

# Learning Reward Timing Using Reinforced Expression of Synaptic Plasticity

Harel Shouval<sup>1</sup>, Jeff Gavornik<sup>1</sup>, Marshall Shuler<sup>2</sup>, Mark Bear<sup>2</sup>, Brian Blais<sup>3</sup>

1. University of Texas Medical School at Houston and University of Texas at Austin
2. HHMI and MIT
3. Bryant University

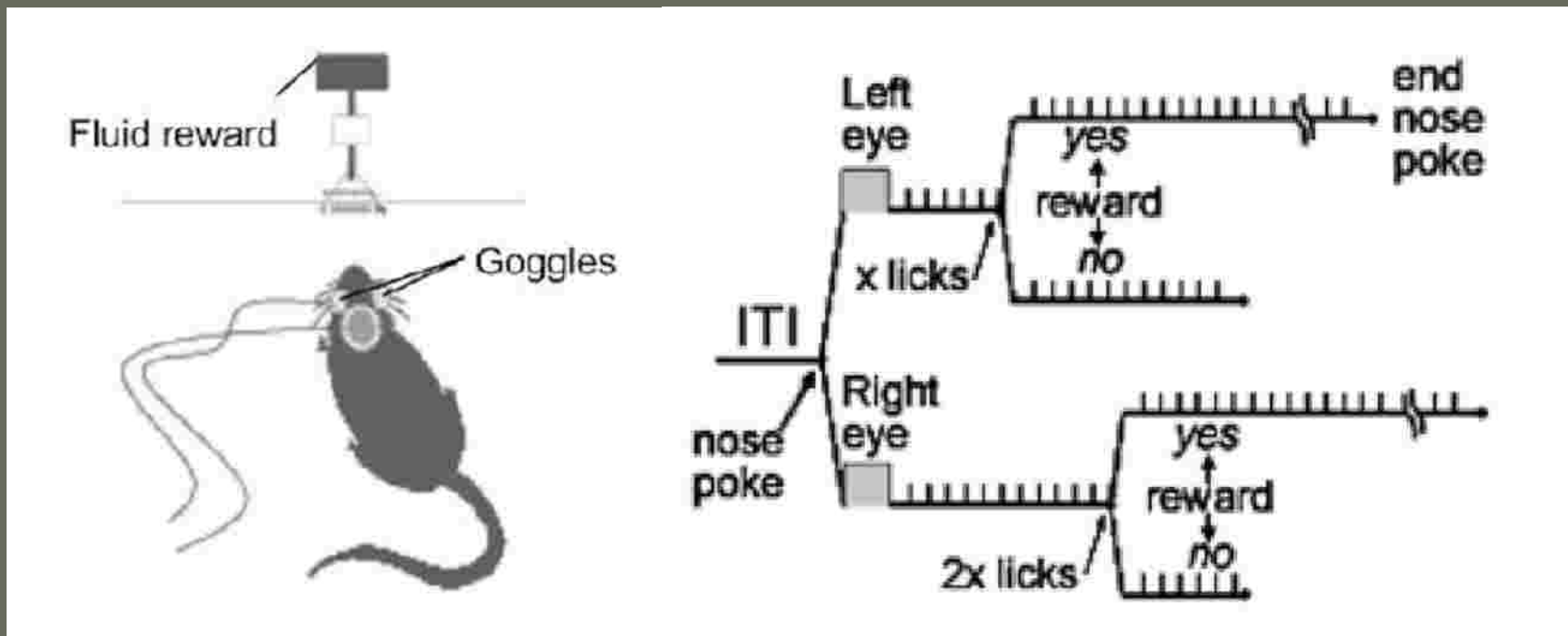
## Different projects:

1. Accounting for deprivation experiments in mouse using BCM and Natural images.
2. Using biophysical models to account for multi-spike plasticity protocols.
3. Using biophysical (spiking) models to account for RF formation with natural images.
4. -

# Reward Timing in the Primary Visual Cortex

Marshall G. Shuler and Mark F. Bear

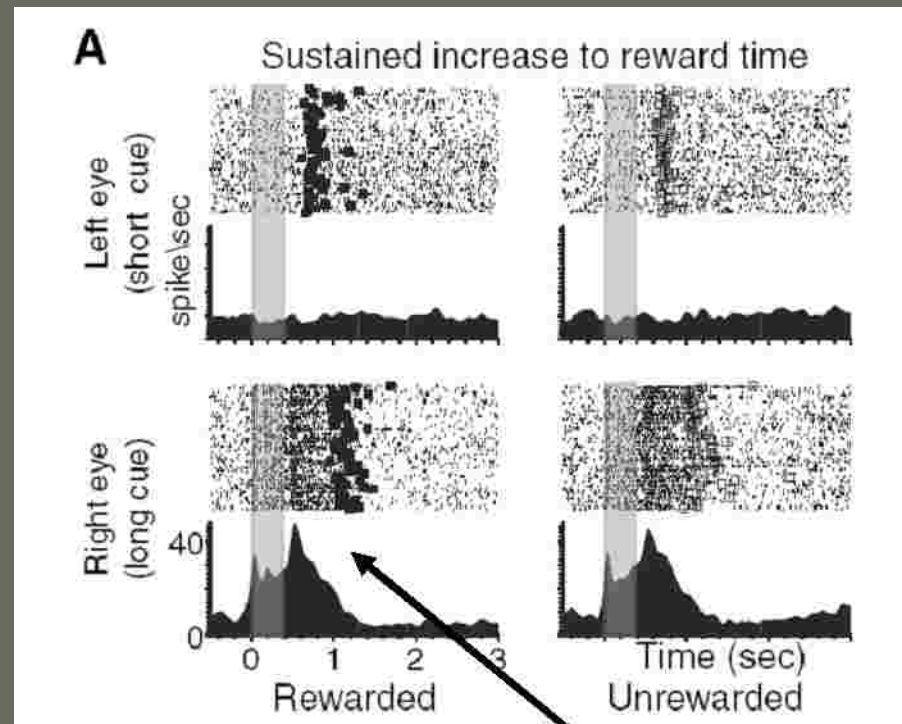
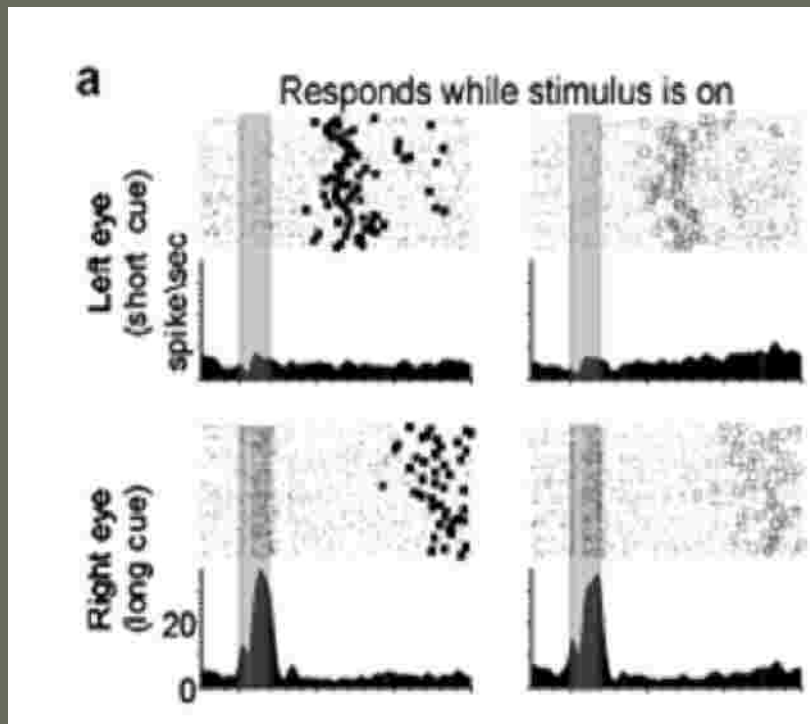
Task: Animal has an LED goggle on each eye.  
If light is flashed to left eye it gets a reward after  $x$  licks.  
if light is flashed to right eye it gets a reward after  $2x$  licks.



Shuler and Bear, 2006

# Learning to predict time: a computational model

Task: Animal has an LED goggle on each eye.  
If light is flashed to left eye it gets a reward after  $x$  licks.  
if light is flashed to right eye it gets a reward after  $2x$  licks.



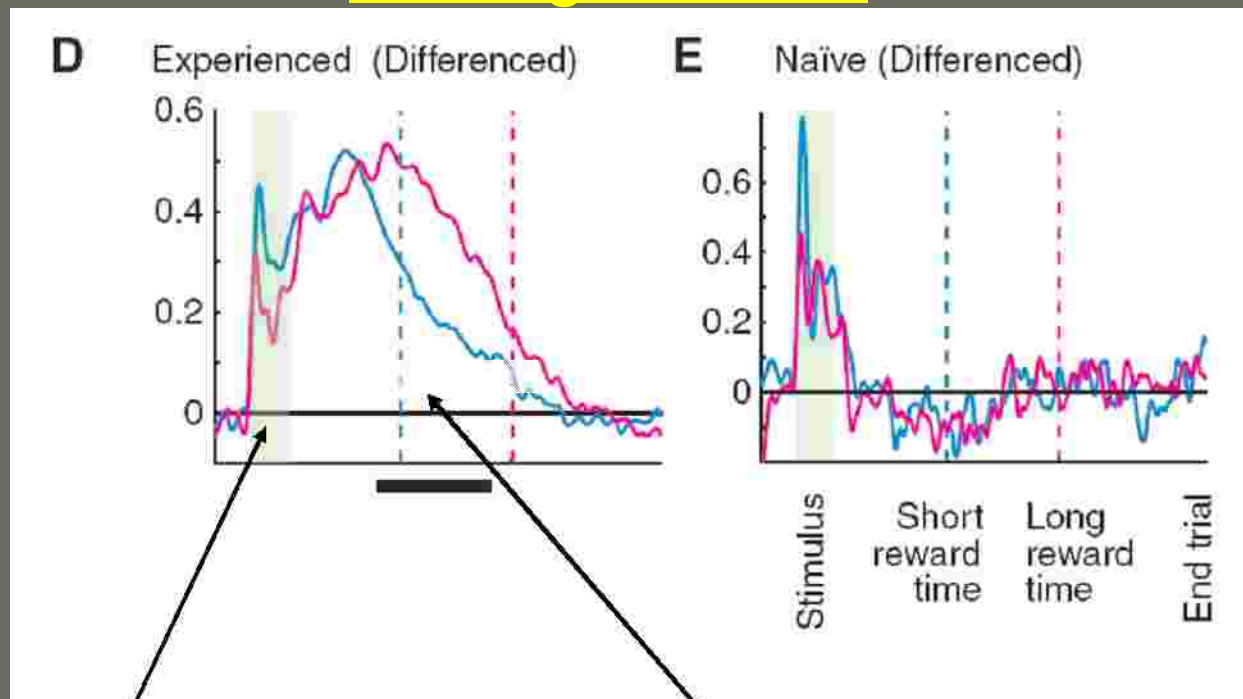
Shuler and Bear, 2006

Sustained response with no visual input,  
reward dependent, interval timing – in  
V1?

## Various response types:

- 43% of neurons showed reward timing after training
- Of those
  - 50% show transient increase
  - 22% transient inhibition
  - 28% peak at reward

## Average results



Stimulus time

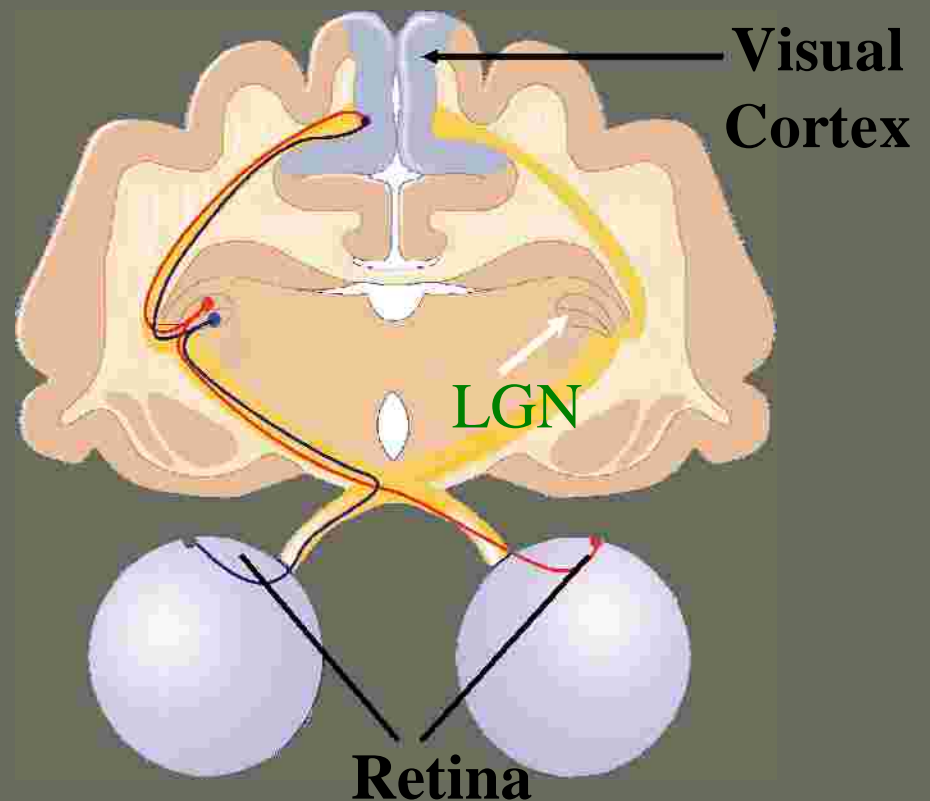
response without stimulus

**Approach: Find a simple model that can qualitatively capture central features of the results**

Questions:

1. How can a neural network represent time?
2. How can a neural network learn this representation?

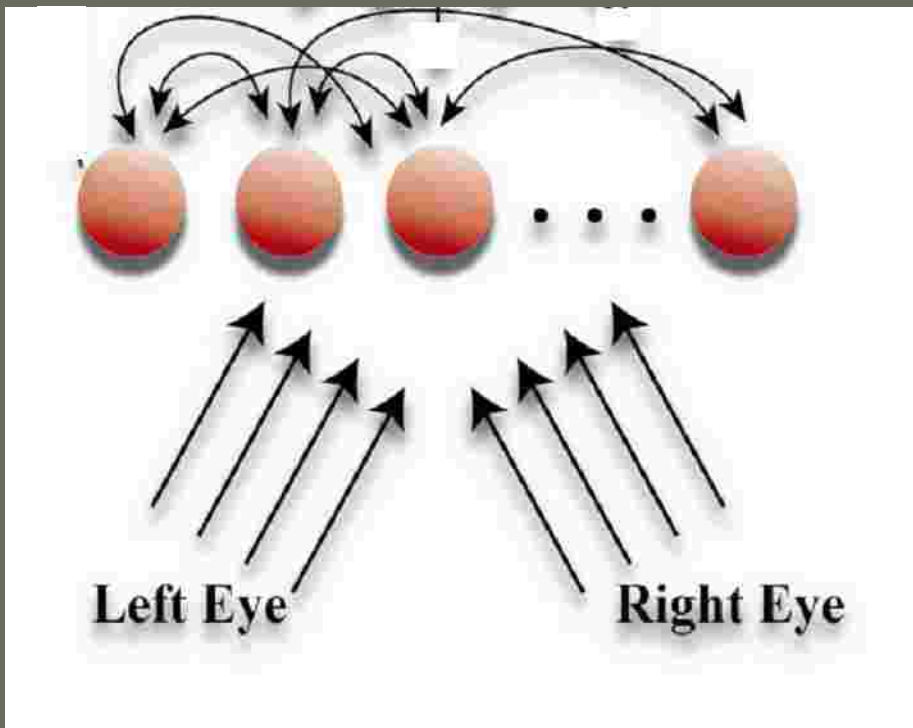
## Visual Pathway



# 1. Representing time

There are many options, one simple option is through a recurrent network architecture.

A simple example of a linear leaky integrator network model

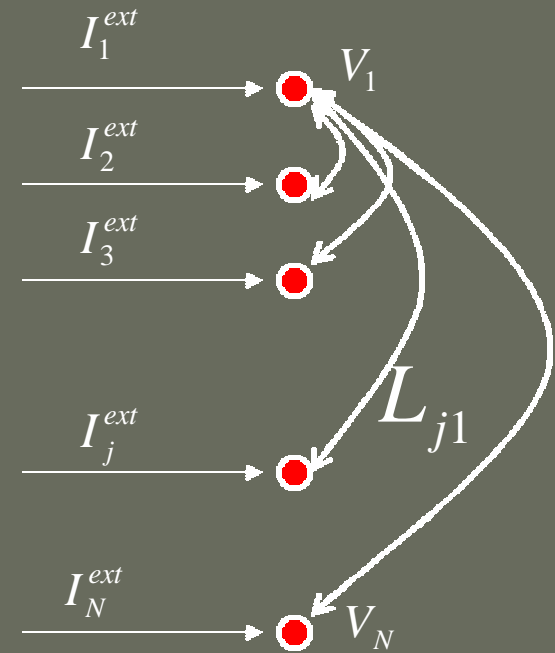


$$\frac{dV_i}{dt} = -\left(\frac{1}{t_m}\right)V_i + I^{ext} + \sum_j L_{ij}V_j$$

$$\frac{dV_i}{dt} = -\left(\frac{1}{t_m}\right)V_i + I_i^{ext} + \sum_j L_{ij}V_j$$

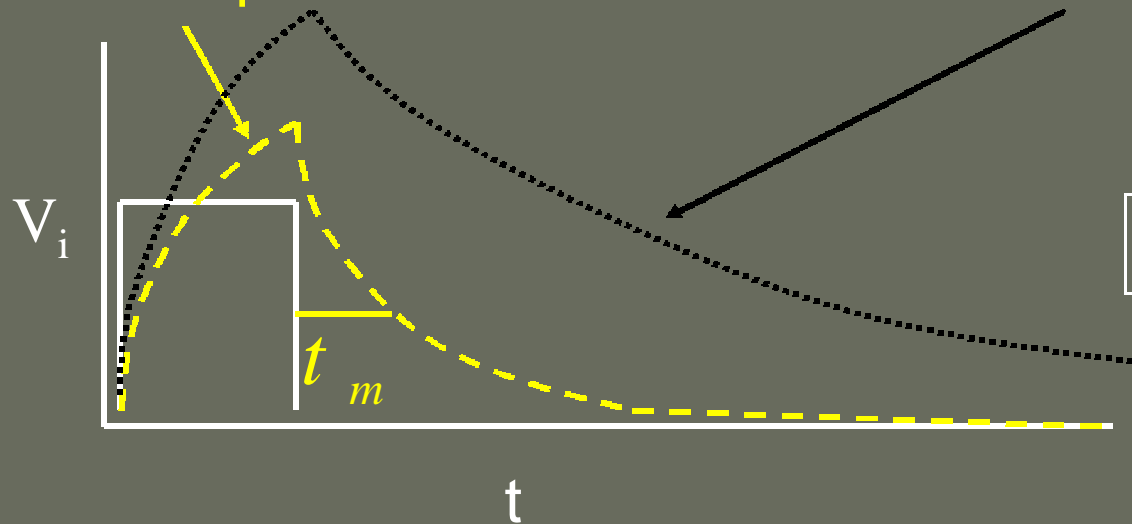
Vector form:

$$\frac{d\mathbf{V}}{dt} = -\left(\frac{1}{t_m}\right)\mathbf{V} + I^{ext} + \mathbf{L} \cdot \mathbf{V}$$



Example:  $L=0$

Example  $L$  is non zero (positive)



$$\mathbf{V}(t) = (V_1(t), \dots, V_N(t))$$

How can we set the values in  $L$  to obtain a desired, reward dependent time constant -  $t_d$  ?

Network dynamics in the falling phase ( $I=0$ ):

$$\frac{d\mathbf{V}}{dt} = -\left(\frac{1}{t_m}\right)\mathbf{V} + \mathbf{L} \cdot \mathbf{V}$$

We can set the weights such that:

$$L_{ij} = \left(\frac{1}{t_m} - \frac{1}{t_d}\right) I_i^{ext} I_j^{ext}$$

In this case the network has exponential dynamics in the falling phase with a decay constant:  $t_d$

However, this method has a significant problem:

$$a = \frac{1}{t_m} - \frac{1}{t_d}$$

For example,  $t_m = 50$  ms:

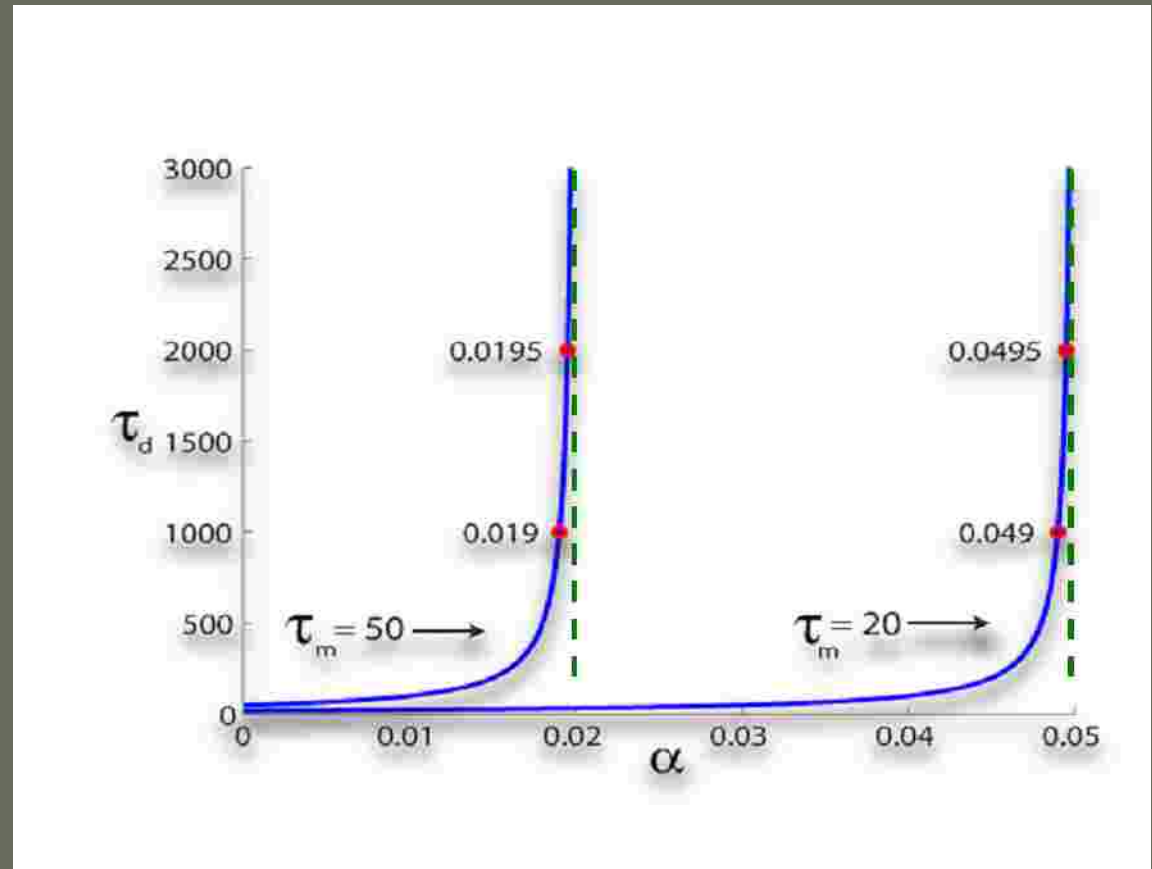
$$a(1000\text{ms}) = 0.0190$$

$$a(2000\text{ms}) = 0.0195$$

$$a(1000)/a(2000) = 0.974$$

(2.5 % difference)

- $t_m = 20$  ms: difference is 1%
- $t_m = 200$ ms: difference is 11%



**What happens for  $a$  to right of the asymptote?**

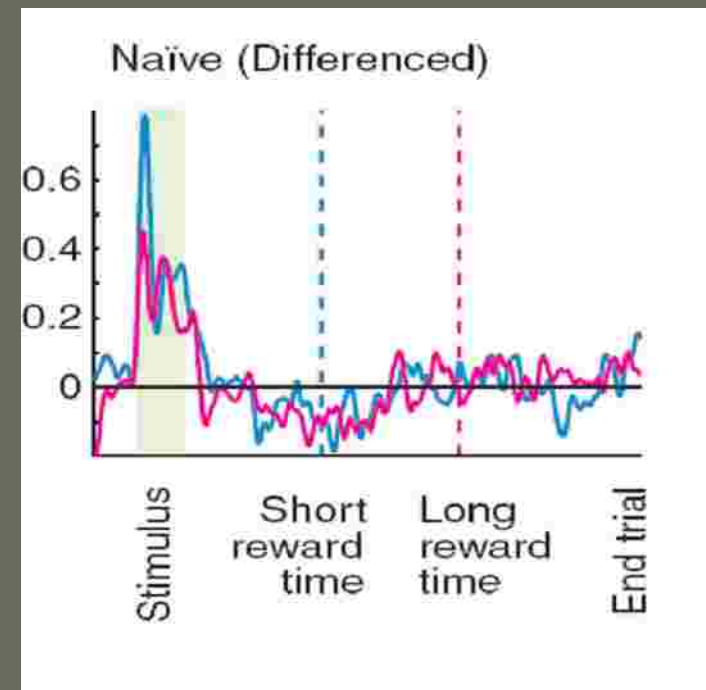
## 2. How can a network *learn* the appropriate weights?

Observations:

1. Learning requires a reward signal
2. The reward comes significantly after the period of neural activity.

3. The term:  $L_{ij} \propto V_i V_j$

Implies roughly that the  
Connections between coactive  
neurons should be enhanced



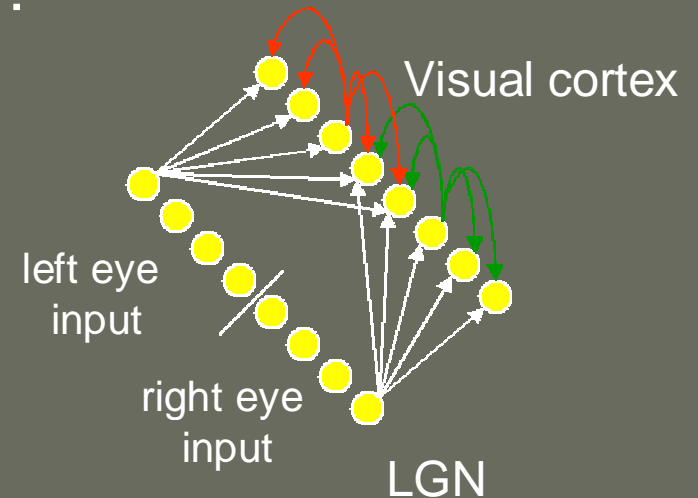
# Learning the “Protoweights” (trace)

## Principles of the learning rule:

1. There are two sets of ‘synaptic weights’:

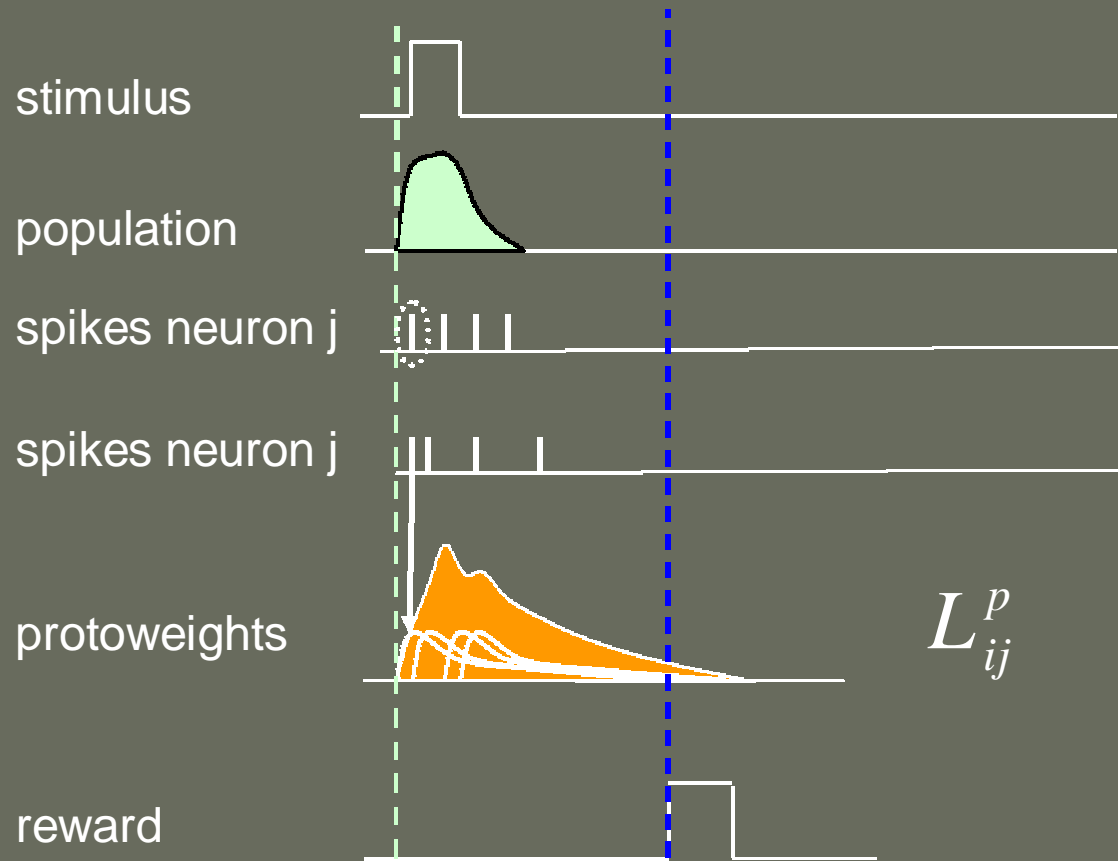
- I. Permanent (expressed) weights  $L$
- II. Proto (unexpressed)  $L^p$

2. At each time step  $L_{ij}^p$  is increased proportionally to the activities in neurons  $i$  and neuron  $j$ , and decays with a time constant  $t_p$ .



$$L_{ij}^p(t + \Delta t) = L_{ij}^p(t) + a V_i(t) V_j(t) - \left( \frac{1}{t_p} \right) L_{ij}^p$$

# Learning the “Protoweights” (trace)

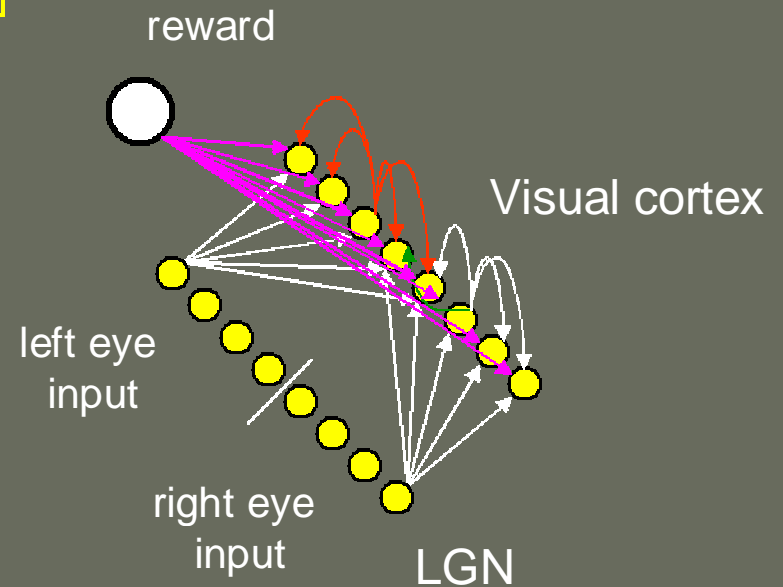
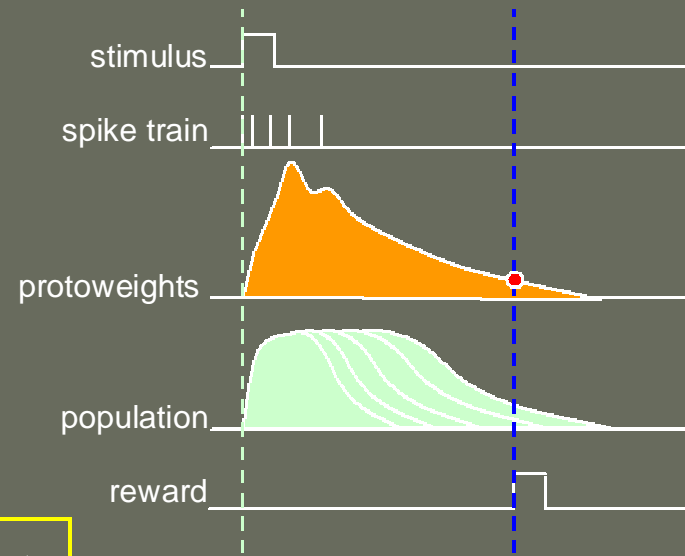


# Global reward signal

Once a reward signal ( $r$ ) is given, the temporary weights are added to the permanent weights:

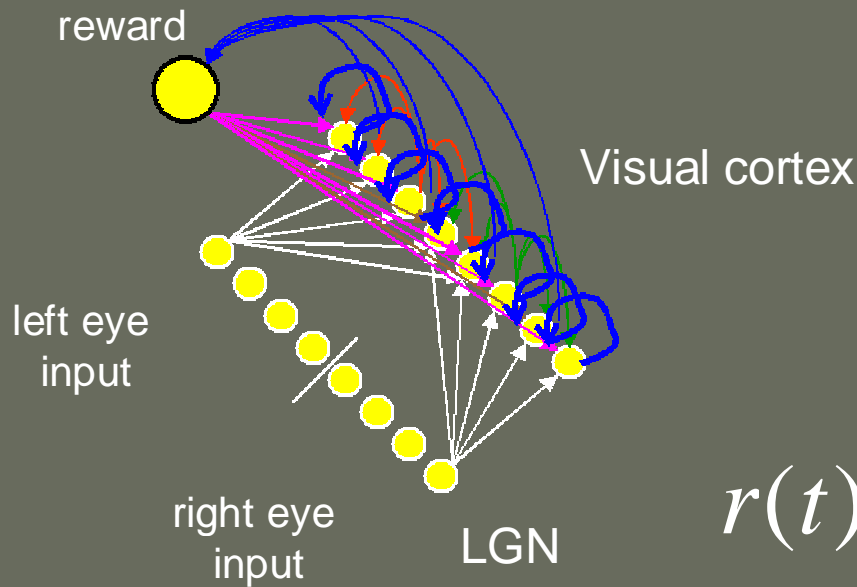
$$L_{ij}(t) = L_{ij}(t - \Delta t) + h L_{ij}^p(t) \cdot r(t)$$

**Problem: weight will keep growing indefinitely.**



# Feedback inhibition of reward

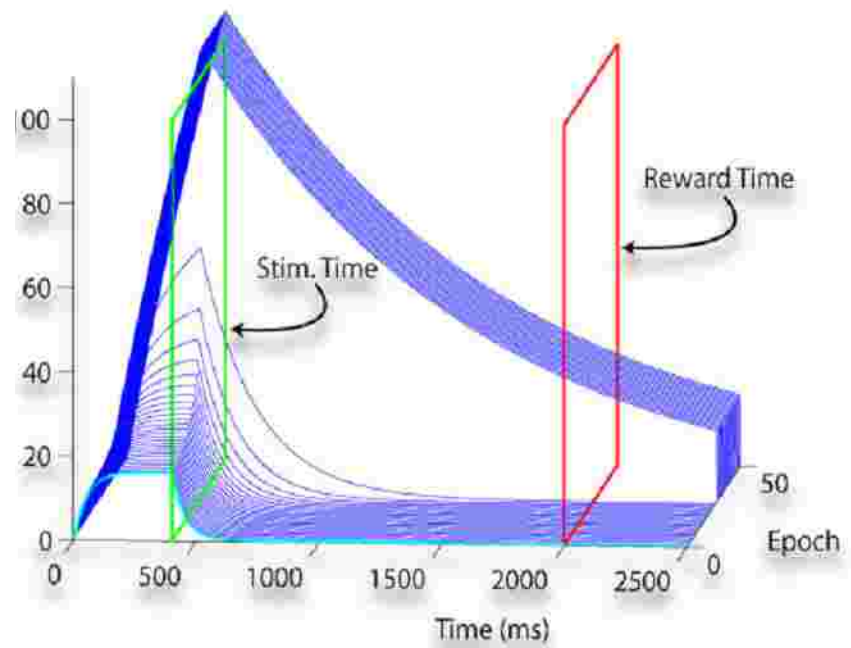
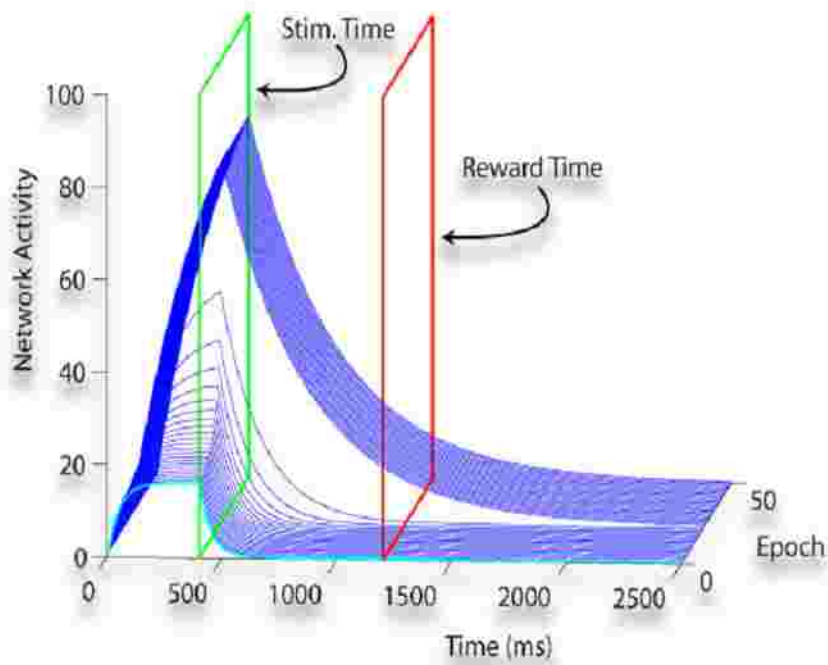
Option II: global inhibition



$$r(t) \rightarrow r(t) - b\bar{V}$$

Rascola Wagner

# Results: with passive integrator



## Results

### What is similar to experiments? (qualitative features)

- Response becomes extended beyond stimulus time.
- The time scale of the response is correlated with the reward time

### What is different? (quantitative features)

- Exponential time course in simulation

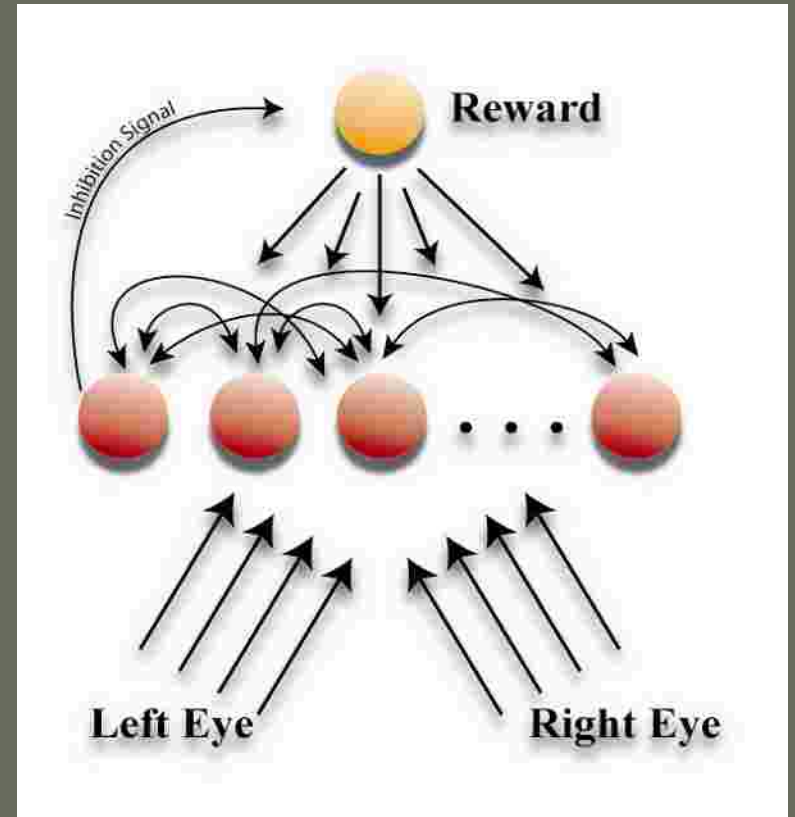
# Replace passive integrator neurons with leaky integrate and fire neurons.

$$C\dot{V} = g_L(E_L - V) + g_E(E_E - V) + g_I(E_I - V)$$

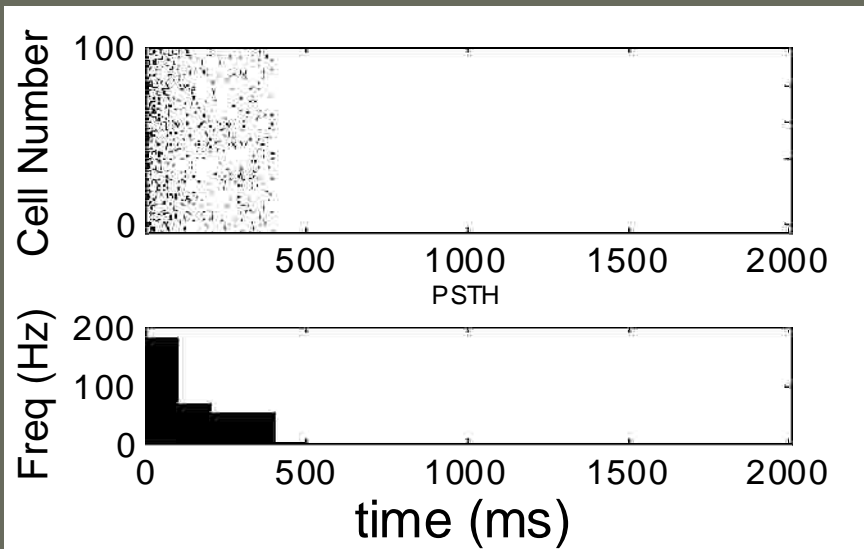
$$g_I = \sum_i w_i s_i \quad t \quad \& = -s$$

$$g_E = \sum_j w_j s_j$$

When the voltage  $V$  reaches a threshold  $V_\gamma$ , a spike is 'generated' and the voltage is set back to a resting value.



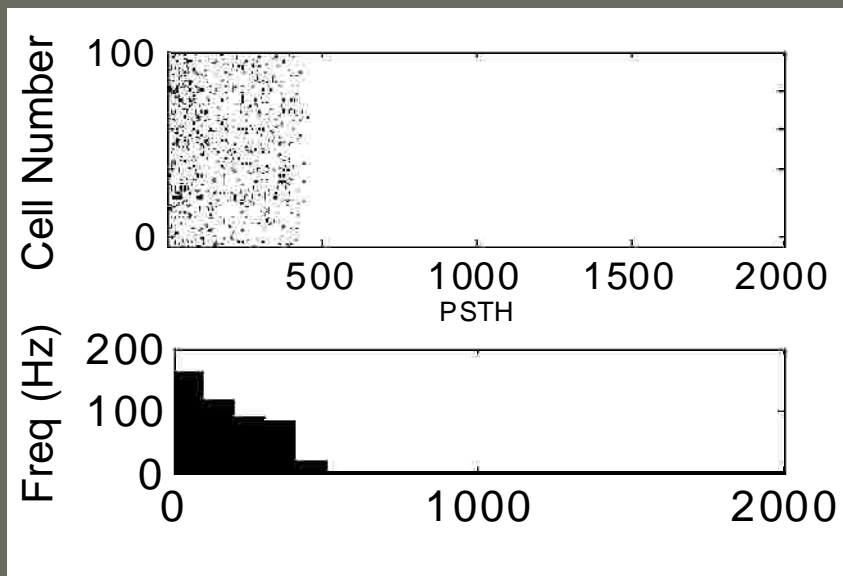
## Input to cortex (Mastronarde 1987)



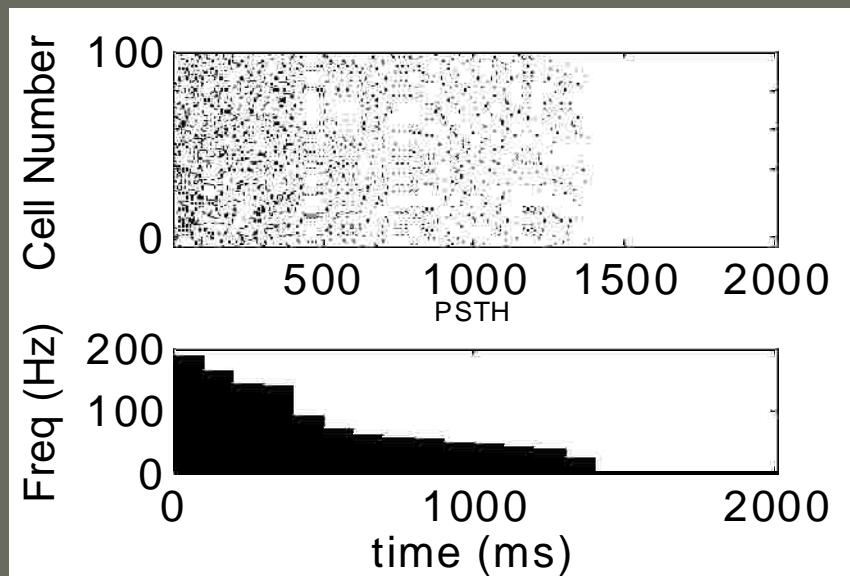
Training a network of spiking neurons with a reward time of **1400 ms**

100 cortical neurons

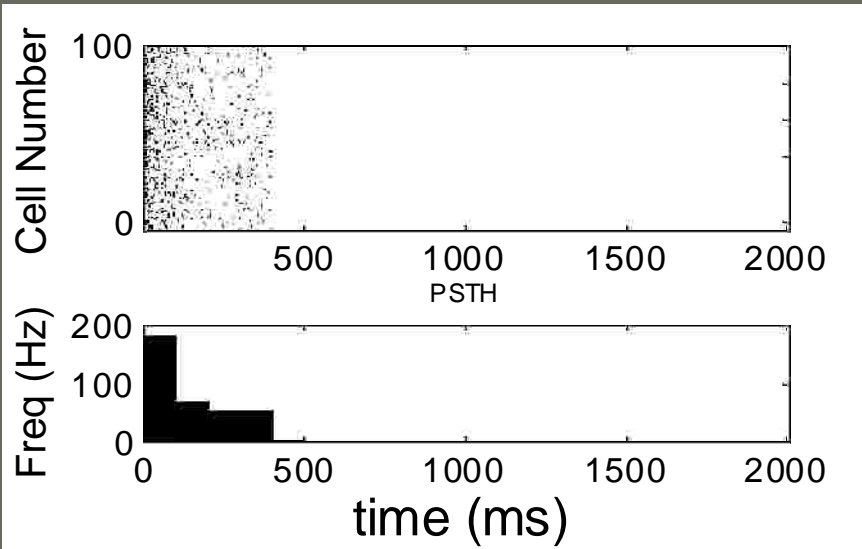
## Naive cortex



## Trained cortex



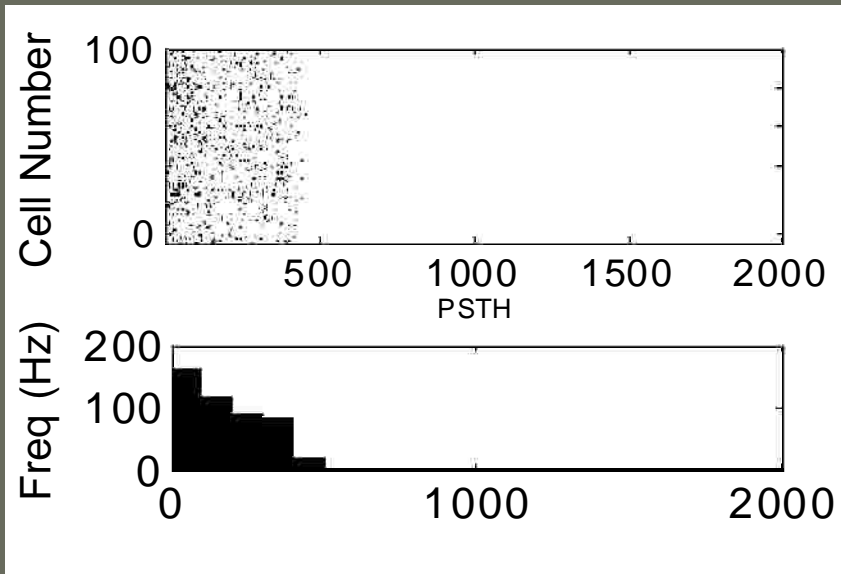
## Input to cortex



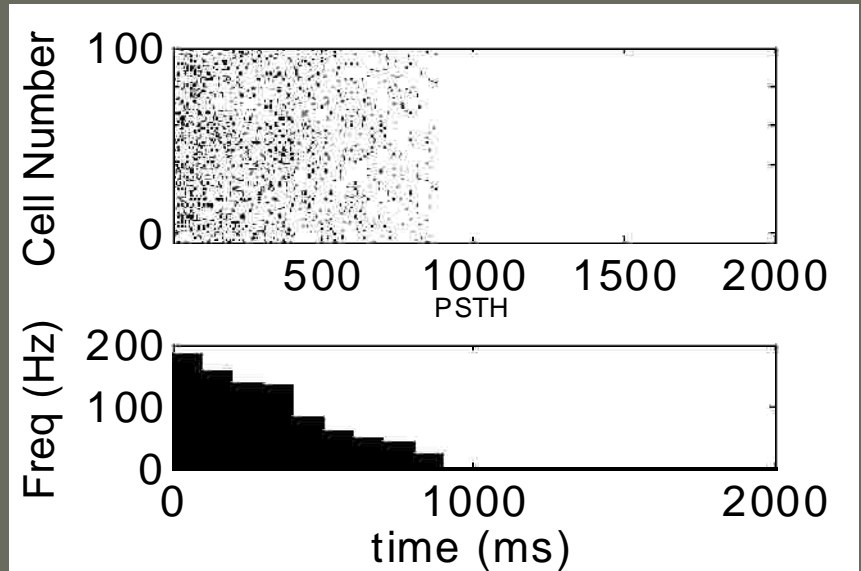
Training a network of spiking neurons with a reward time of 900 ms

100 cortical neurons

## Naive cortex



## Trained cortex



**Aim: A network that learns to represent reward time.**  
**Method – reward dependent expression of synaptic plasticity**

### **Assumptions**

- Time  $\rightarrow$  recurrent network, determined by recurrent weights
- Learning of transient weights is Hebbian
- Expression is reward dependent
- Reward is inhibited by network activity (or locally)
- Both Passive Integrator and Integrate and Fire

### **Comments**

- not robust
- no tapped delay lines assumed

More stuff

Analysis of the learning processes:

$$L_{ij}^p(t) = L_{ij}^p(t - \Delta t) + h L_{ij}^p(t) \cdot (r(t) - b \bar{V}(t))_+$$

stops when  $r(t) \leq b \bar{V}(t)$

If  $r(t) = rd(t - T)$

And since the dynamics require that:  $\bar{V}(t) \propto \exp(-t/t_d)$

We get that at F.P:

$$t_d = T / \log(b / r)$$

If  $\beta/r=e$  then:

$$t_d = T$$

# How many patterns- time pairings can such a network represent?

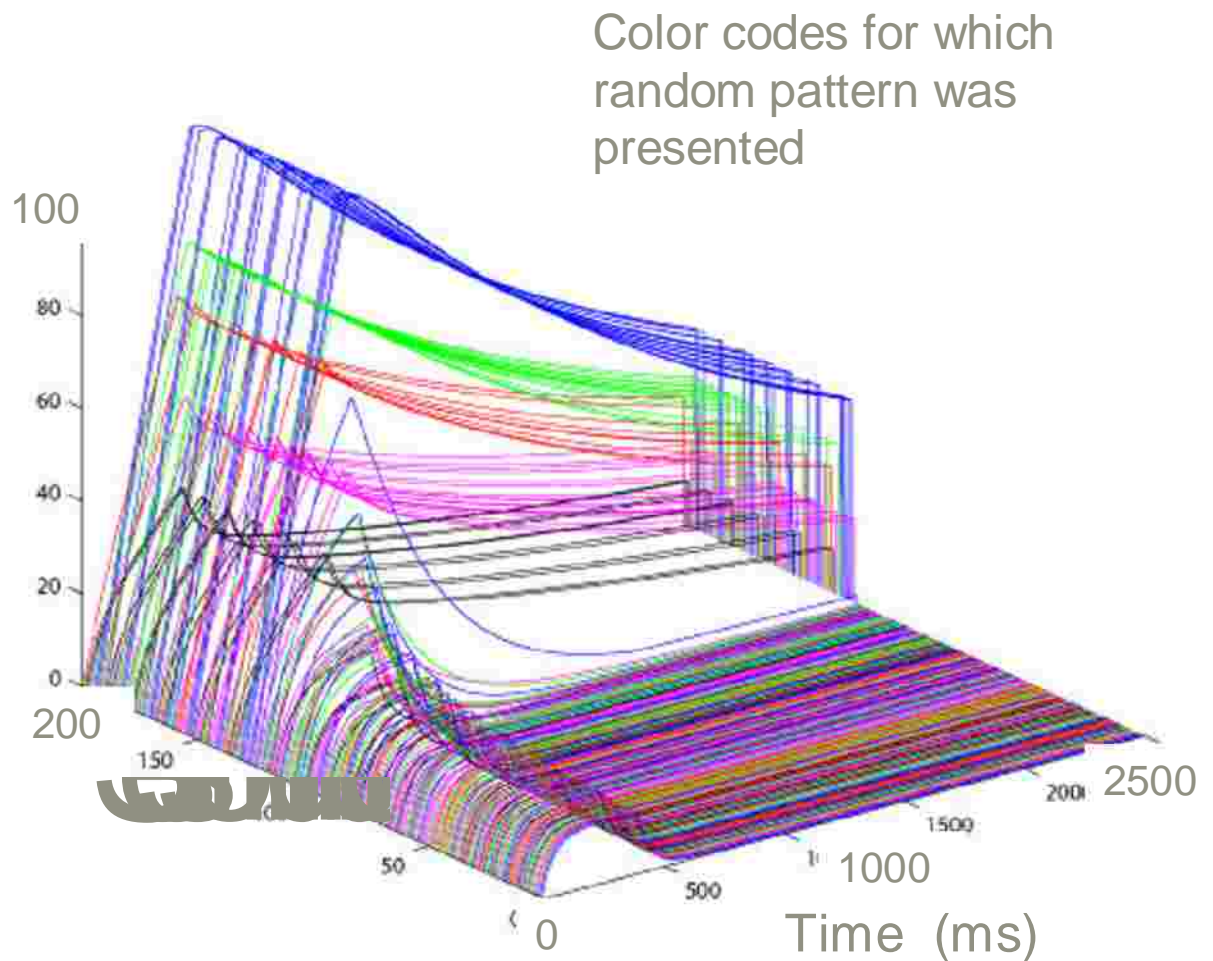
Example:

5 random patterns

100 neurons

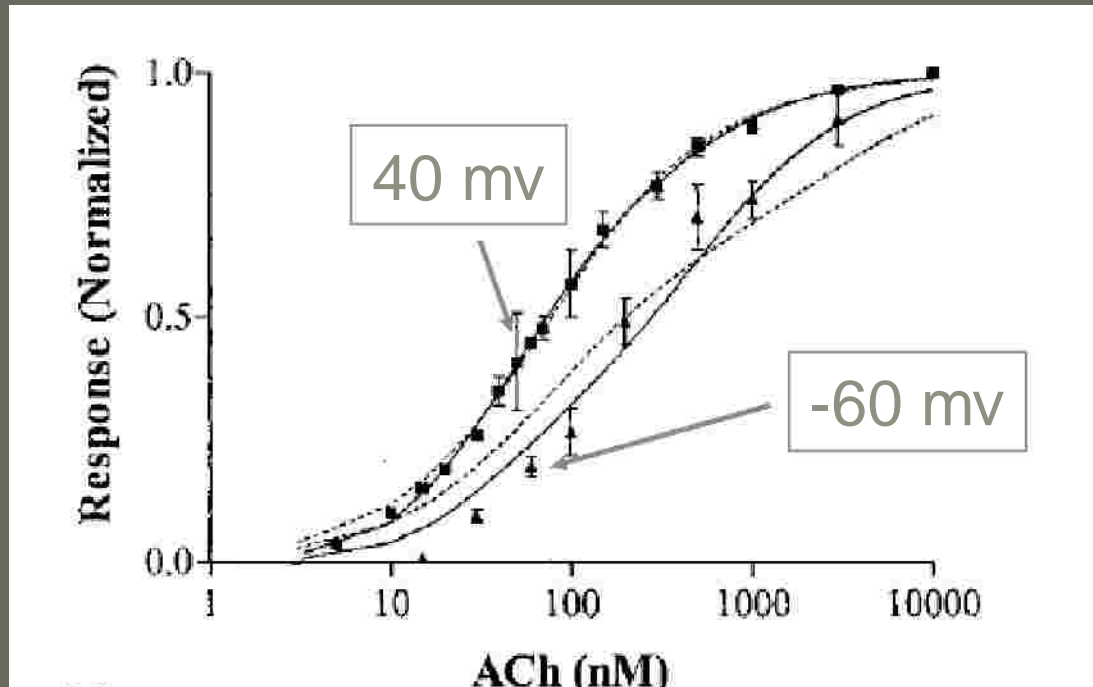
Sparsity 0.25

Paired with random times in the range:  
500-2000 ms



# The M2 Muscarinic G-protein-coupled Receptor Is Voltage-sensitive

Yair Ben-Chaim, Oded Tour, Nathan Dascal, Itzchak Parnas, and Hanna Parnas



The Journal of Biological Chemistry, 2003

# Different predictions from option I and option II

## Thought experiment

One we could replace the cortical reward by an artificial pulse of the reward juice (Ach), option I an option II would have different outcomes.

Option I – learning will not terminate, the network would keep increasing its time constant until non stability is reached.

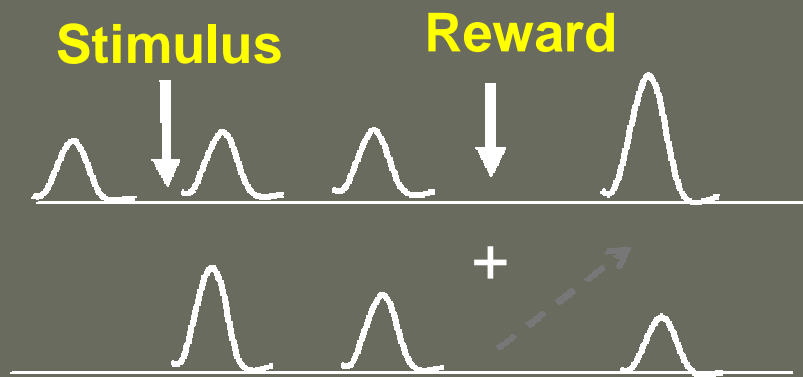
Option II - learning will terminate because the inactivation of response to reward is a single cell phenomena and will occur with the artificial reward as well.

# Speculations regarding mechanism

## 1. What are the proto weights?

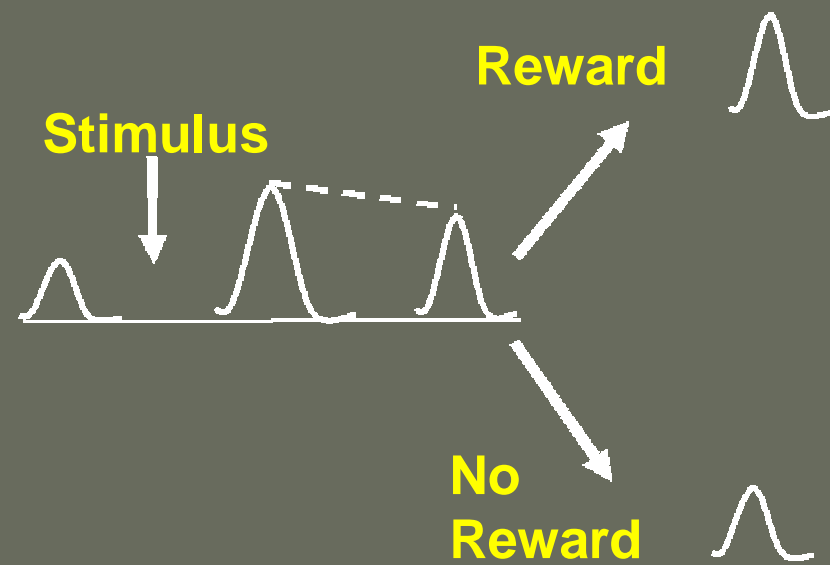
### Option 1

Synaptic weights do increase by Hebbian rule, instead there is a biochemical trace that is converted to weights given reward



### Option 2

Synaptic weights increase by Hebbian rule but decay in the absence of reward



How can we set the values in  $\mathbf{L}$  to obtain a desired time constant ( $t_d$ ) ?

Some math:  $\frac{d\mathbf{V}}{dt} = -\left(\frac{1}{t_m}\right)\mathbf{V} + \mathbf{L} \cdot \mathbf{V}$  in the falling phase when  $I^{\text{ext}}=0$

If  $\mathbf{L}$  is set such that  $\mathbf{L} \cdot \mathbf{V} = a\mathbf{V}$  ( $\mathbf{V}$  is an eigen-vector of  $\mathbf{L}$  with eigen-value  $a$ )

Then  $\frac{d\mathbf{V}}{dt} = -\left(\frac{1}{t_m} - a\right)\mathbf{V} = -\left(\frac{1}{t_d}\right)\mathbf{V}$  so this new neural network has a time constant of decay  $t_d$ .

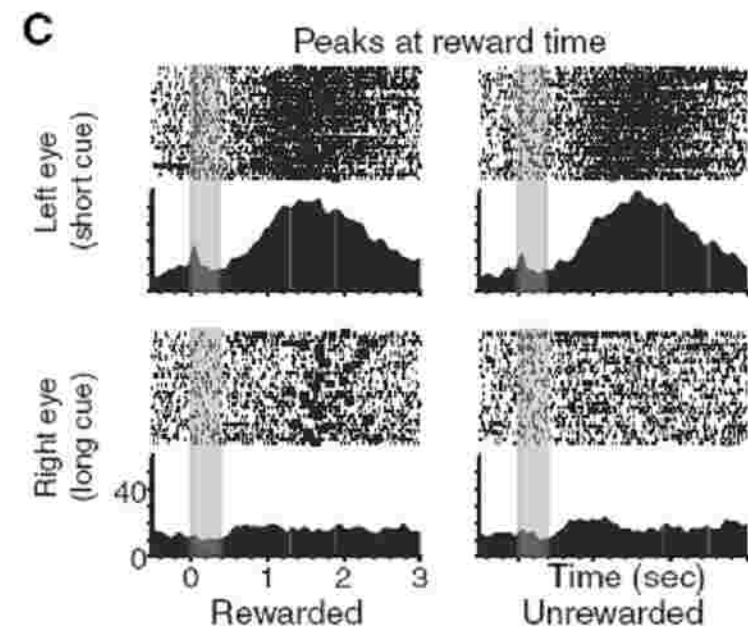
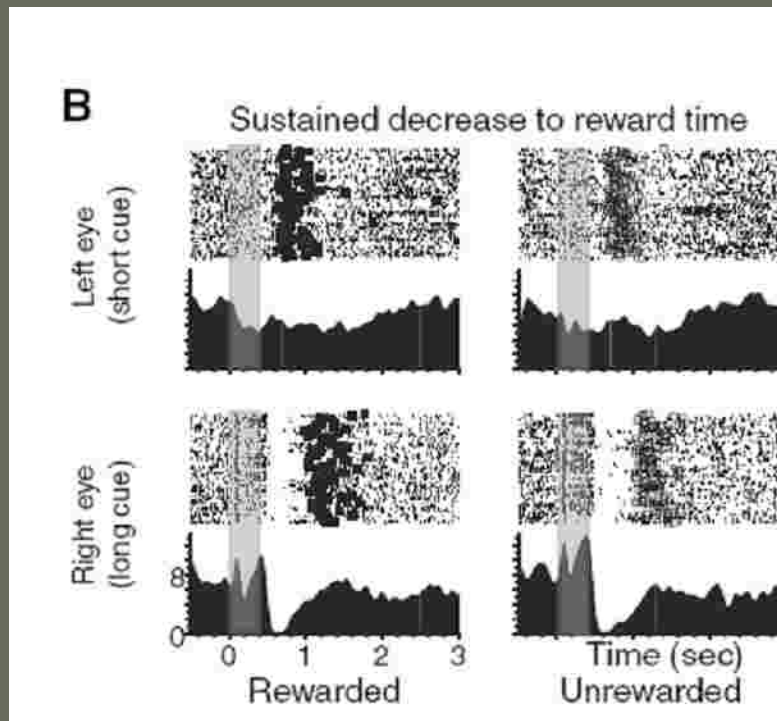
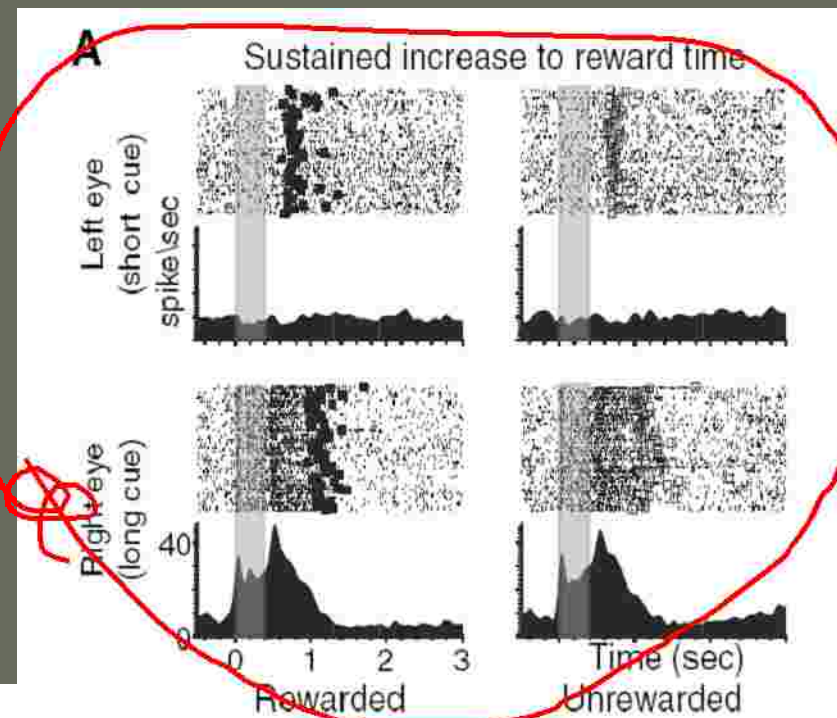
So that:  $a = \frac{1}{t_m} - \frac{1}{t_d}$

How can this be done? Choose:  $L_{ij} = \left(\frac{1}{t_m} - \frac{1}{t_d}\right)V_i V_j$

Can a network learn this rather than assume it is set externally?

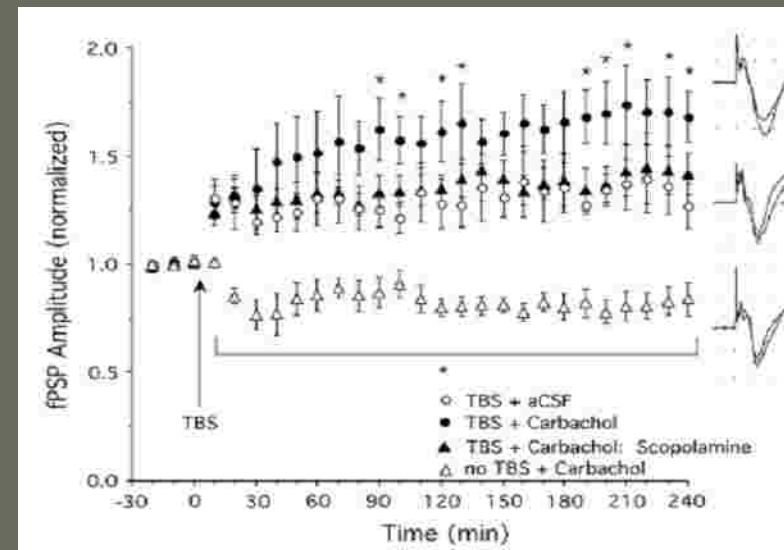
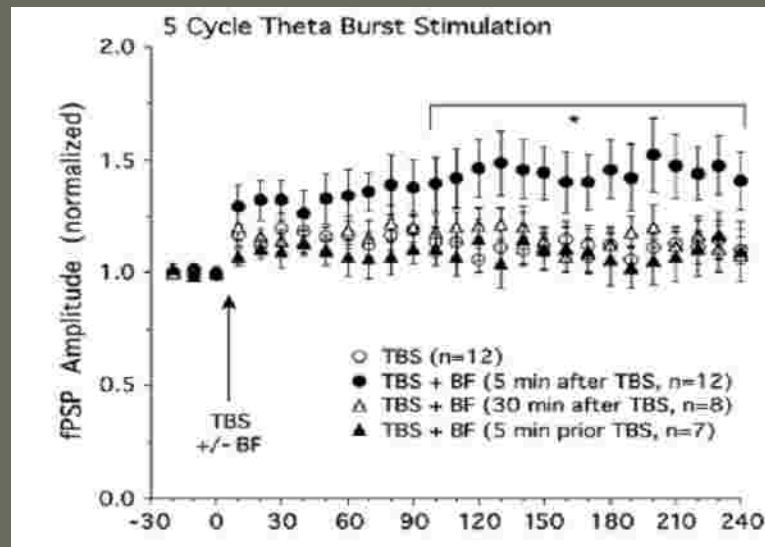
## Various response types:

1. ~ 43% of neurons showed reward timing after training
2. Of them 50% type A, 22% type B, 28% type C.



## 2. What is the reward signal?

- Dopamine – everyone's favorite reward signal, but
- Acetylcholine?



Dringenberg et. al. 2006

- Other Neuromodulators

# Training

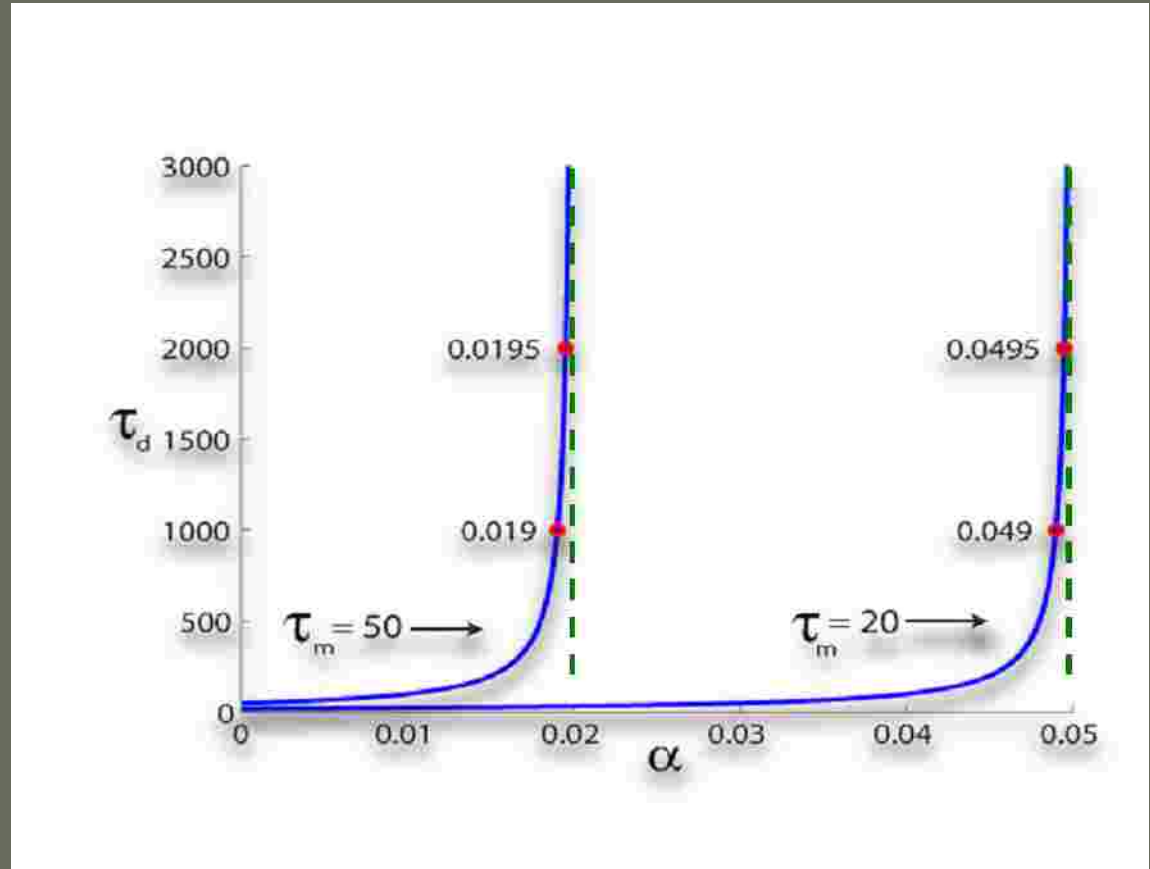
1400 ms  
reward

900 ms  
reward

Mean  
activity of  
whole  
network  
smoothed  
with a  
window of  
100 ms

However, this method has a significant problem:

$$a = \frac{1}{t_m} - \frac{1}{t_d}$$



For example: if  $t_m = 50$  ms,  $a(1000) = 0.019$ ,  $a(2000) = 0.0195$  and  $a(1000)/a(2000) = 0.974$  (2.5 % difference)

If  $t_m = 20$  ms the difference is 1%  
and if  $t_m = 200$  ms the difference is 11%

**What happens for  $a$  to right of the asymptote?**