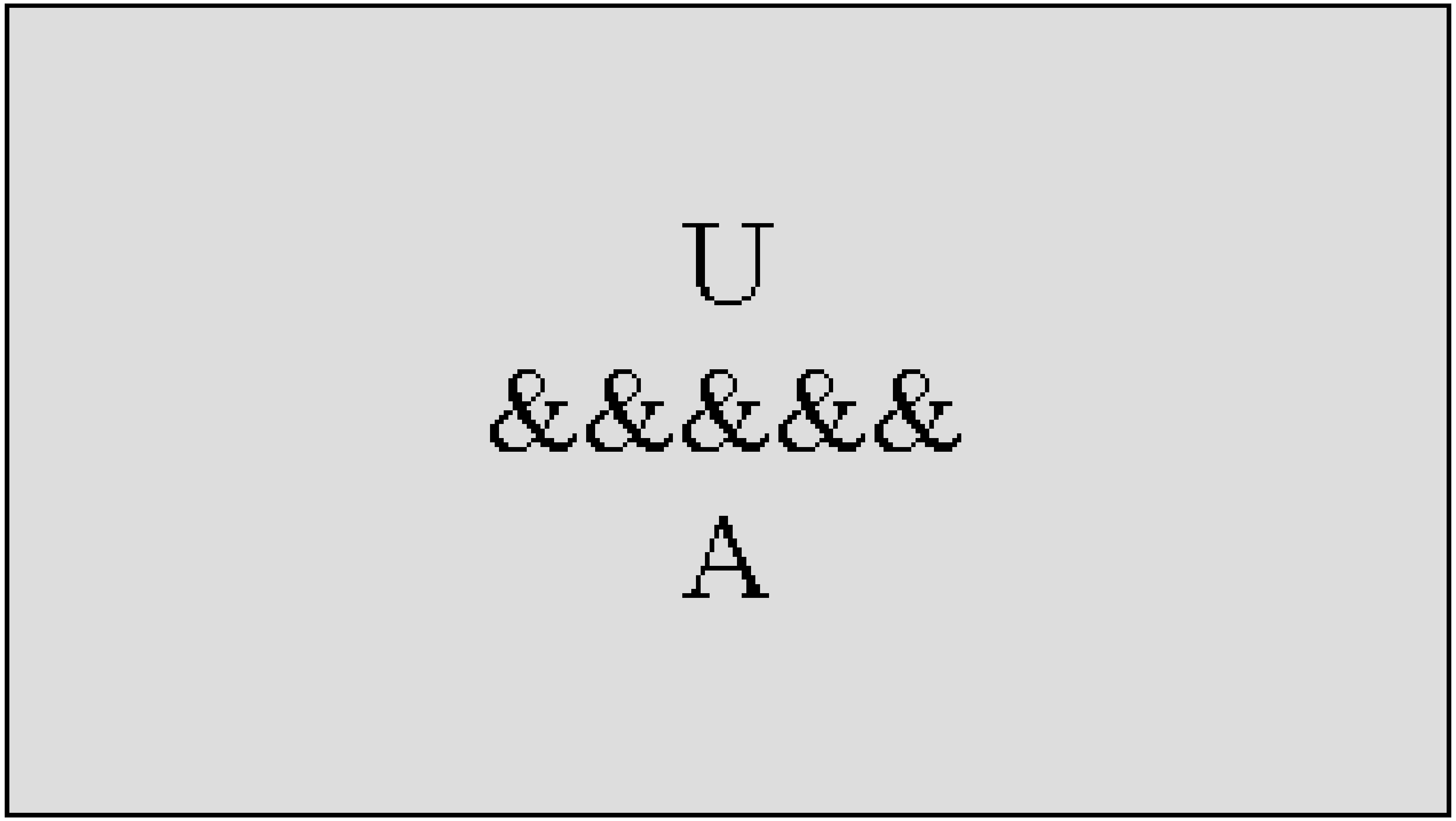
It illustrates how we might use some aspects of our network (processing times, sequence, etc) to mimic measurable behavioural variables.



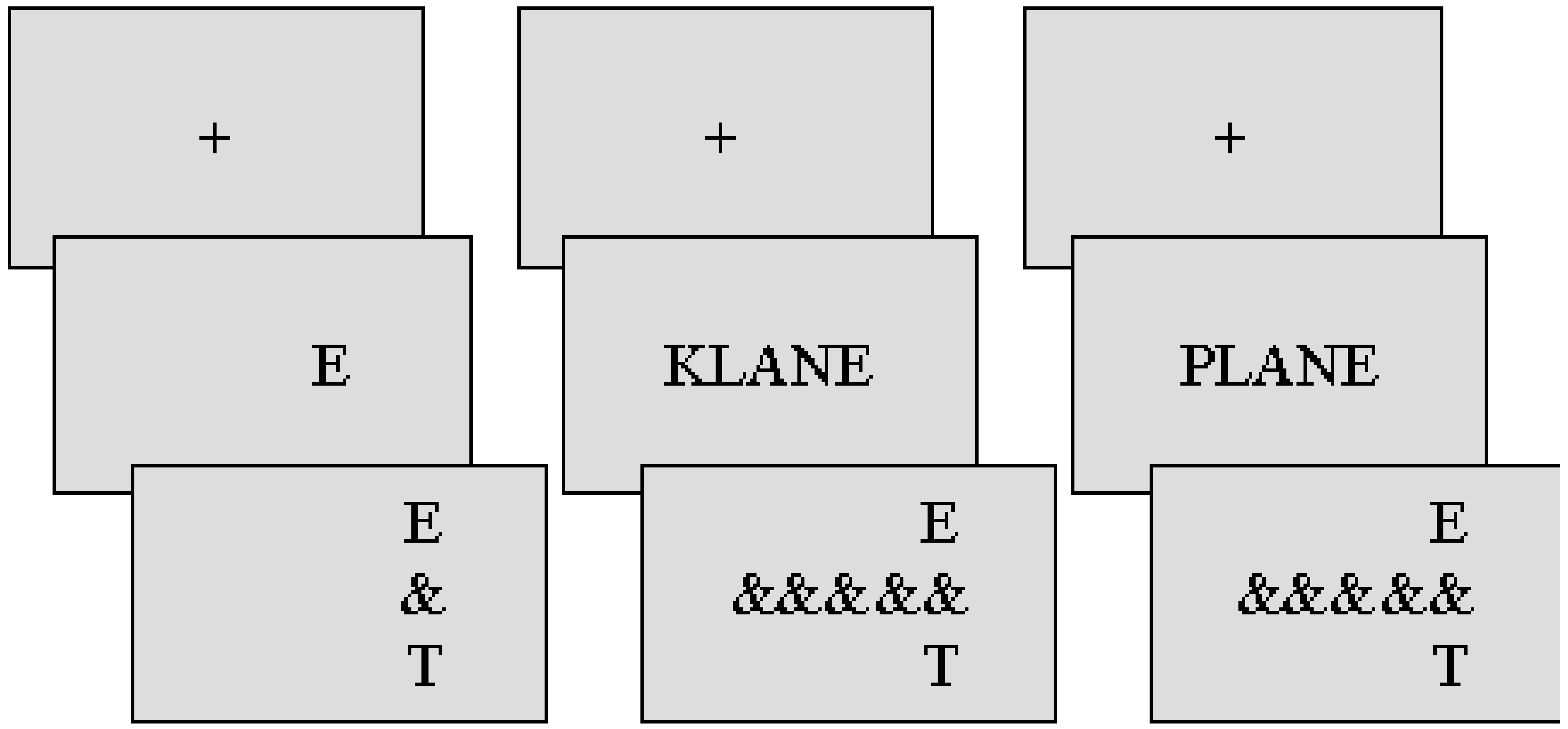
Until the participant hits some start key



Presented briefly … say 25 ms



Mask presented with alternatives above and below the target letter … participants must pick one as the letter they believe was presented in that position.



Letter only
Say 60%

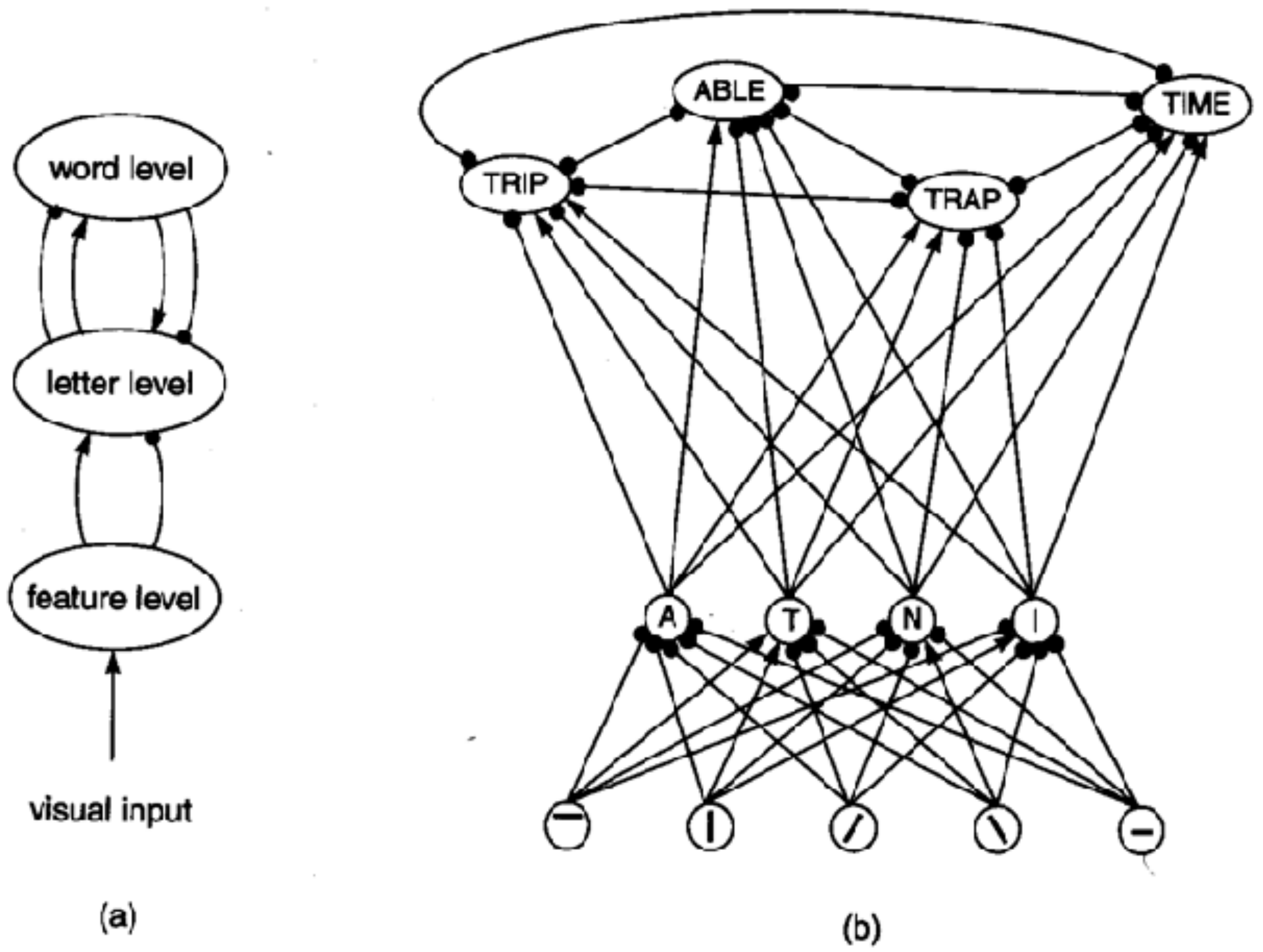
Letter in Nonword
Say 65%

Letter in Word
Say 80%

McClelland and Rummelhart’s 1981 model of the Word Superiority effect:

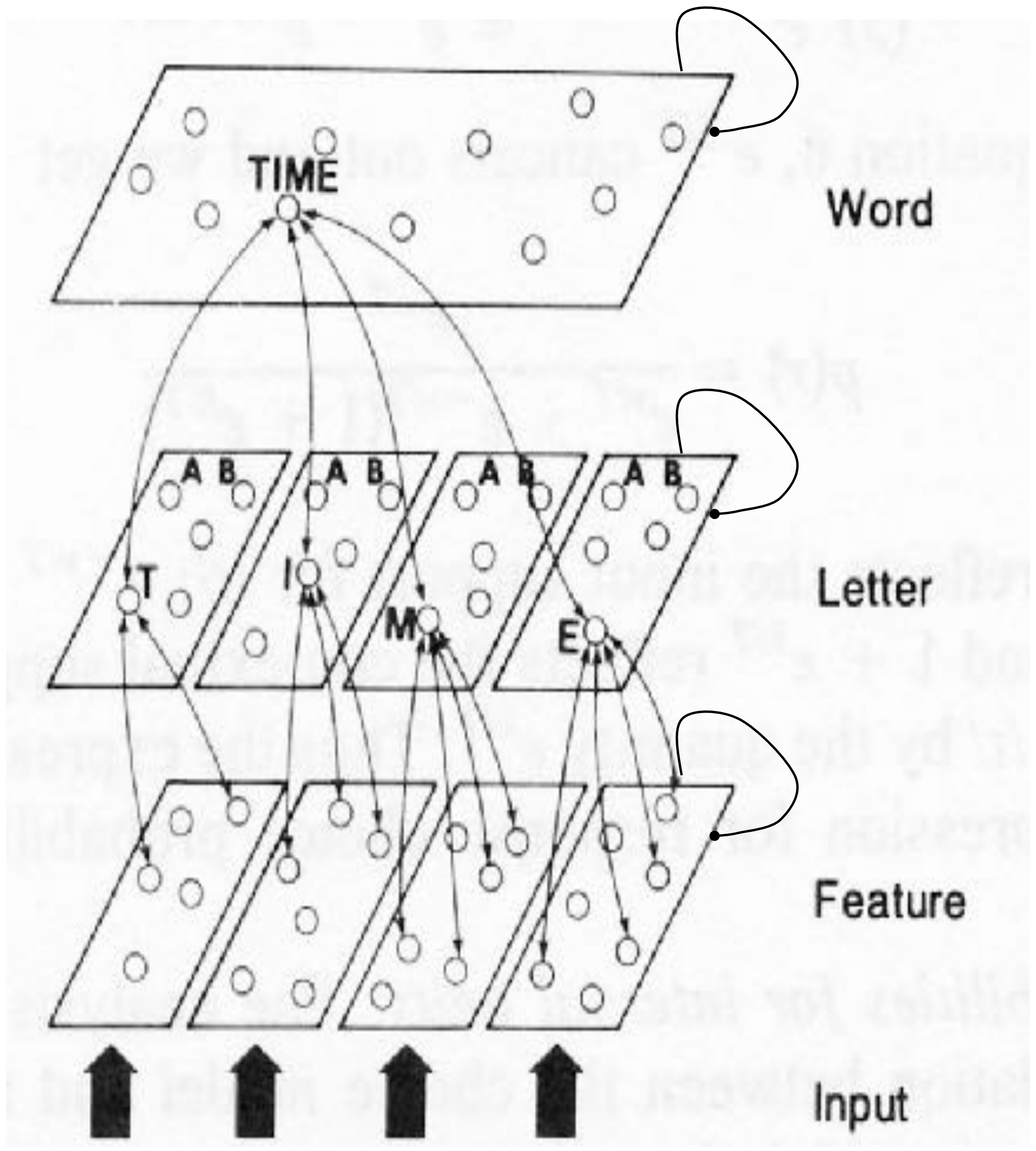
•Weights are inhibitory (dot) or excitatory (arrow)

•Weight values are hand crafted to achieve desired results



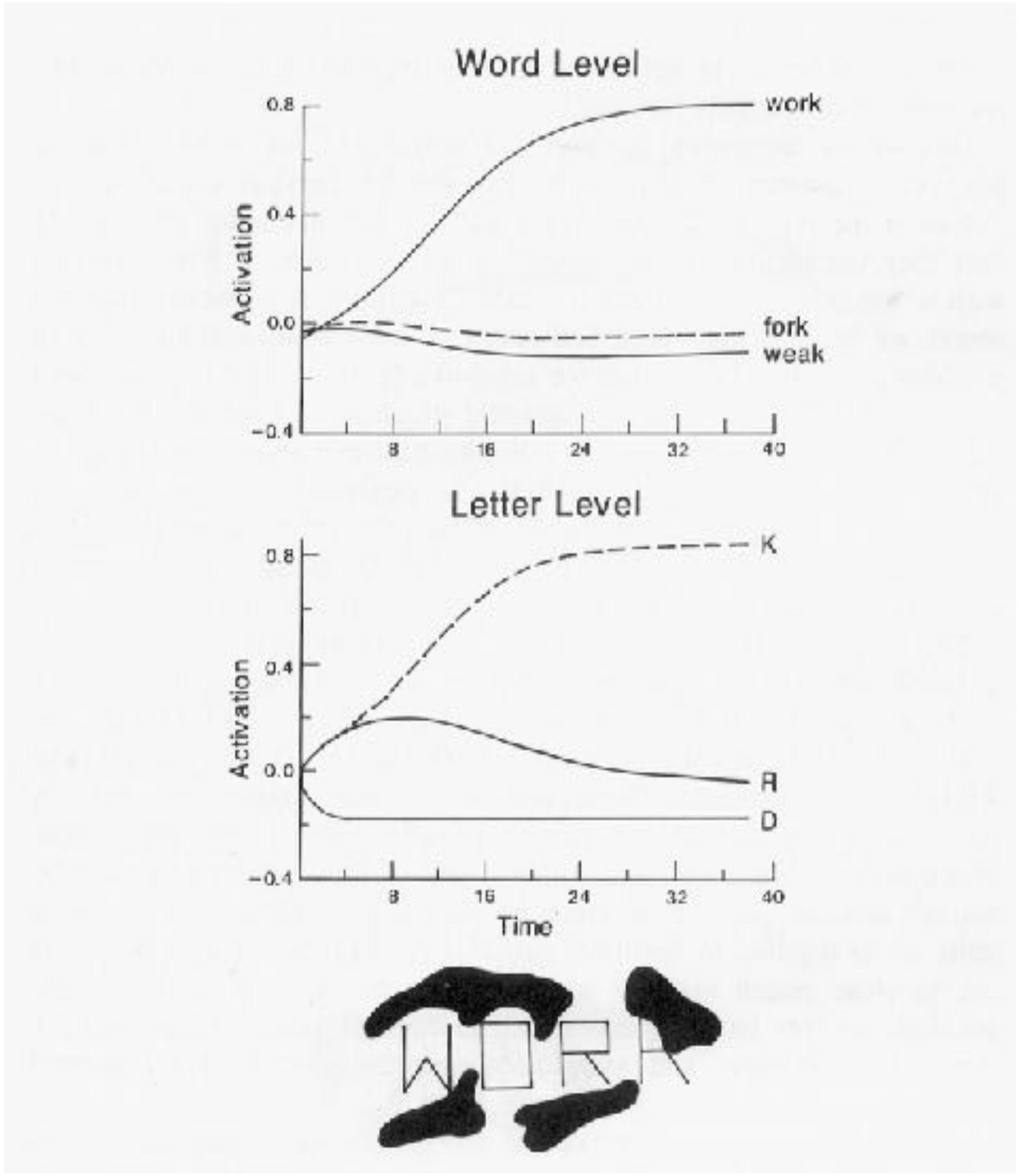
The interactive Activation Model: a Gradual Mutual Constraint Satisfaction Process

- Units represent hypotheses about the visual input at several levels and positions.
  - Features
  - Letters
  - Words
- Connections code contingent relations:
  - Excitatory connections for consistent relations
  - Inhibitory connections for inconsistent relations
  - Lateral inhibition for competition among mutually inconsistent possibilities within levels.
- Connections run in both directions
  - So that the network tends to evolve toward a state of activation in which everything is consistent.

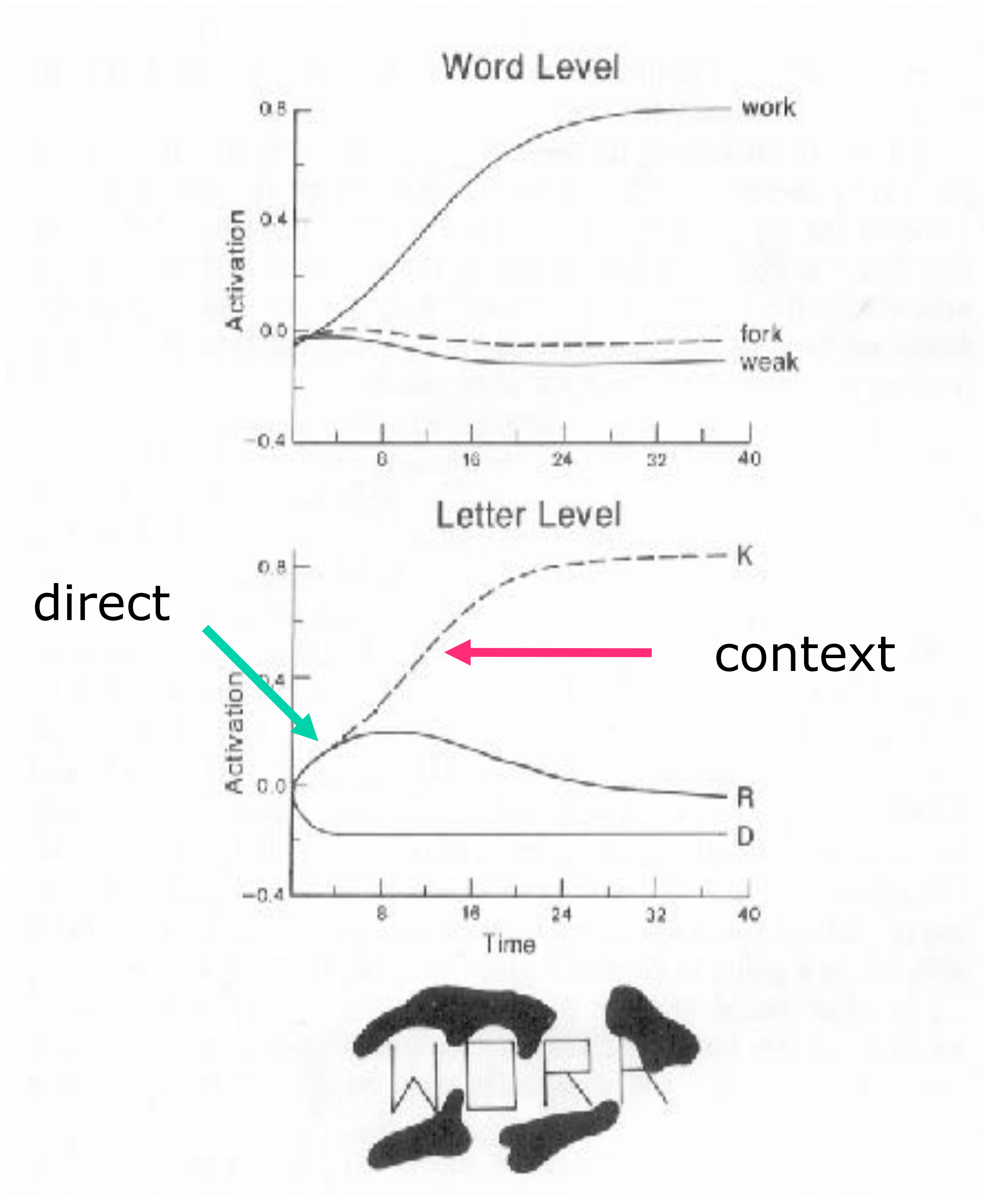


Interactive Activation *Simultaneously* Identifies Words and Letters

- Stimulus input comes first to letter level, but as it builds up, it starts to influence the word level.
- Letter input from all four positions makes *work* the most active word unit (there is no word *worr*).
- Although the bottom up input to the letter level supports K and R equally in the fourth letter position, feedback from the word level supports K, causing it to become more active, and lateral inhibition then suppresses activation of R.



- The patterns seen in the physiology are comparable to those seen in the interactive activation model in that the effect of direct input is manifest first, followed somewhat later by contextual influences, presumably mediated in the physiology by neurons sensitive to the overall configuration of display elements.



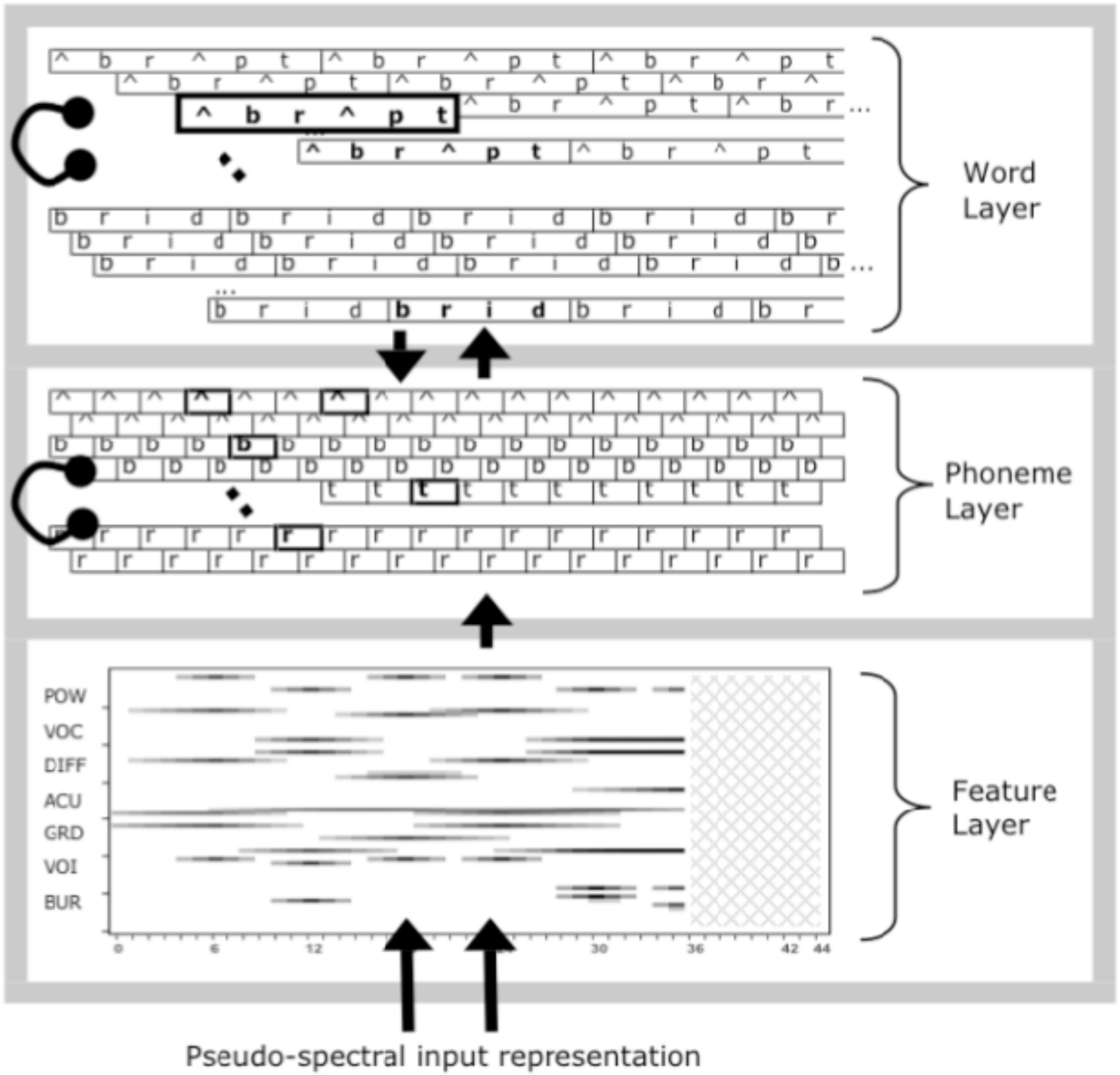
Web app: http://www.psychology.nottingham.ac.uk/staff/wvh/jiam/

There is a javascript implementation of this model you might like to play with. Stick to those aspects that you learned in class, before exploring fine detail.

Web app: http://www.psychology.nottingham.ac.uk/staff/wvh/jiam/

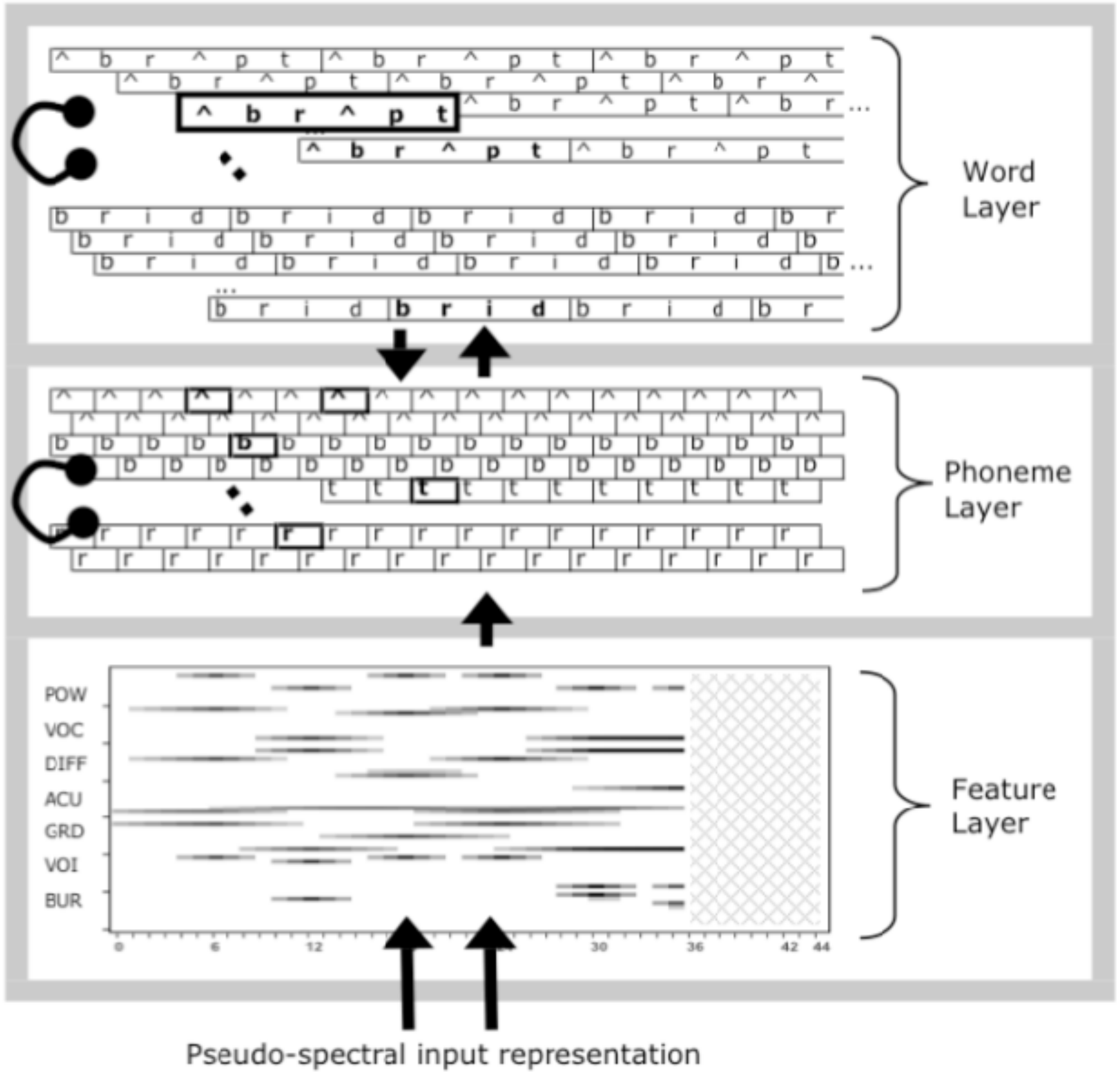
McClelland and Elman: TRACE: model of spoken word recognition (1984)

Mapping from sound input to word output, via phonemes



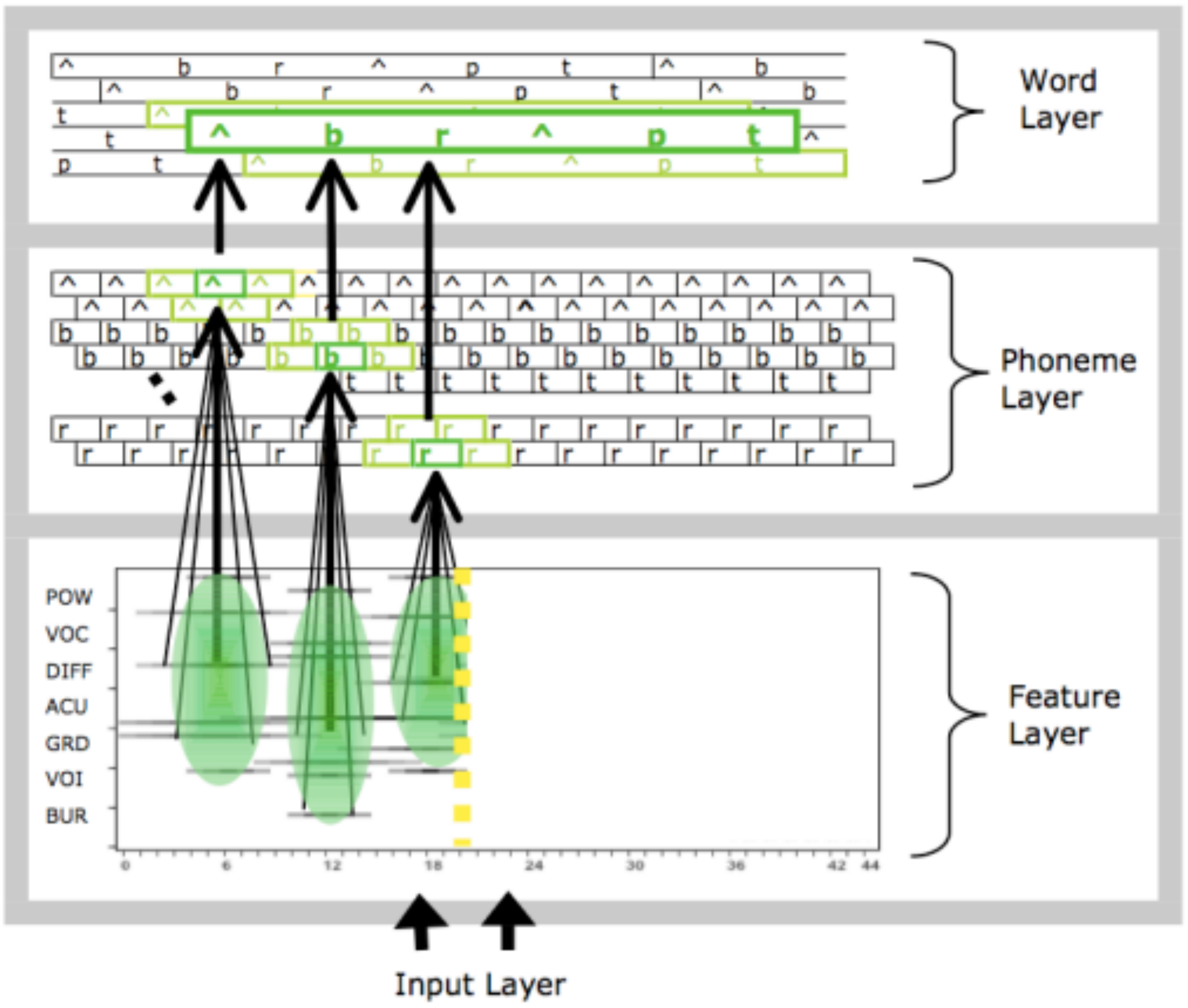
Trace

Inhibition only within layers
Sequential input
Features extracted from speech



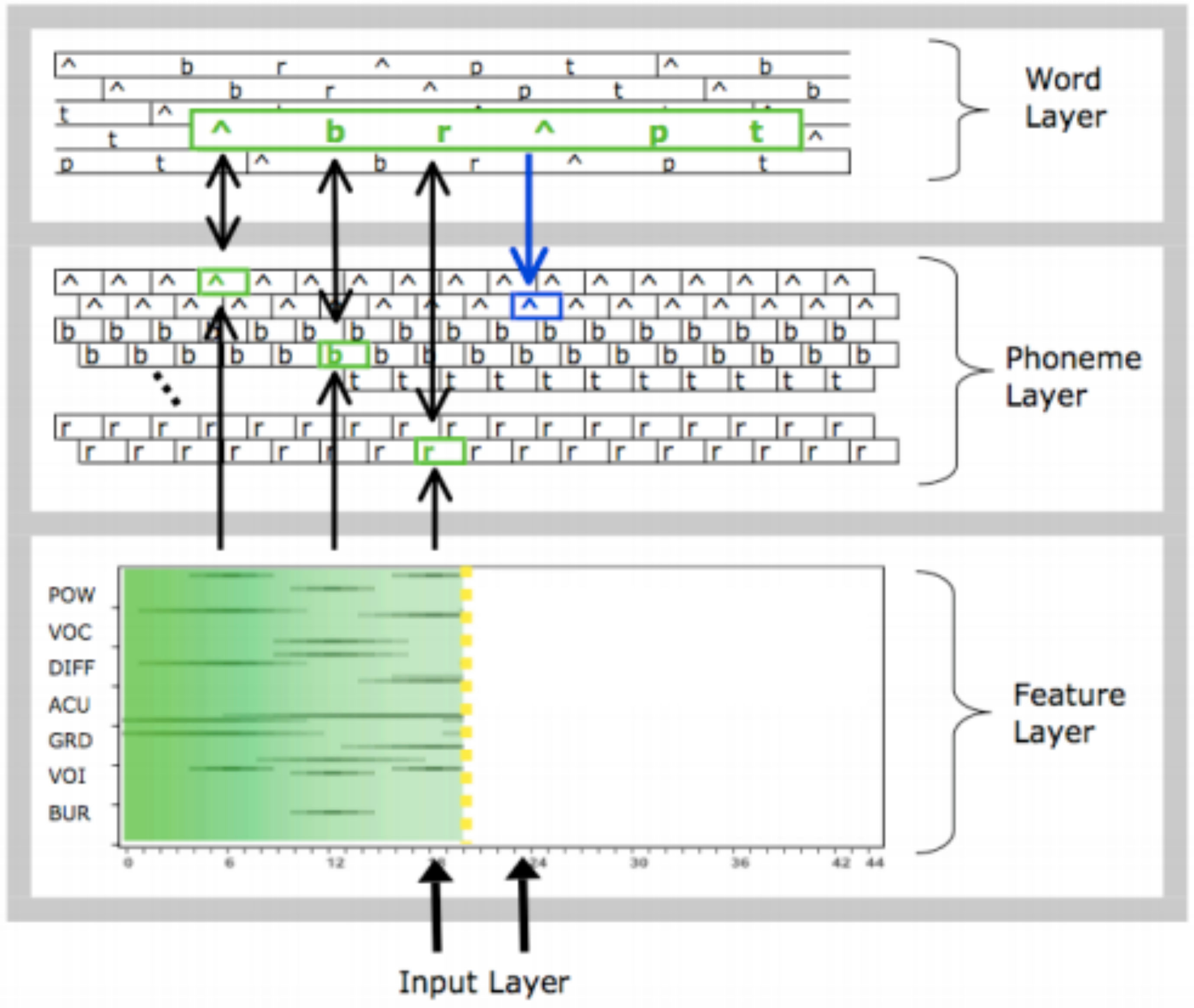
Activation of phonemes & words

- Bottom-up (**feed-forward**) excitation causes initial activation
- Excitation increases with temporal overlap and similarity



Lexical-to-phoneme feedback

- Word units feed back to constituent phonemes
- Constituent phonemes that were not "heard" (earlier or later) via lexical **feedback**



A Java-based implementation of TRACE is available at http://magnuson.psy.uconn.edu/jtrace/ if you feel like playing with it.

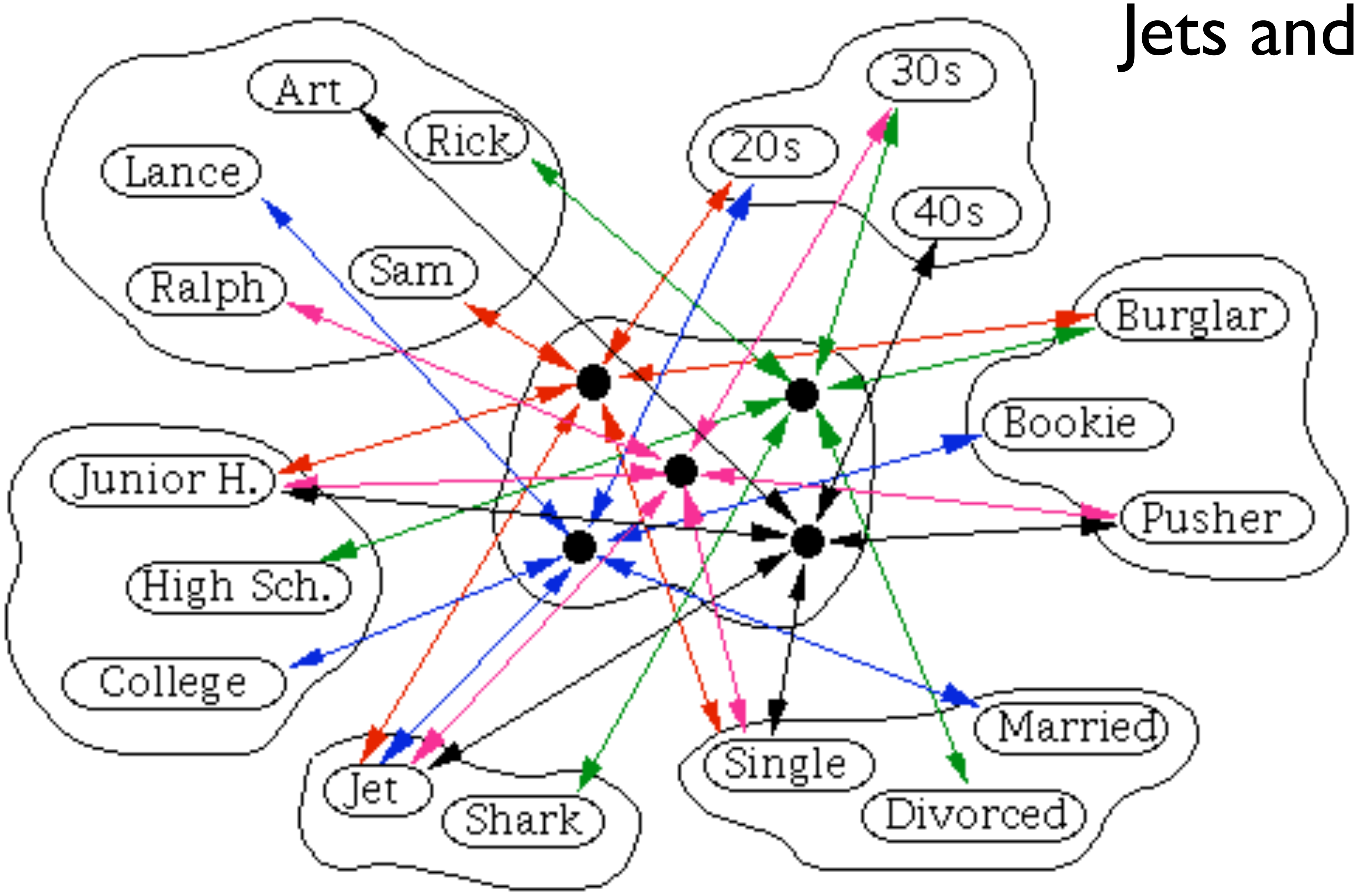
The wikipedia page is also fairly good in the overview it provides. Don’t expect much depth to the discussion though

https://en.wikipedia.org/wiki/TRACE_(psycholinguistics)

Both models are examples of

Interactive Activation Networks

Jets and Sharks



In Lab 2, we will be playing with an implementation of the Jets and Sharks network.

Further detailed information is available, e.g.

http://staff.itee.uq.edu.au/janetw/cmc/chapters/IAC/

http://www.cs.indiana.edu/~port/brainwave.doc/IAC.html

Learning (Take 1)

- The Word-Superiority Network and Trace used *hand crafted* weights.
- It would be nice to learn the appropriate values from data
- Why might this be good?
- Why might this be bad?

Hebbian Learning

Hebbian learning

Hebb's Postulate: When an axon of cell A...excites cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells so that A's efficiency as one of the cells firing B is increased

$$\Delta w_{ij} = \eta a_i a_j$$

- Learns pairwise correlations (and nothing else)
- Can generate intriguing structure in large multilayered networks
- One of a family of **unsupervised learning** techniques

- Hebbian learning find the *thing in the world* that most reliably activates the unit, and *tunes* the unit to like that thing even more!

Hebbian Learning

- Units that Wire together, Fire together
- It is an associative learning method
- Similar things are stored in similar ways.

Learning rate

$$\Delta w_{ij} = \eta a_i a_j$$

Change in
weight to unit *i*
from unit *j*

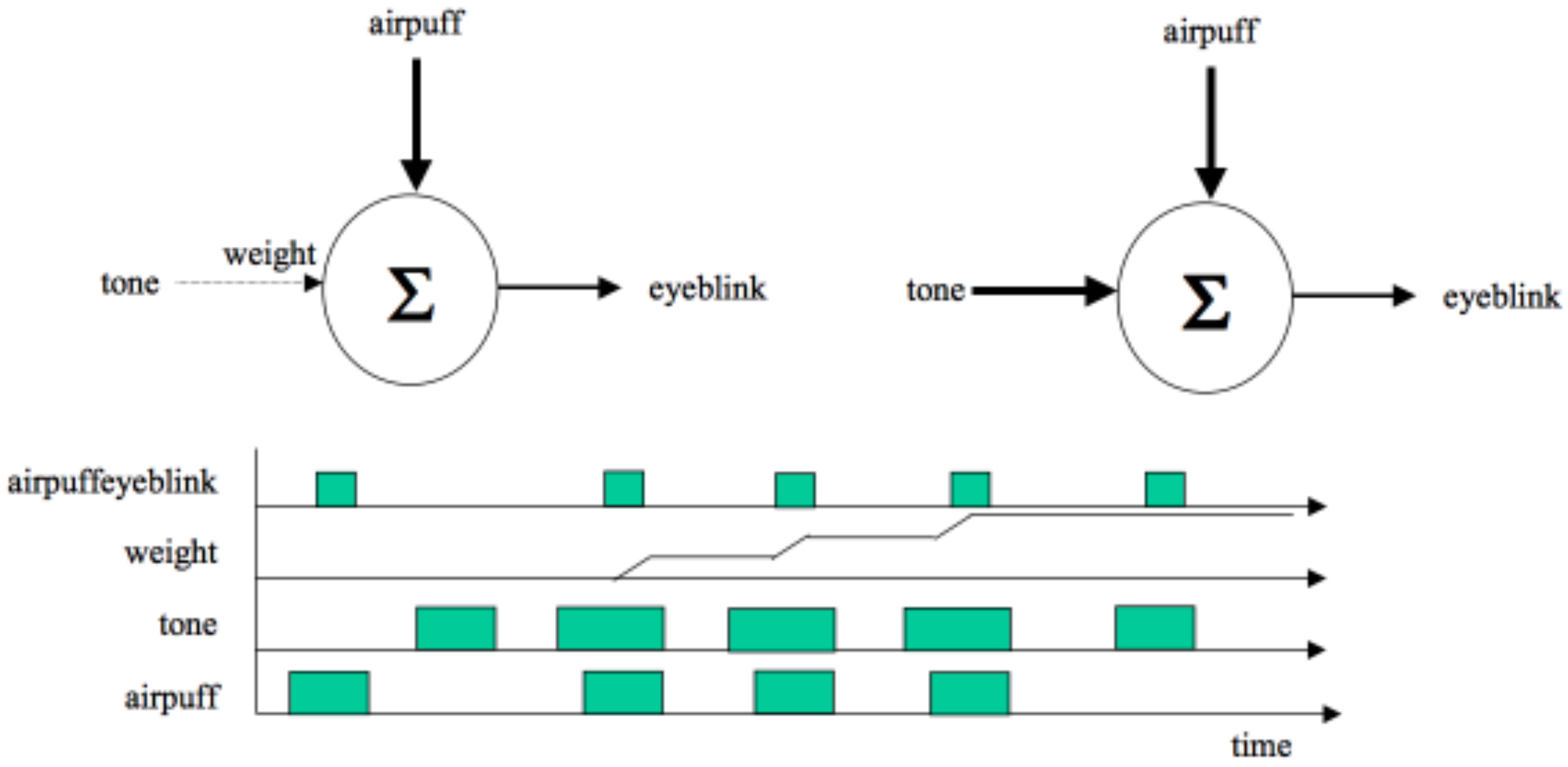
Activation of unit *i*

Issues with Hebbian Learning

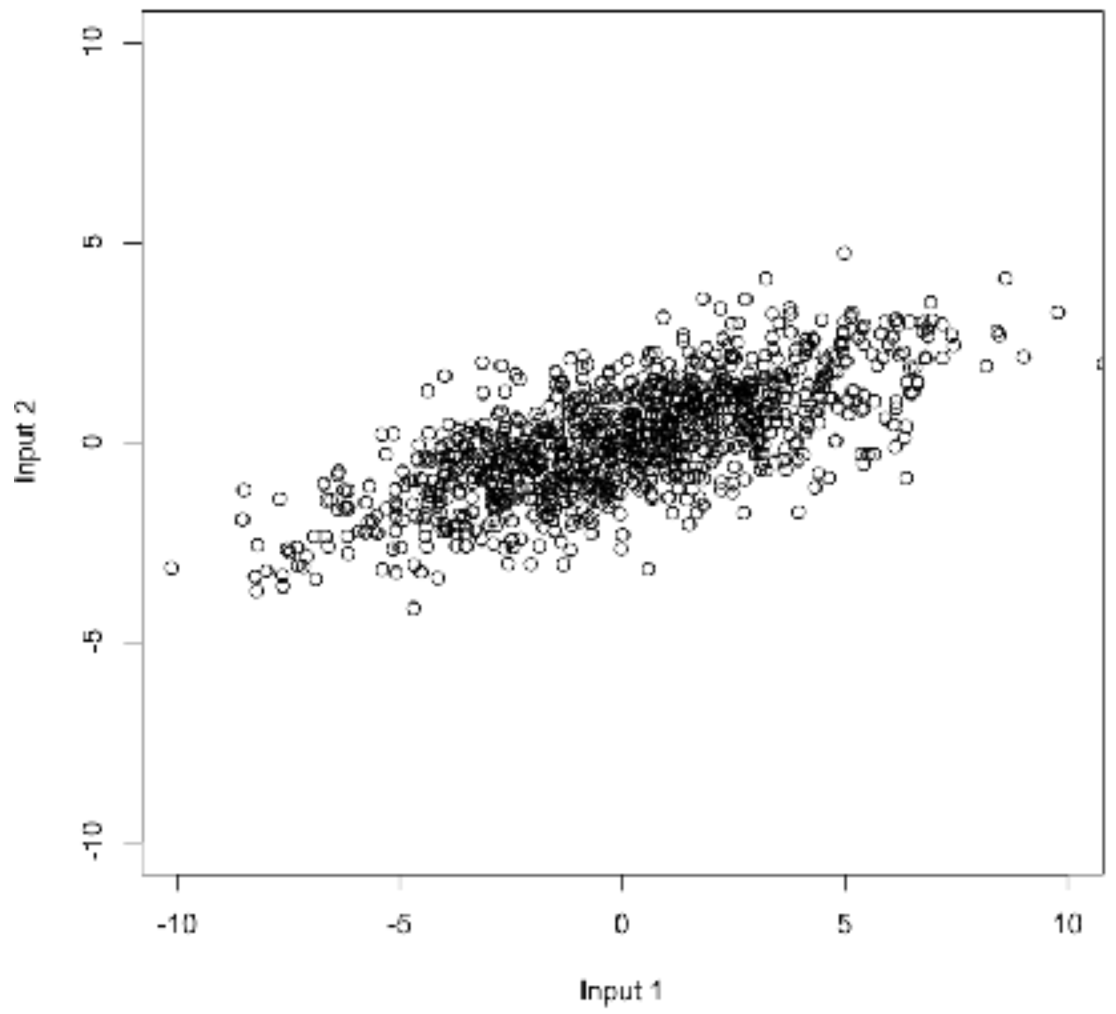
$$\Delta w_{ij} = \eta a_i a_j$$

- Local knowledge
- Learns *correlations*
- Unstable (in simple form)
- Basis for most unsupervised learning techniques
- Simple..... Adaptable...

Classical conditioning, implemented in the style of Hebb.



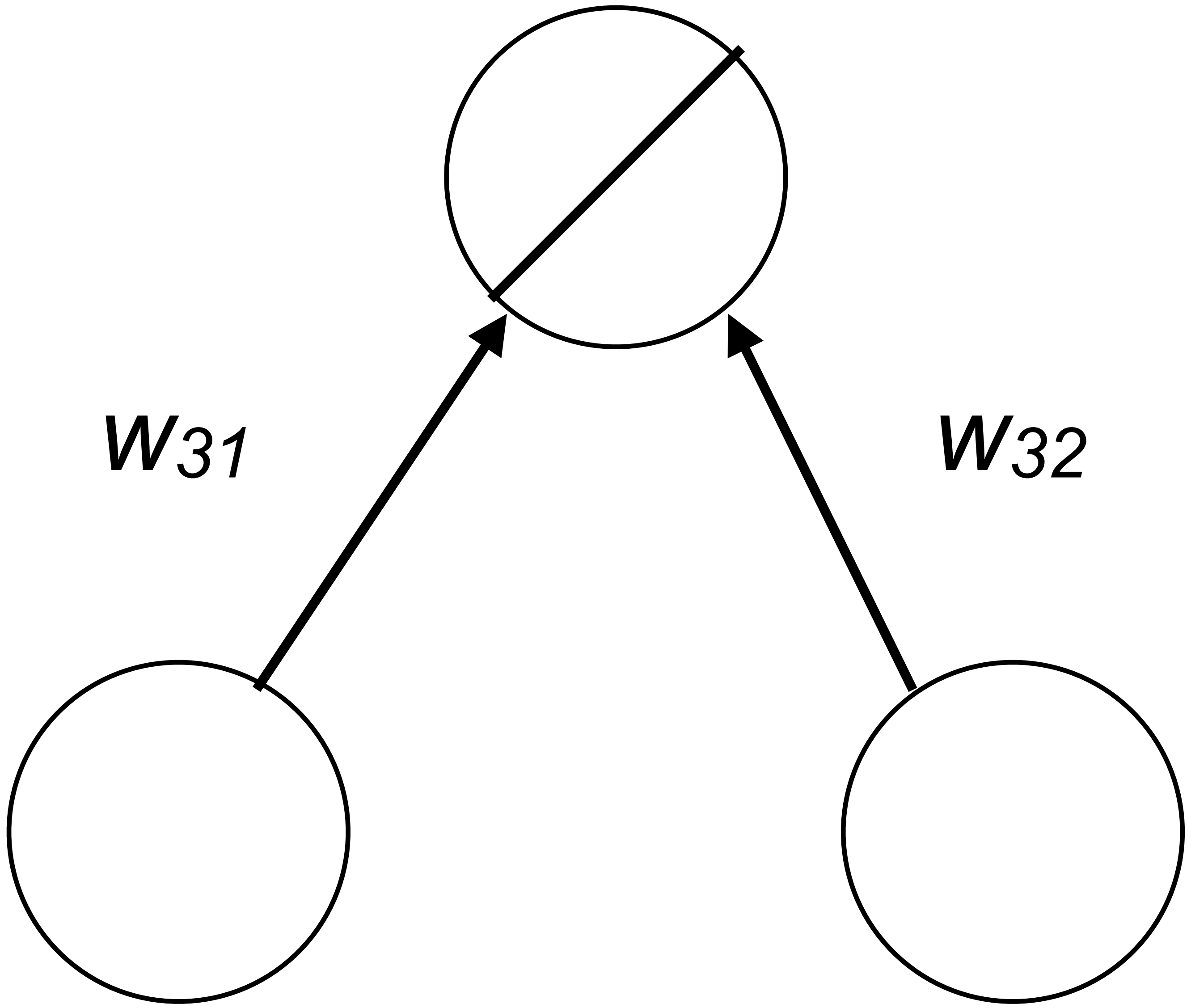
These data have been generated to illustrate the simplest Hebbian learning.



Notice that the 2 inputs are correlated, and Input 1 is typically about twice the size of Input 2

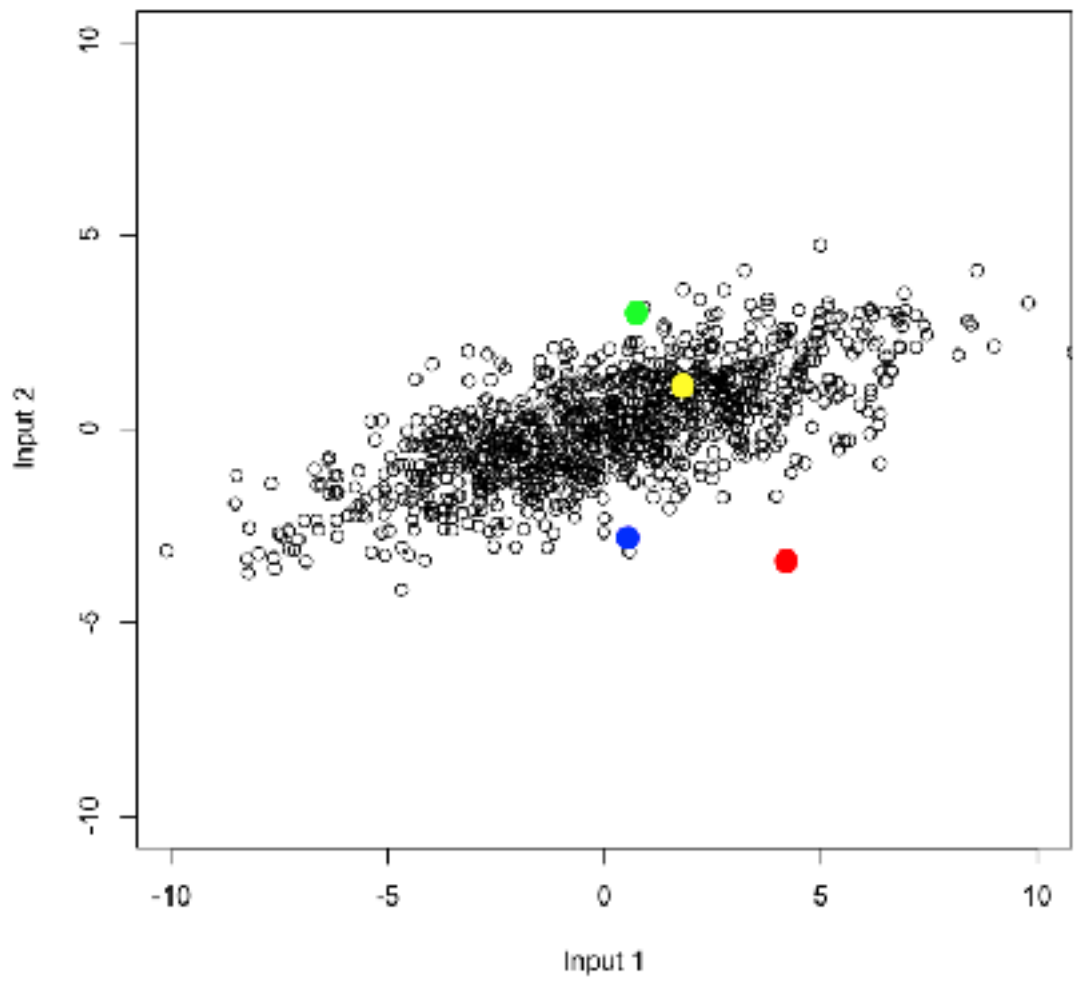
There are 2 weights.

We set these to random values.



Let's do that 4 different times. Each time, we pick 2 random numbers. We can plot these in the same 2-D space as the patterns.

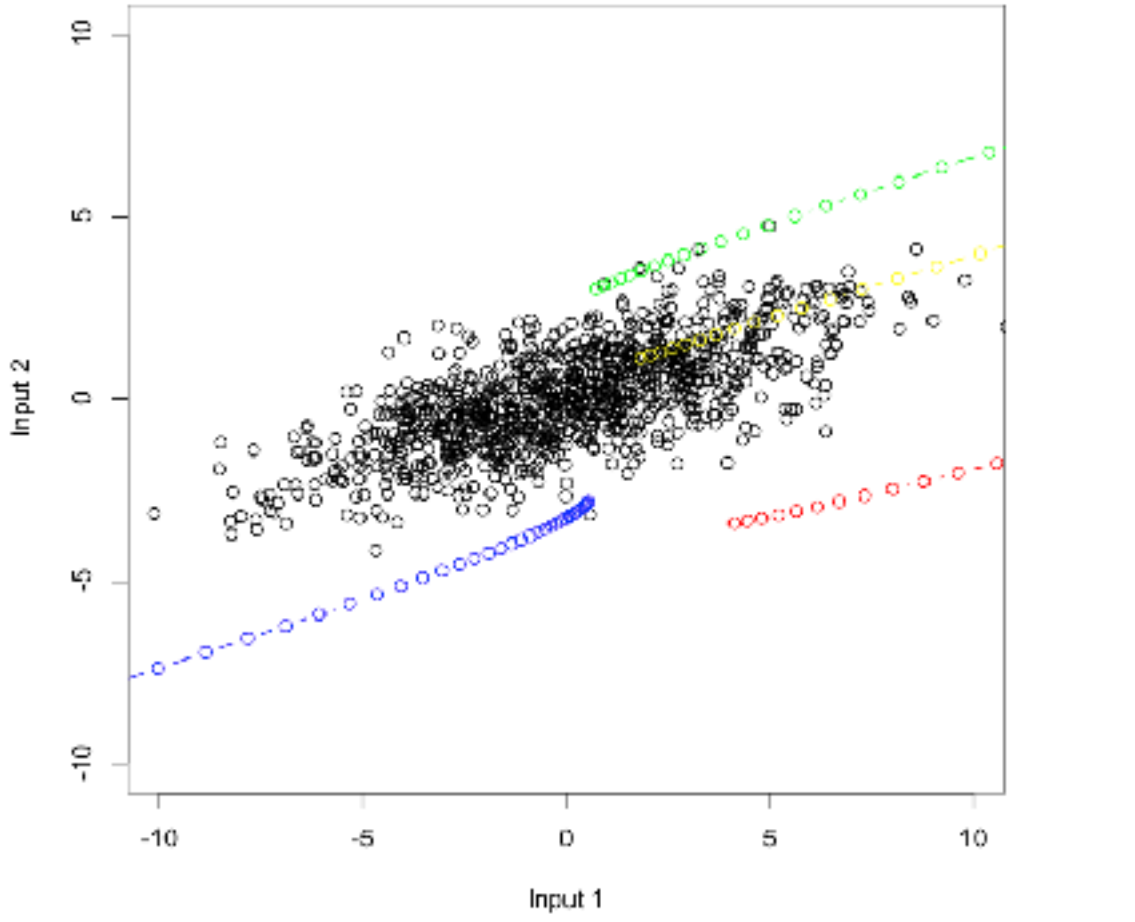
```
 1.9669226  0.87938114
-0.4060477 -0.29767872
-1.4883600  1.46278894
 2.5347912  2.11150930
-0.7017127 -2.00009350
 0.3338588 -0.22183585
 0.8340077  0.37764276
 3.8722375 -0.04791973
 3.4633267  0.72791296
 7.4028873  2.66630372
-1.4012518 -0.89913399
-4.0869216 -0.13160496
-4.3041923 -1.20494477
-1.3574328 -0.29008036
 3.0573006  0.29865487
 2.2056722  1.29260299
-4.0475503 -1.03311485
 2.7027344  1.13994032
-2.8312812 -0.69010906
 6.4583367  3.02454072
```



Here are the first 20 of 1000 input patterns. Now we take each pattern in turn, and calculate how much we would change the weights, based on that pattern alone.

We keep track of this for 1000 patterns, then we change the weights by a small fraction of this amount (the size is given by the learning rate)

Here you see the changes in the weights from 4 different starting points, as we repeat this process over and over.



Notice that it doesn't matter where we start, we always end up with weights with a ratio of about 2:1. I.e the weight values start to reflect the correlation evident in the data set.

Limits on pure Hebbian Learning

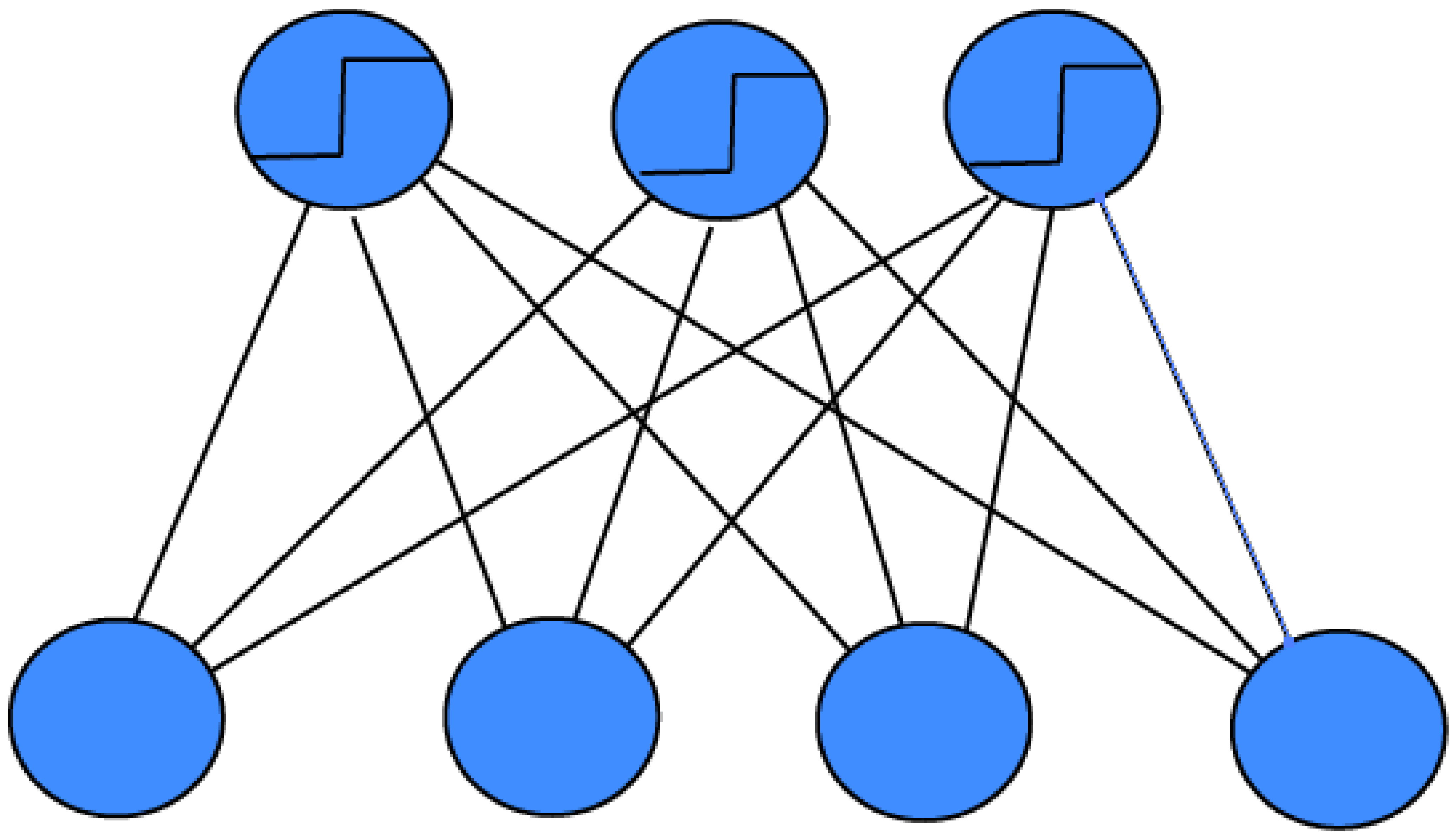
Hebbian learning learns correlations. Only.

Some means to stop unlimited growth of weights is necessary

Long-Term Inhibition needed as a counter-mechanism to Long-Term Potentiation

Behold, the PERCEPTRON!!!!

The Perceptron



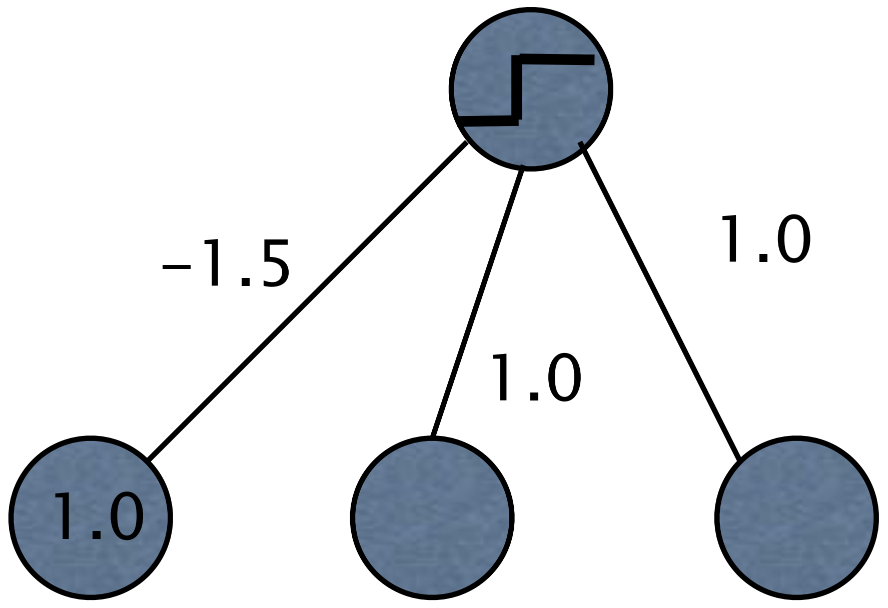
Perceptron Convergence Procedure

- Perceptron: 2-layer network with threshold activation function at the output units (+/- 1)
- Trained on a data set for which we have both input and target pairs
- Wt changes based on error at outputs.
- Wt change depends on error produced and activation coming along a weight from a given input (credit and blame algorithm)

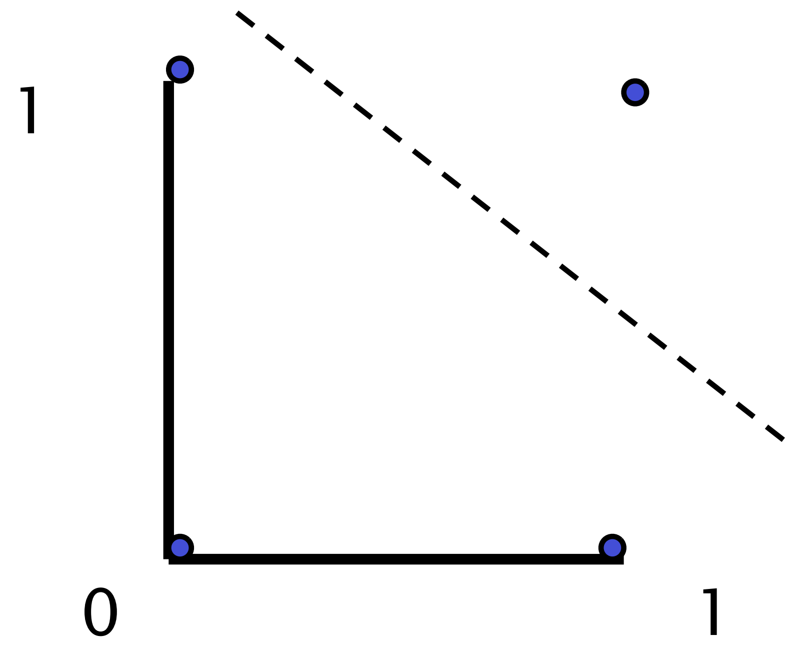
Perceptron Convergence, contd...

- PCP requires an explicit teacher
- Similar inputs yield similar outputs (cf also Hebbian Learning)
  - not a bad idea in principle
- Many problems cannot be solved with this limitation:
  - famous example: learning XOR

Learning AND

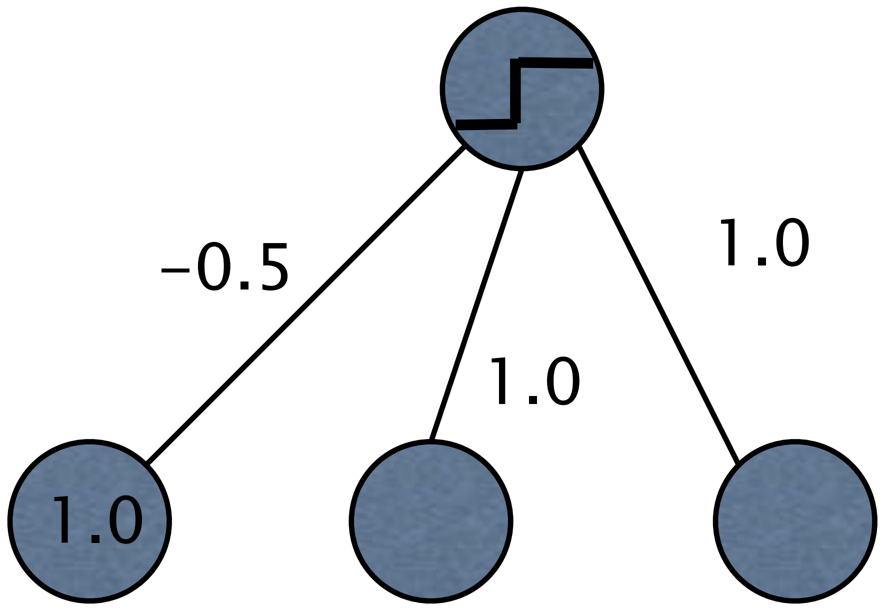


inputs	output
0 0	0
1 0	0
0 1	0
1 1	1



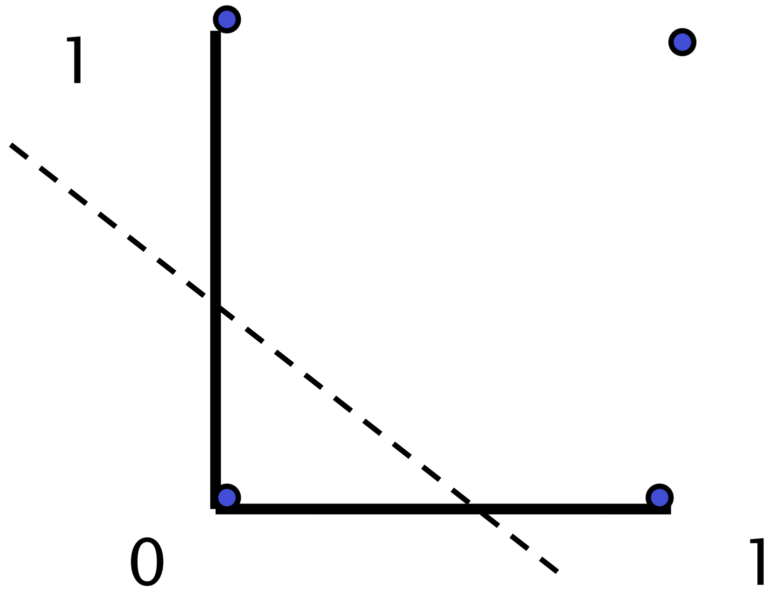
This is "input space"

Learning OR

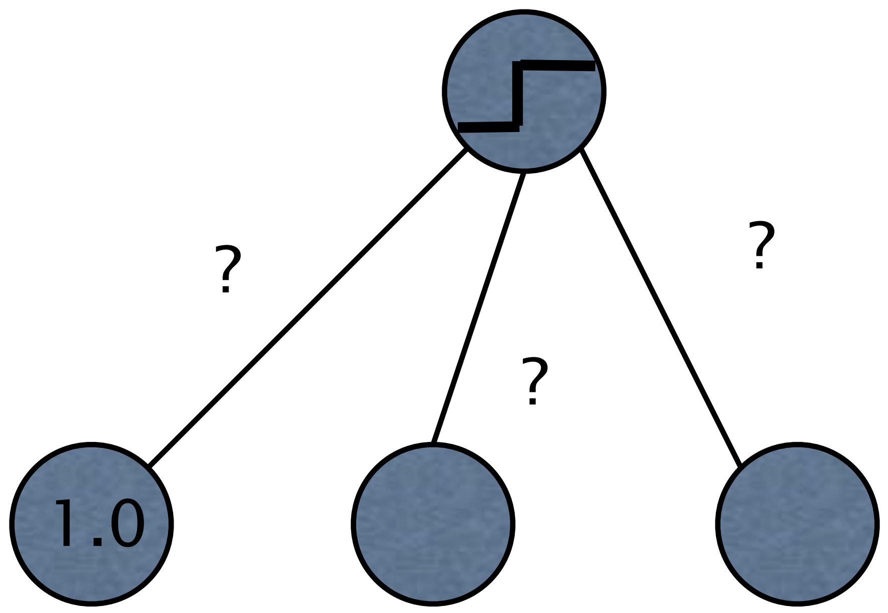


inputs	output
0 0	0
1 0	1
0 1	1
1 1	1

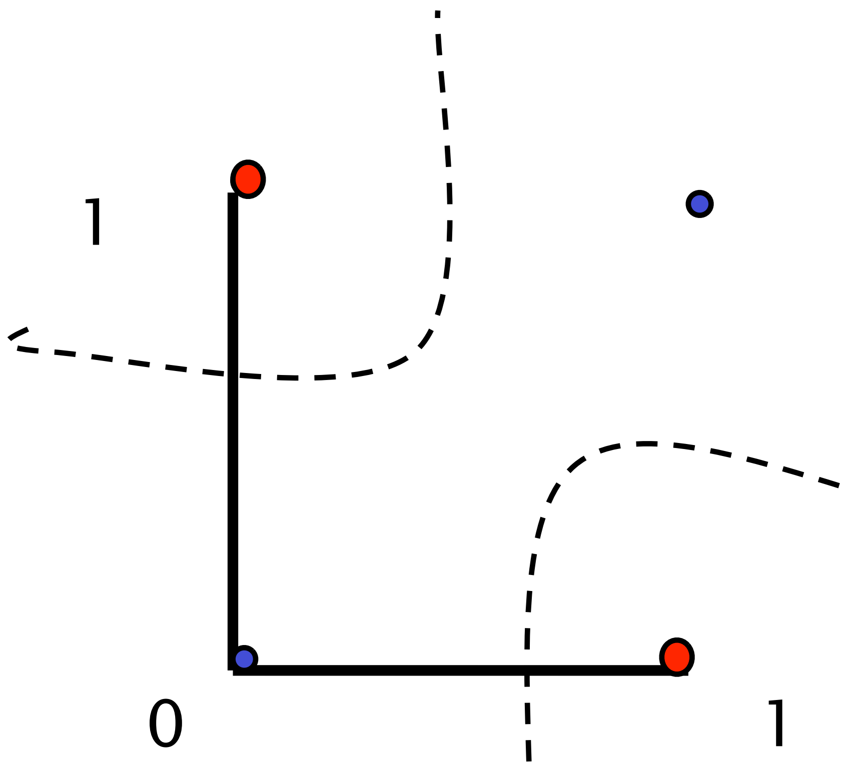
input space



Learning XOR

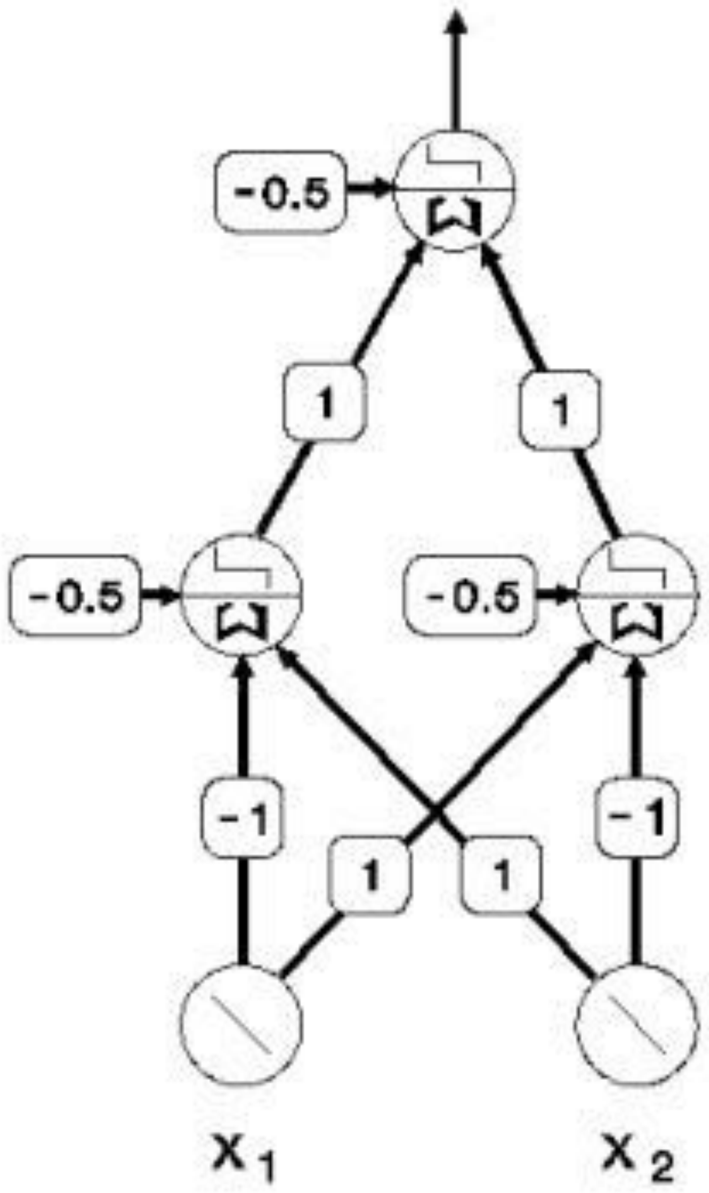


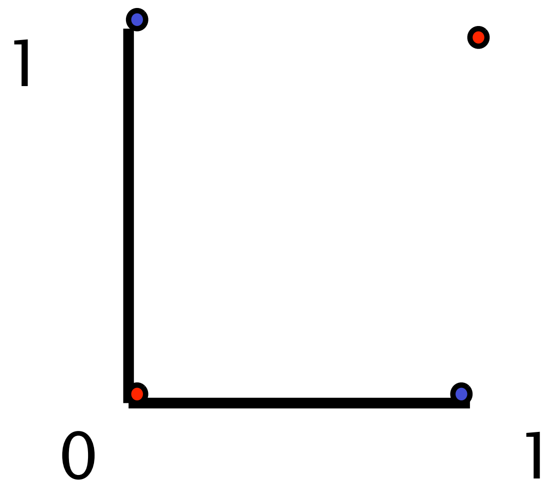
inputs	output
0 0	0
1 0	1
0 1	1
1 1	0



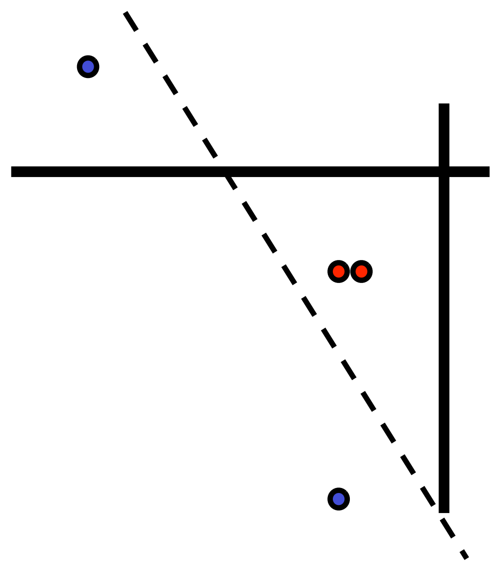
Patterns are not linearly separable in input space

Adding hidden units





input space



hidden unit space

A perceptron is a classifier. Strictly speaking, each output node is a classifier (output = 1 or 0)

If the classes are *linearly separable*, then the Perceptron Convergence Procedure will reach a solution that correctly classifies all items in the training set.

If the classes are not linearly separable, it won't.

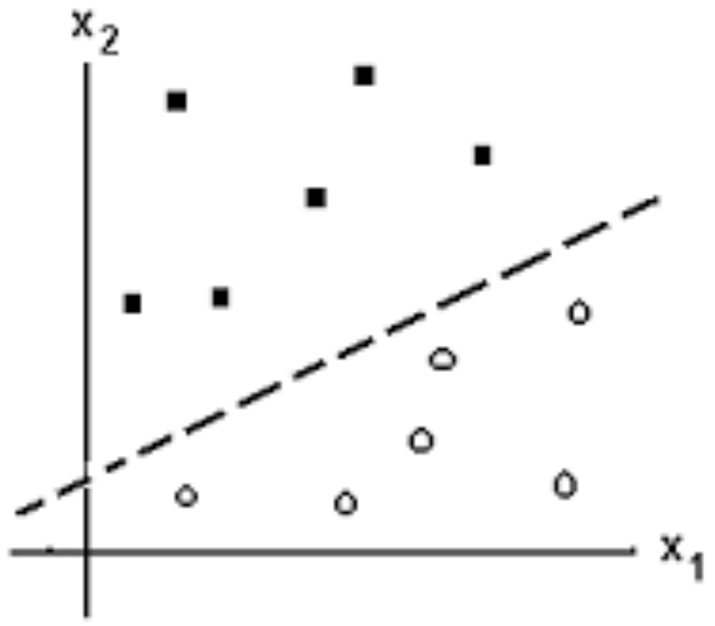
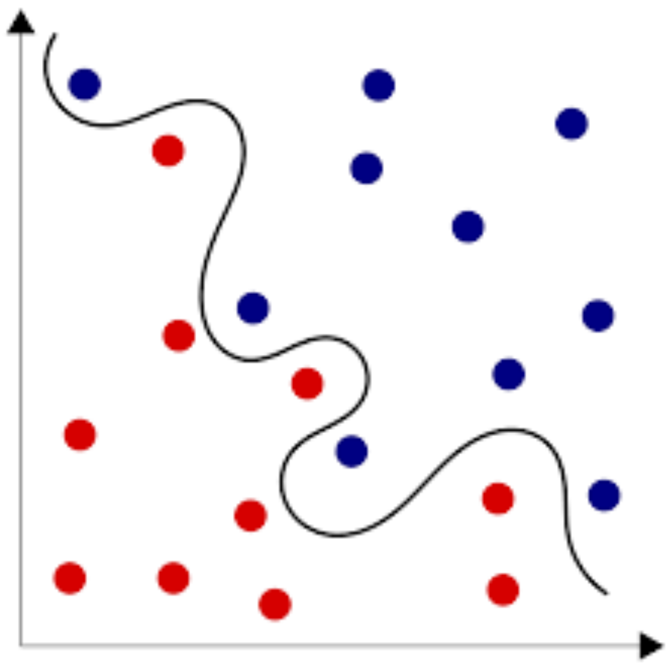
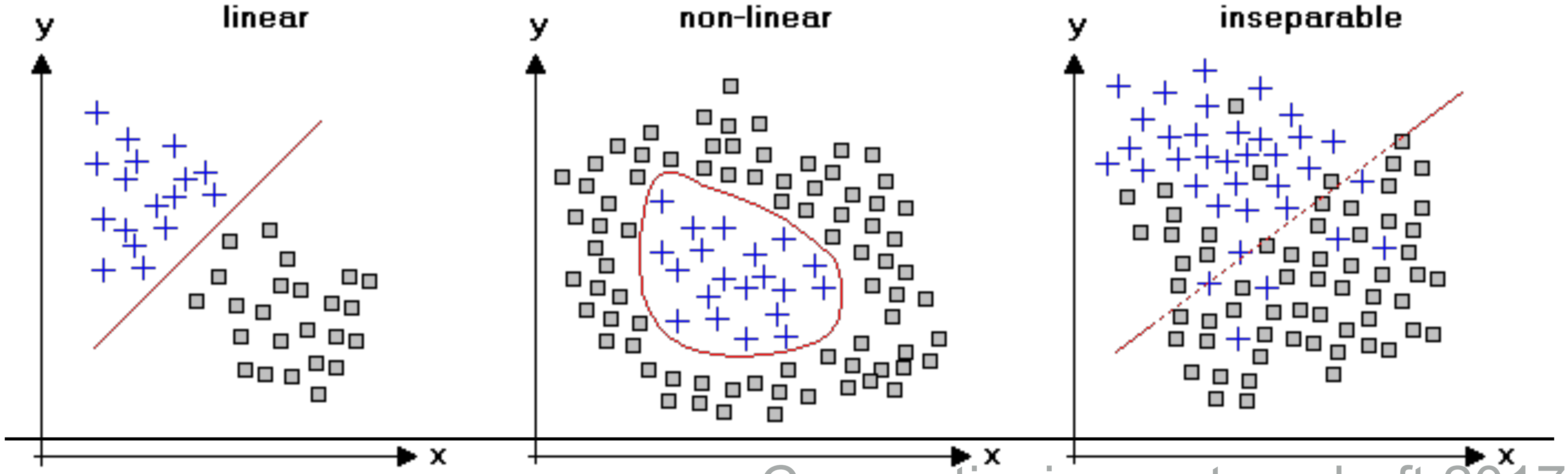


Fig 7. Linearly separated sets of points





The Minsky and Papert Challenge

- A straightforward training procedure for 2-layer linear networks was long known
- It was also known that multi-layered networks with non-linear hidden units could solve much tougher problems
- Minsky and Papert (*Perceptrons,* 1969) famously claimed that such complex networks could not be readily trained
- Backpropagation (back prop) famously solved this problem (for many cases)