CN510: Principles and Methods of Cognitive and Neural Modeling

Models of Serial Order

Lecture 16

Instructor: Anatoli Gorchetchnikov <anatoli@bu.edu>

Chaining Outstars

Using multiple outstars allows to create associative avalanche networks to learn serial order in a sequence

Examples of sequence learning:

- Temporal sequence of distributed sensory inputs, or a memory for an episode
- Temporal sequence of multi-muscle activations, or a complex motor action like playing the musical instrument (although note the durations)

What is a minimal number of cells to encode an arbitrarily complex sequence?

Simplest Way to Represent Time

Define spatio-temporal vector

$$I(t) = \left(I_1(t), I_2(t), ..., I_n(t) \right)$$

where time is continuous and space is discrete

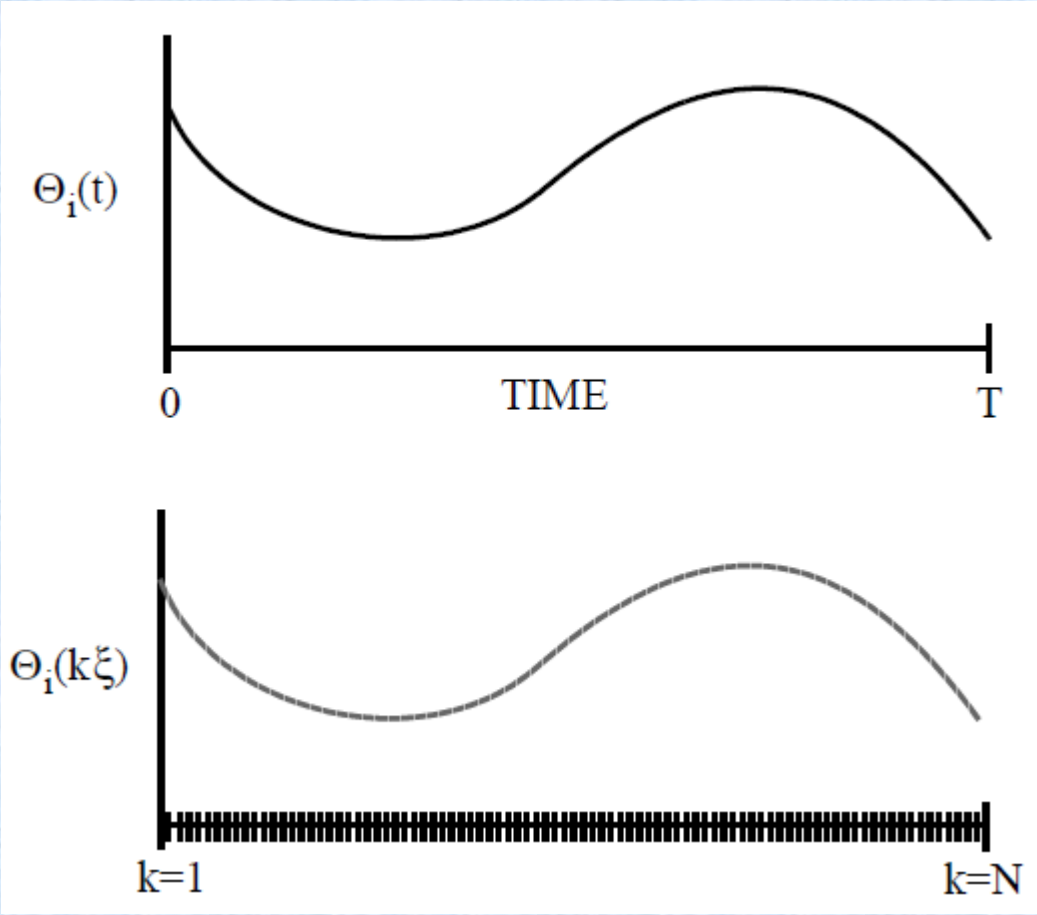
Define pattern functions for all *n* sites

$$\Theta_i(t) = \frac{I_i(t)}{\sum_{k=1}^{n} I_k(t)}$$

Each of these functions can be arbitrarily well approximated by a sequence of sampled values

$$\Theta_i(\tau), \Theta_i(2\tau), ..., \Theta_i(N\tau),$$

Simplest Way to Represent Time



Interval of discretization shall be short enough so that function does not change much between samples

Instead of learning a continuous function, learn a sequence of stills:

$$\Theta(k) = \left(\Theta_1(k\tau), \Theta_2(k\tau), ..., \Theta_n(k\tau)\right)$$

In case of motor sequence the inertia in the joints and muscles will smooth out the transitions

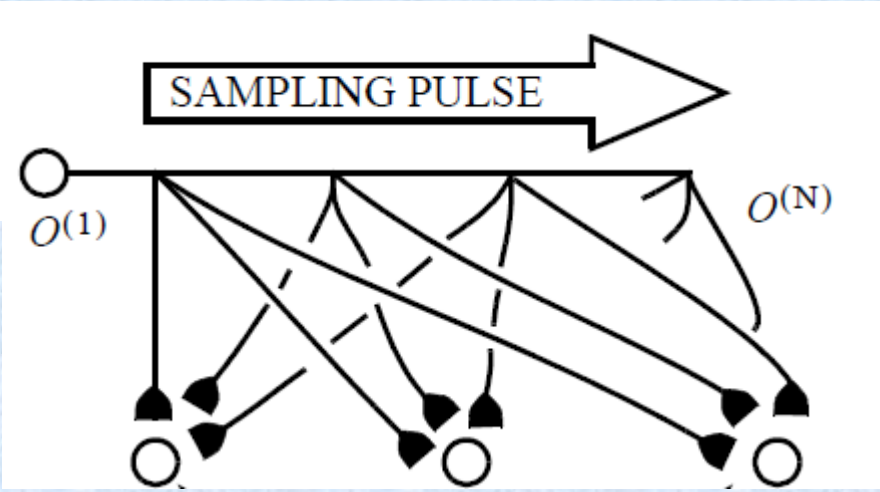
To maintain a note for longer time, just repeat the input over several time steps

In the case of sensory inputs there is also temporal smoothing in the system (that's how we see movies rather than sequences of frames)

Each still can be learned by an outstar

Minimal Realization

One cell with branching axon:



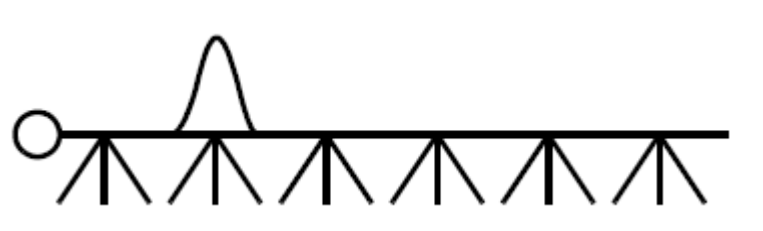
There is a time delay τ for a signal to travel between branching points

This network is called simple avalanche

Constraints

Short sampling pulse and intersample interval comparing to the rate of change of the input

For a rapidly changing input denser packing and shorter pulse



Unbiased outstars, all with the same parameters

Sampling pulse is released in synchrony with the sequence start, so that on every trial same synapses sample same frames

Equations

$$\dot{x}_0 = -a_0 x_0 + I_0(t)$$

$$\dot{x}_i = -a(t) x_i + \sum_{k=1}^{N} b_{ki}(t) w_{ki} + I_i(t)$$

$$\dot{w}_{ki} = -c(t) w_{ki} + d_{ki}(t) x_i$$

where

$$b_{ki}(t) = b\left[x_0(t - k\tau) - \Gamma\right]_+$$

$$d_{ki}(t) = d\left[x_0(t - k\tau) - \Gamma\right]_+$$

note the $k\tau$ term comparing to last lecture

Avalanche Properties

Learns spatio-temporal pattern perfectly up to a desired resolution

The number of input (border) cells controls spatial resolution

The number of branching points on the axon of the source cell controls the temporal resolution

After learning it can perform a spatio-temporal pattern following the activation of the source cell

- Without interruptions
- With a constant rate

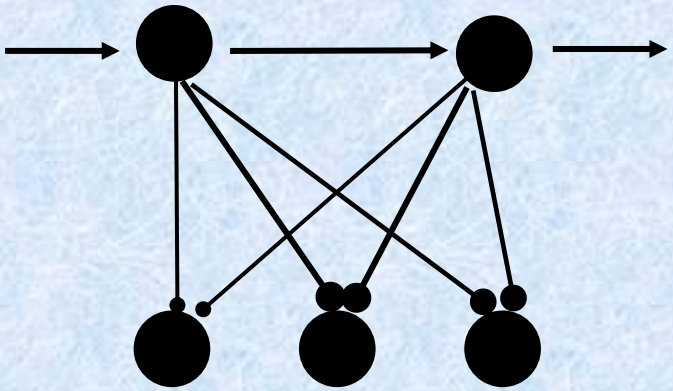
In addition to these limitations, the speed of action potential propagation is too fast for all but the fastest sequences

We need to

- decrease a pulse speed
- make it interruptible, and
- allow variable speed performance

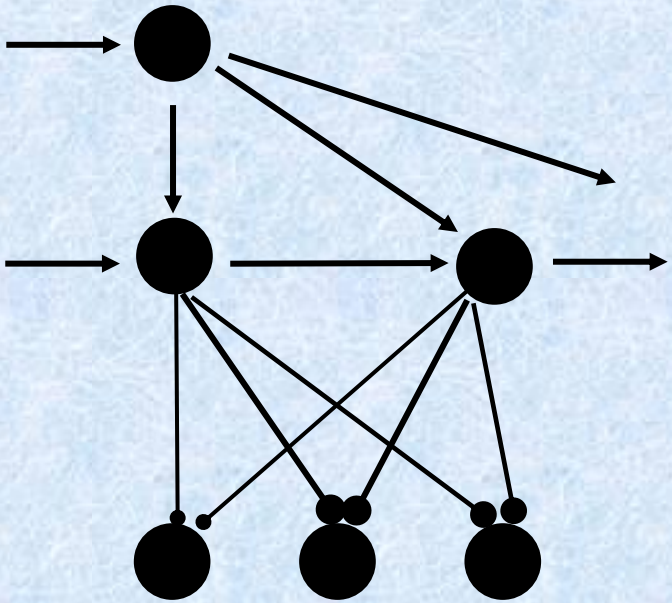
Possible solution:

Step 1: Introduce one cell per still



Modified Avalanche

Step 2: Introduce non-specific signal (arousal, GO-signal, performance control…)



Step 3: Ensure that each successive source cell activates only if it gets both the input from the GO cell and the input from the previous source cell

Equations

$$\dot{x}_{0k} = -a_0 x_{0k} + b_0 [x_{0k-1} + G(t) - \Gamma_0]_+$$

$$\dot{x}_i = -a(t) x_i + \sum_{k=1}^{N} b_{ki}(t) w_{ki} + I_i(t)$$

$$\dot{w}_{ki} = -c(t) w_{ki} + d_{ki}(t) x_i$$

where $G(t)$ is the GO signal and

$$b_{ki}(t) = b\left[x_{0k}(t - k\tau) - \Gamma\right]_+$$

$$d_{ki}(t) = d\left[x_{0k}(t - k\tau) - \Gamma\right]_+$$

Note that the first source cell shall have external input instead of x_{0k-1}

Factorization of Order and Velocity

If higher amplitude of the GO signal can entail faster cell activations, then it can be used to regulate the performance speed

Beyond Pre-programmed Chains

Usually want to be able to learn the "stills" (spatial patterns) on a piecewise basis, then learn to string them together in arbitrary orders (e.g., *Lashley, 1951*)

- Learn various chords (spatial patterns), then learn a sonata as a series of chords

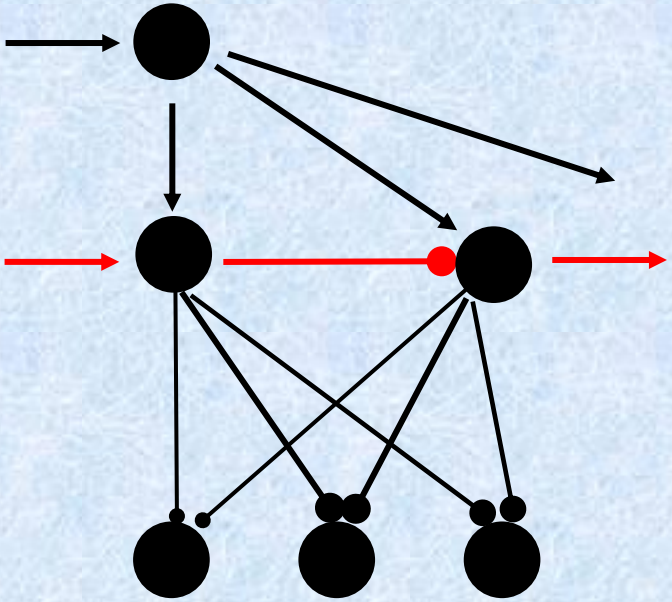
The basic avalanche learns everything at once; the spatial pattern "units" cannot be used in other sequences

A natural extension is an associative avalanche, or associative chain

- Serial order emerges from subsequent activation of sites, but this order is learned

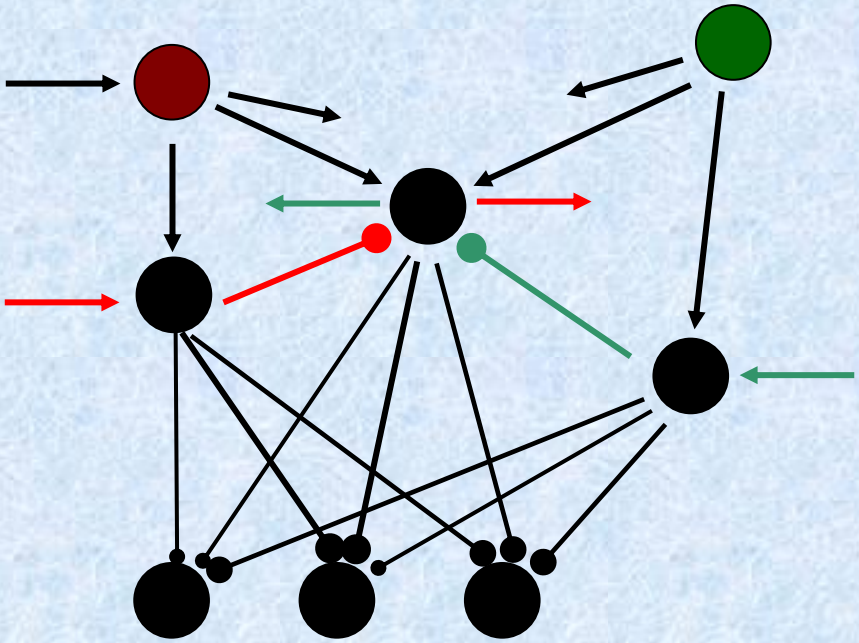
Associative Chain

Similar architecture as before



but the links between source cells are learned for each individual sequence

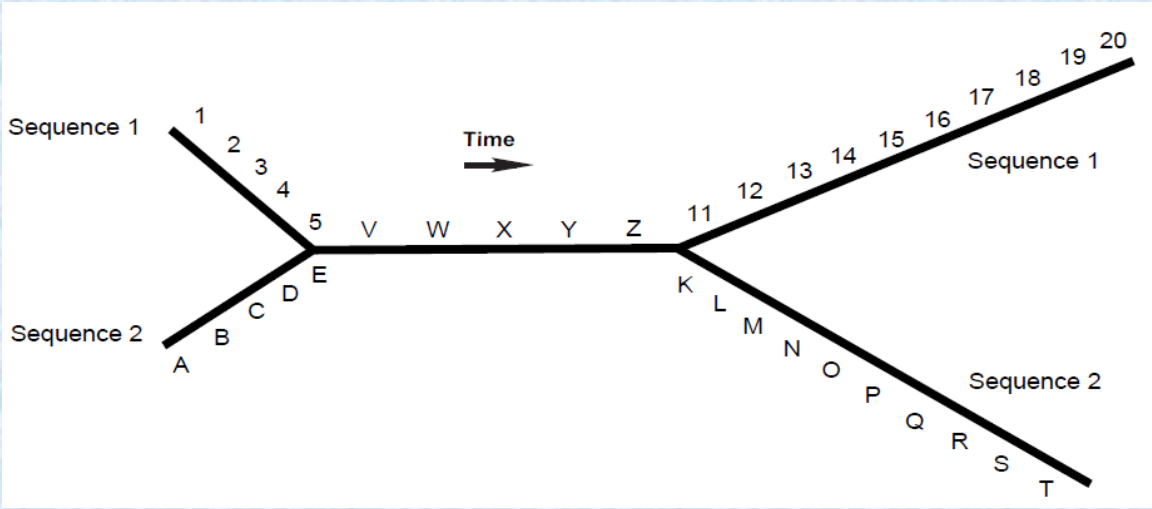
Sequence selection can be done through multiple GO cells

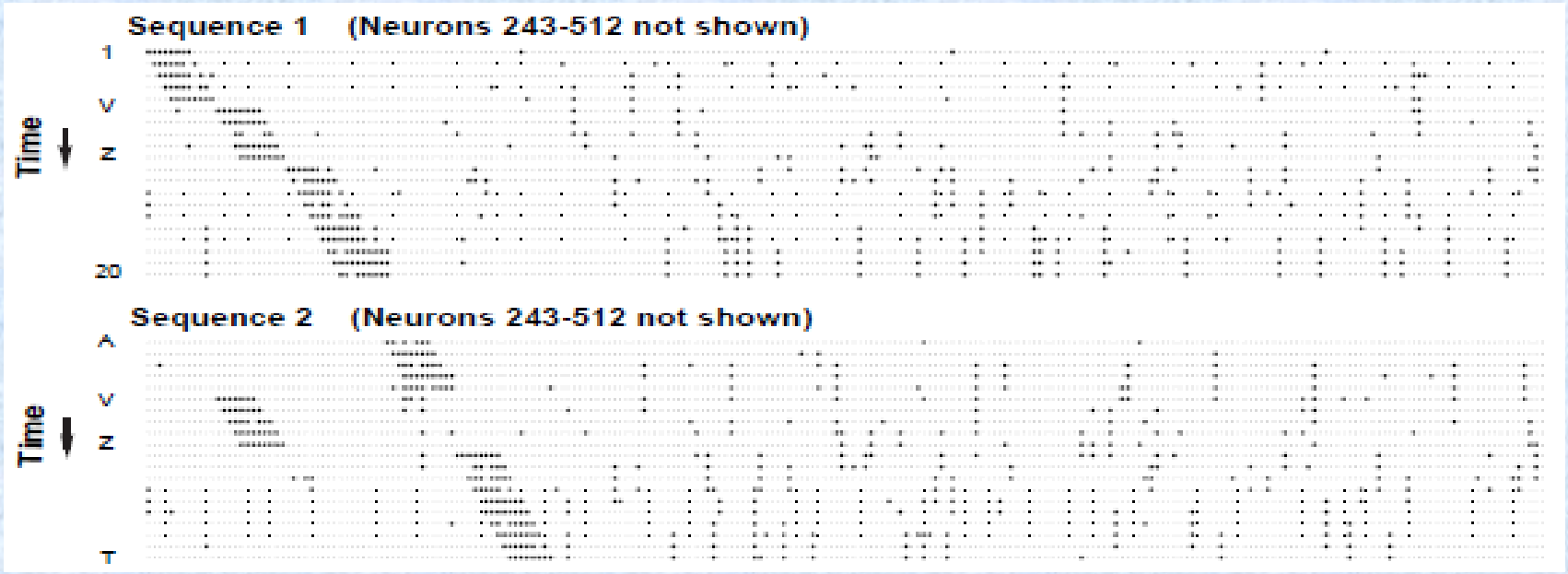


The GO cells can be implemented using similar mechanism to local context neurons of *Levy et al (1995)*

This will allow to learn chunks of these sequences

Levy et al, (1995)



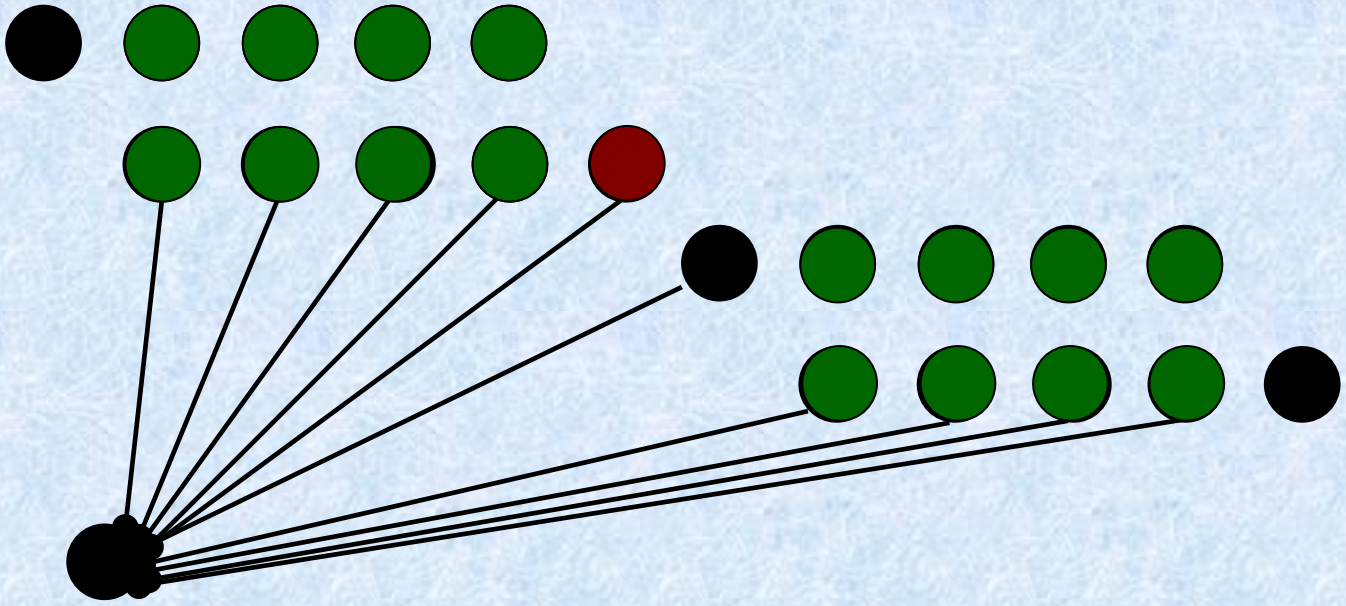


Levy et al, (1995)

Each sequence element is represented as a set of active input neurons

Subsequent elements of a subsequence have a significant overlap of active neurons

Local context neurons have bidirectional adaptive projections



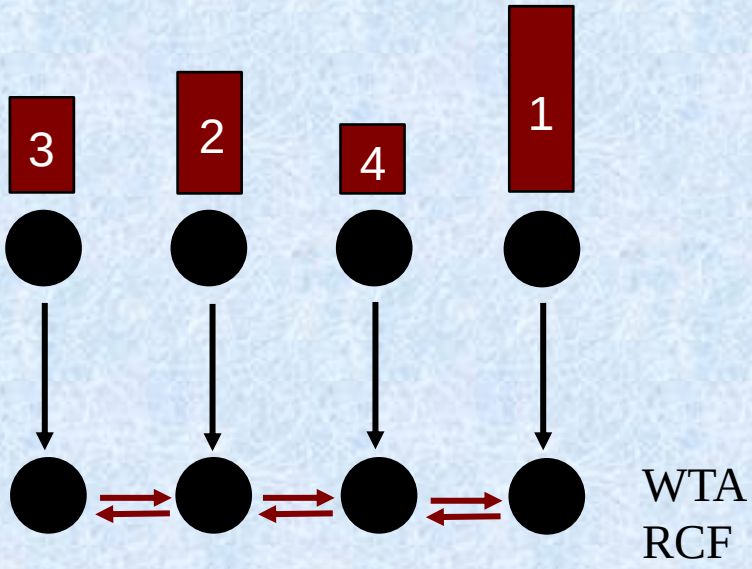
Competitive Queuing Approach

The main problem with associative avalanches: they do not match the data on serial order tasks

Specifically, they do not explain high frequency of errors like typing "taht" instead of "that"

Need a model that is built on a mechanism that would lead to such errors when fails

Competitive Queuing does that



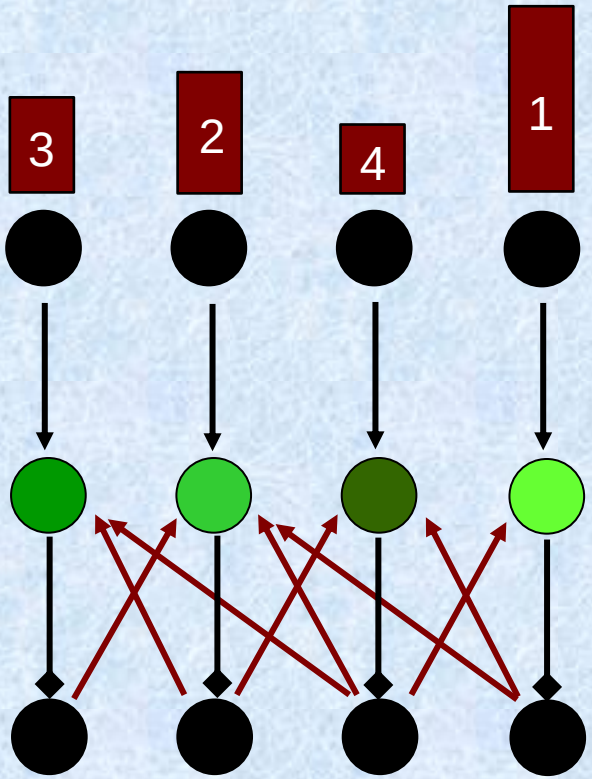
Serial order emerges from simultaneously activated sites, the winner performs the action

Competitive Queuing Approach

If CQ fails, then there is a high chance that instead of the *n-th* element we will perform the one after it, and then recover the *n-th* one – matches the data

How to remove the executed elements from further winning?

Use gated multipole: RCF split in two populations, signal goes through depletable synapses

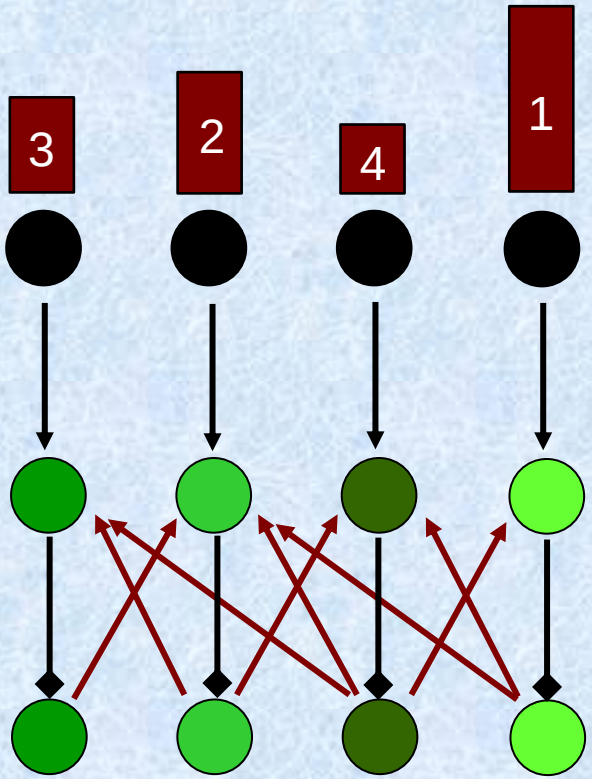


Competitive Queuing Approach

If CQ fails, then there is a high chance that instead of the *n-th* element we will perform the one after it, and then recover the *n-th* one – matches the data

How to remove the executed elements from further winning?

Use gated multipole: RCF split in two populations, signal goes through depletable synapses

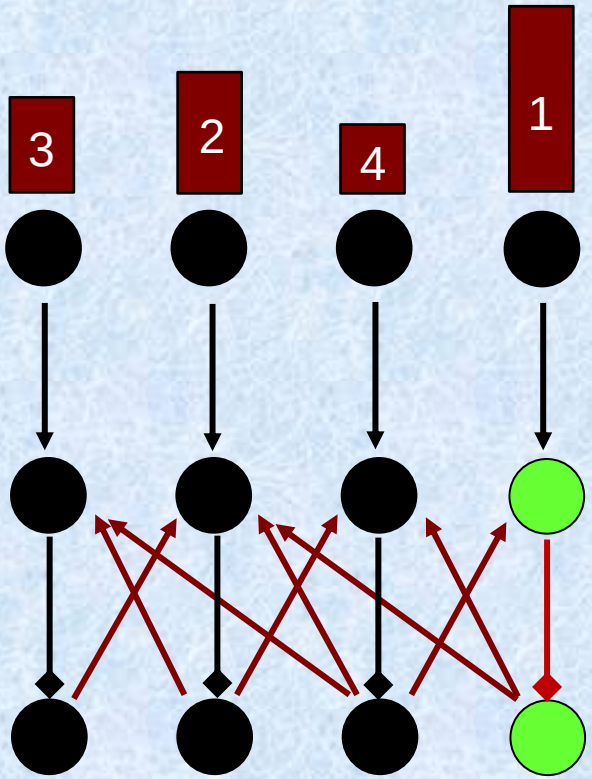


Competitive Queuing Approach

If CQ fails, then there is a high chance that instead of the *n-th* element we will perform the one after it, and then recover the *n-th* one – matches the data

How to remove the executed elements from further winning?

Use gated multipole: RCF split in two populations, signal goes through depletable synapses

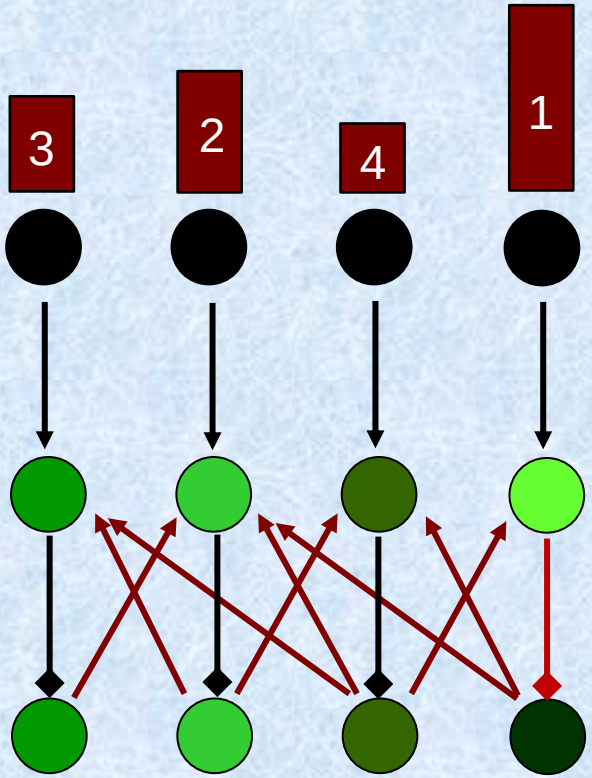


Competitive Queuing Approach

If CQ fails, then there is a high chance that instead of the *n-th* element we will perform the one after it, and then recover the *n-th* one – matches the data

How to remove the executed elements from further winning?

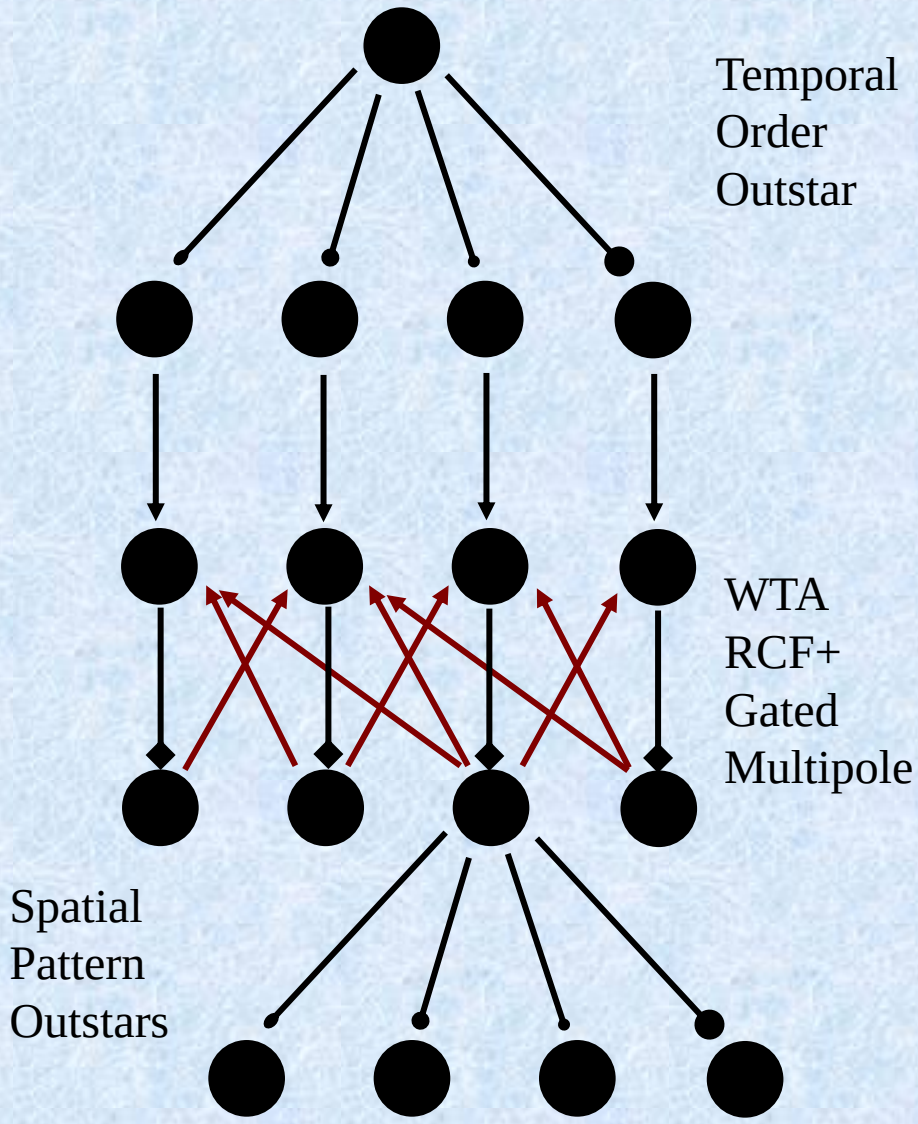
Use gated multipole: RCF split in two populations, signal goes through depletable synapses



Winner gets depleted, so next time it has lower chance of winning

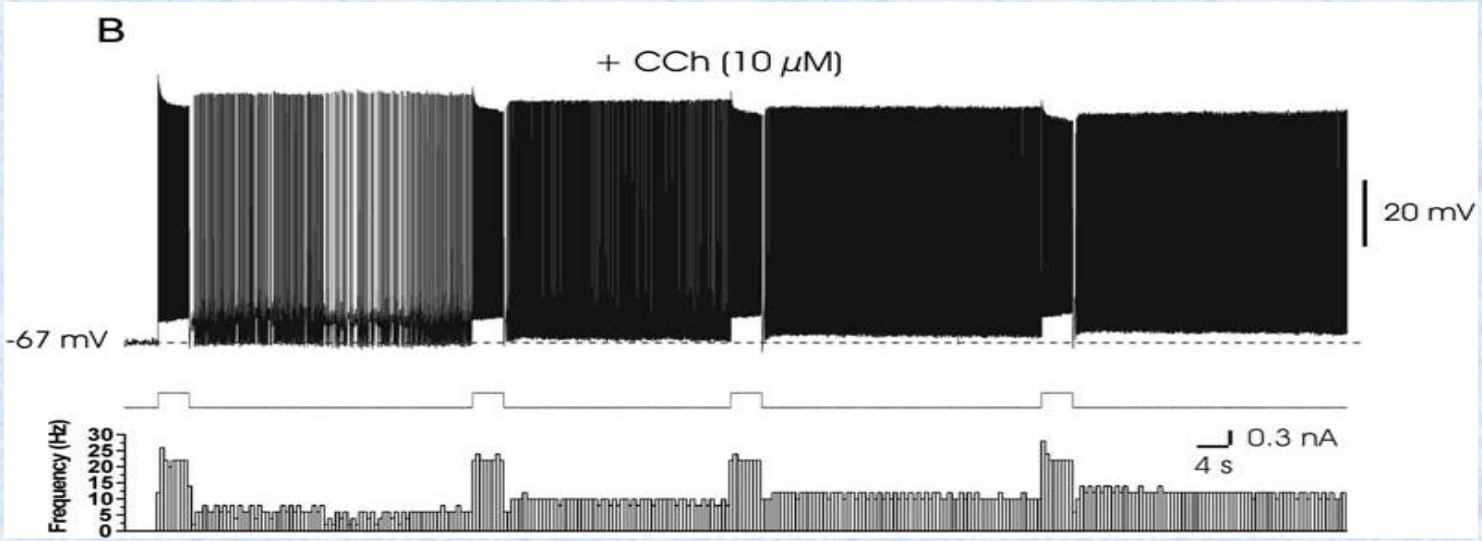
Competitive Queuing Approach

- Temporal pattern of events is encoded as a spatial pattern of synaptic weights
- If weights' sizes correspond to practice order, and if readout is from highest activity site to lowest activity site, then an outstar can encode serial order information
- Note that RCF normalizes the output, so all items will be equally strong



Koene et al (2003) Spiking Serial Buffer

Based on *Alonso et al* studies of persistent spiking neurons in entorhinal cortex



These neurons have non-specific cation current that provides afterdepolarization

They also receive rhythmic inhibition from interneurons driven by theta rhythm

Koene et al (2003) Spiking Serial Buffer

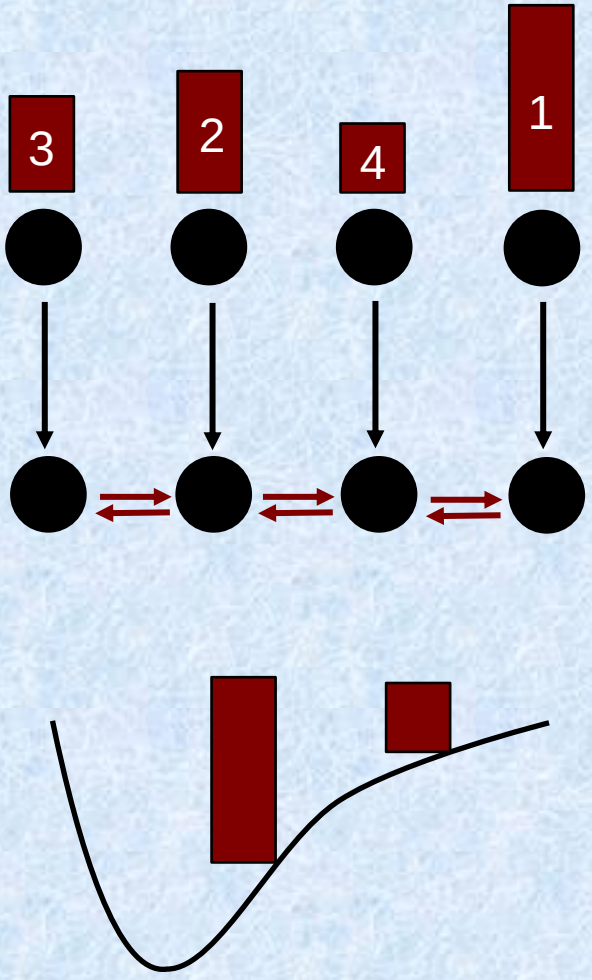
Input strength and (or) timing determines the spike timing of input neurons\

Assume that no two inputs spike on the same theta cycle

Strong input produces a spike earlier in the recovery from inhibition

Lets say it pushes the others to the next cycle

We can remove this winner using same depletable synapses



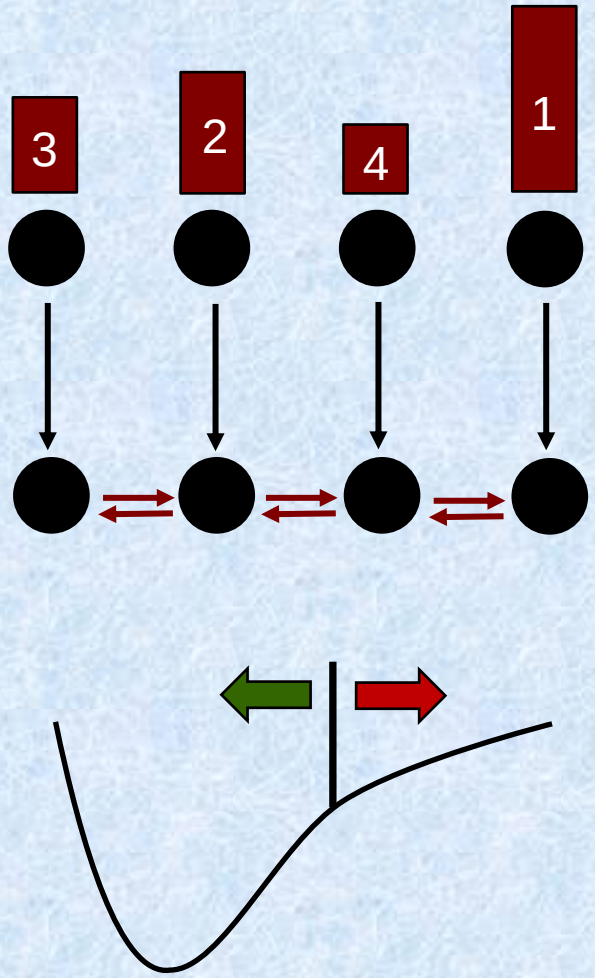
Koene et al (2003) Spiking Serial Buffer

On the second stage spike initially caused by input, so timing is driven from outside

On the next cycle the timing changes:

- If ADP is faster than theta spike will shift earlier
- If ADP is slower, it will be delayed

This determines whether the buffer is FIFO or LIFO (preserves the order or reverses it)



Koene et al (2003) Spiking Serial Buffer

When the next input comes in the first spike is shifted

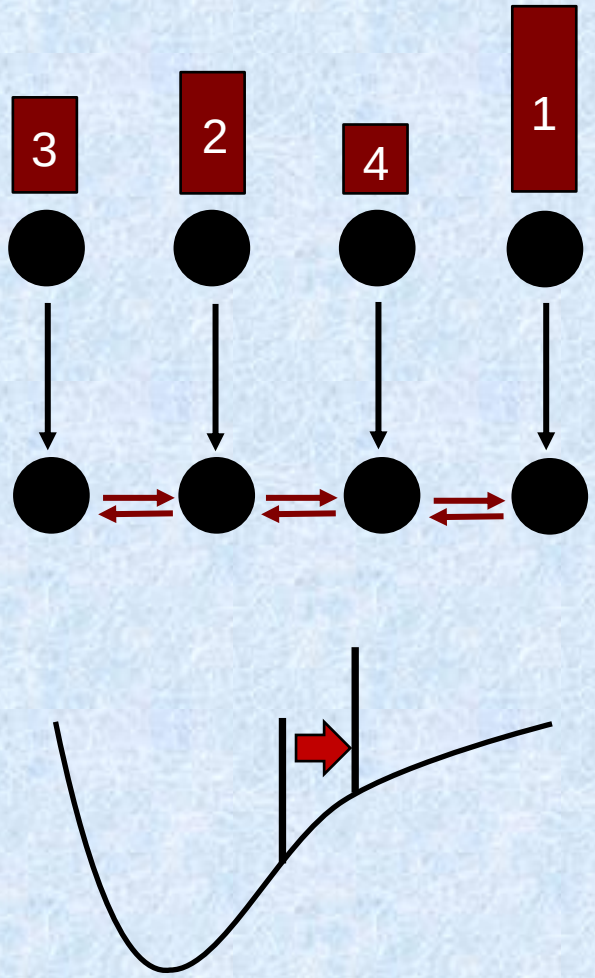
So the next one is inserted either before or after the first one

Recurrent inhibition keeps them separated on subsequent cycles

Can pack about four spikes in a theta cycle

The fifth input:

- Pushes the first one out in LIFO
- Is ignored in FIFO



Next Time

Sensory and motor representations in cortex: input and output pathways, smooth mapping, and cortical magnification

Supplementary Readings:

PNS Chapter 18 [Amaral, D.G. The functional organization of perception and movement.]

PNS Chapter 19 [Saper, C.B., Iversen, S., and Frackowiak, R. Integration of sensory and motor function: The association areas of the cerebral cortex and the cognitive capabilities of the brain.]

Homework Due: Outstar