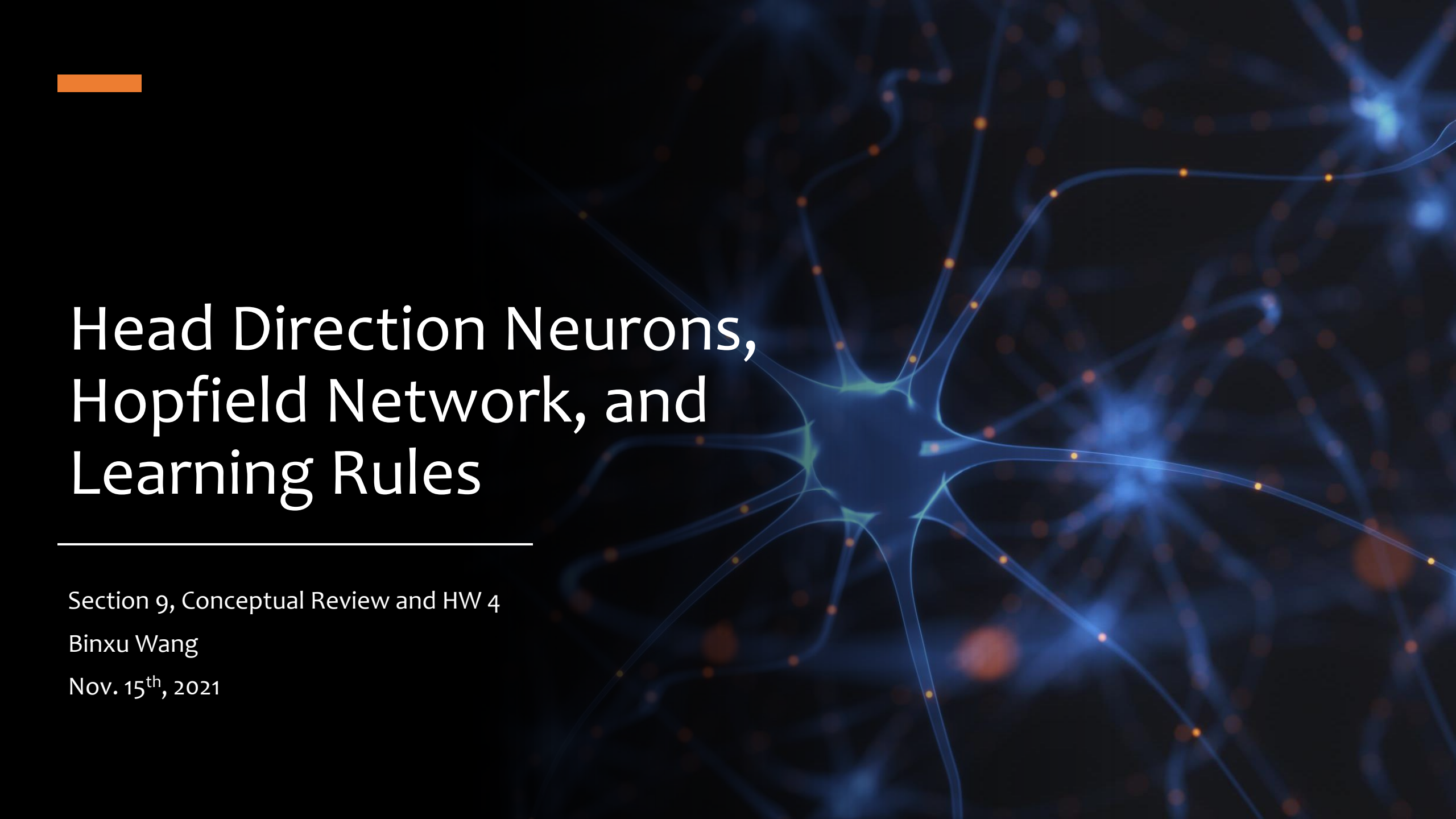


Head Direction Neurons, Hopfield Network, and Learning Rules

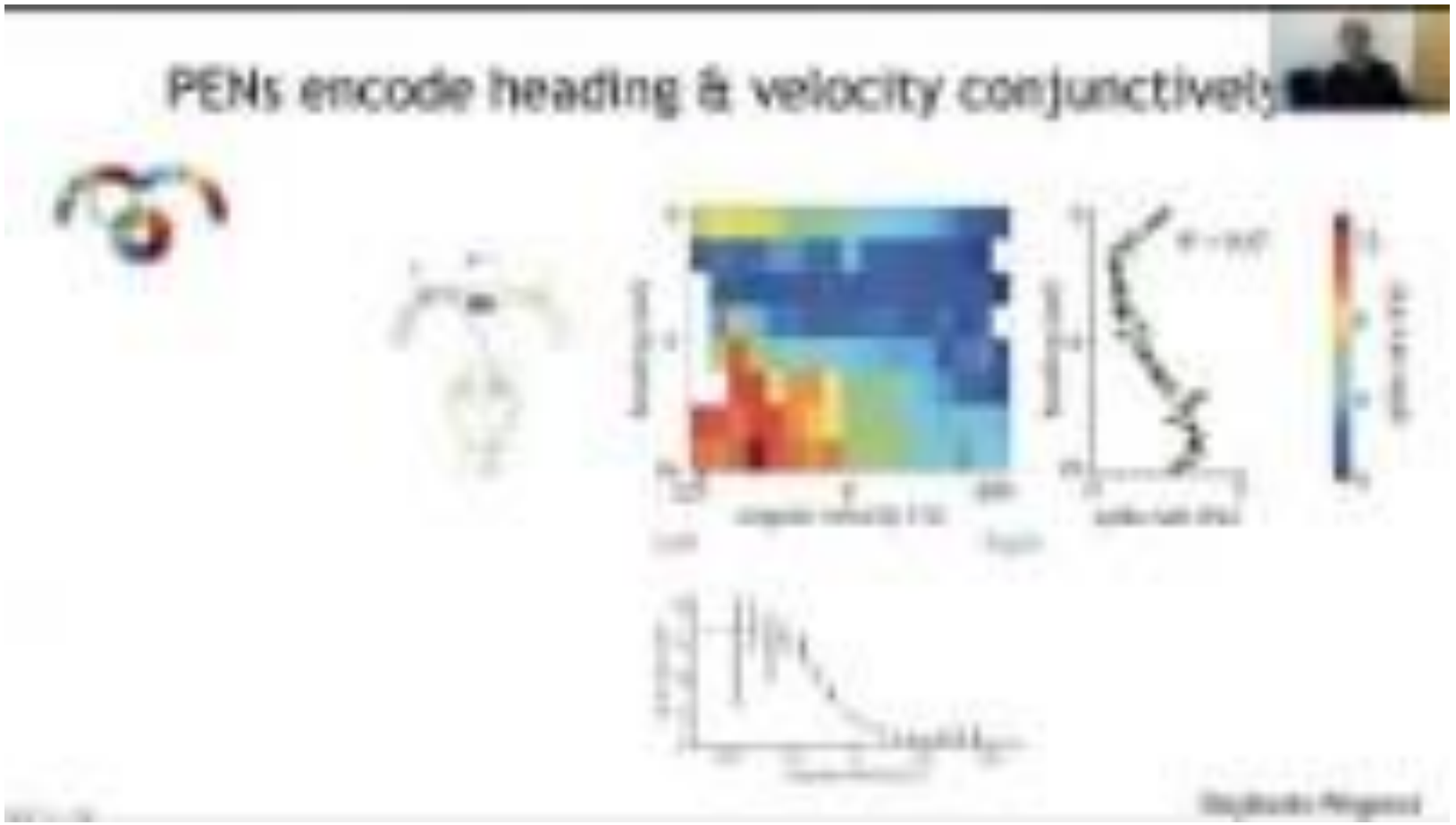
Section 9, Conceptual Review and HW 4

Binxu Wang

Nov. 15th, 2021

Model of Head Direction Neurons

Head Direction Neurons for Real in Fly



Can you see the connection to the continuous attractor network?

https://youtu.be/2fORQ7icGVU?t=325

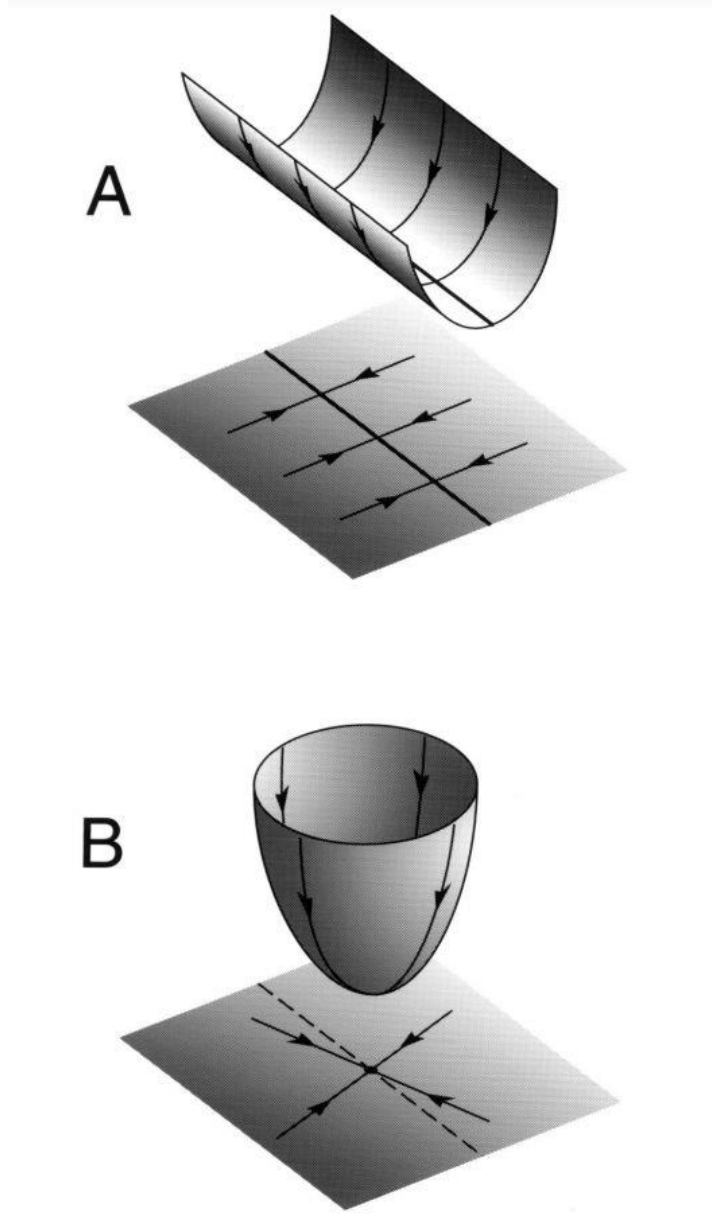
Head Direction Neuron: What to model?

- Desirable property
  - Stable when head is not moving (memory)
  - Track the change when the animal turn. (integrator)
  - Represent a single direction in $(0,2\pi)$ (single value)

Continuous Attractor Network

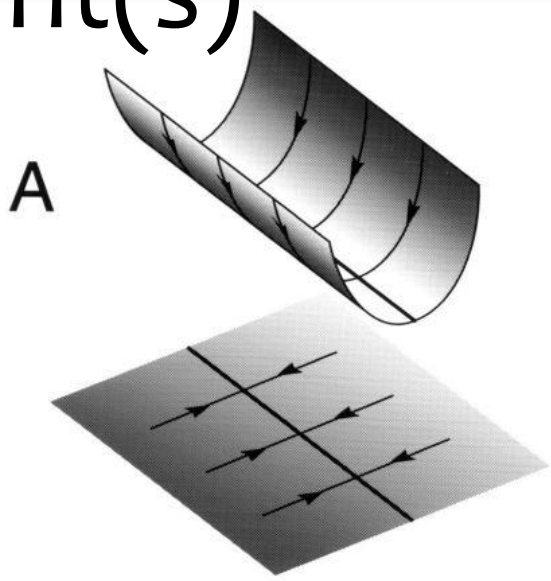
$$\frac{dv}{dt} = -v + Mv + Fu$$

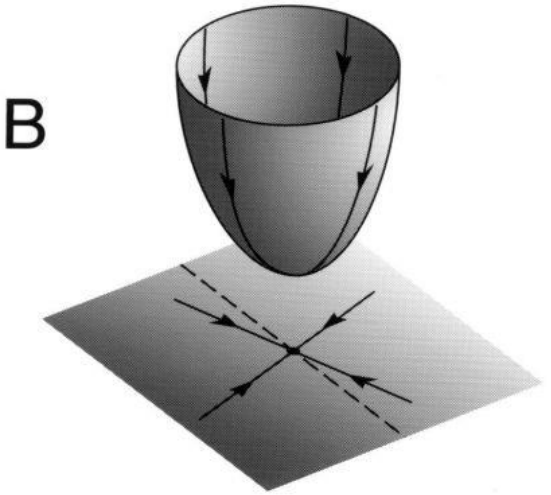
- What kind of M do we need for a continuous attractor?
  - Have more than 1d eigen space of $\lambda = 1$
  - Other eigenvalues with real part $\mathrm{Re}\lambda < 1$



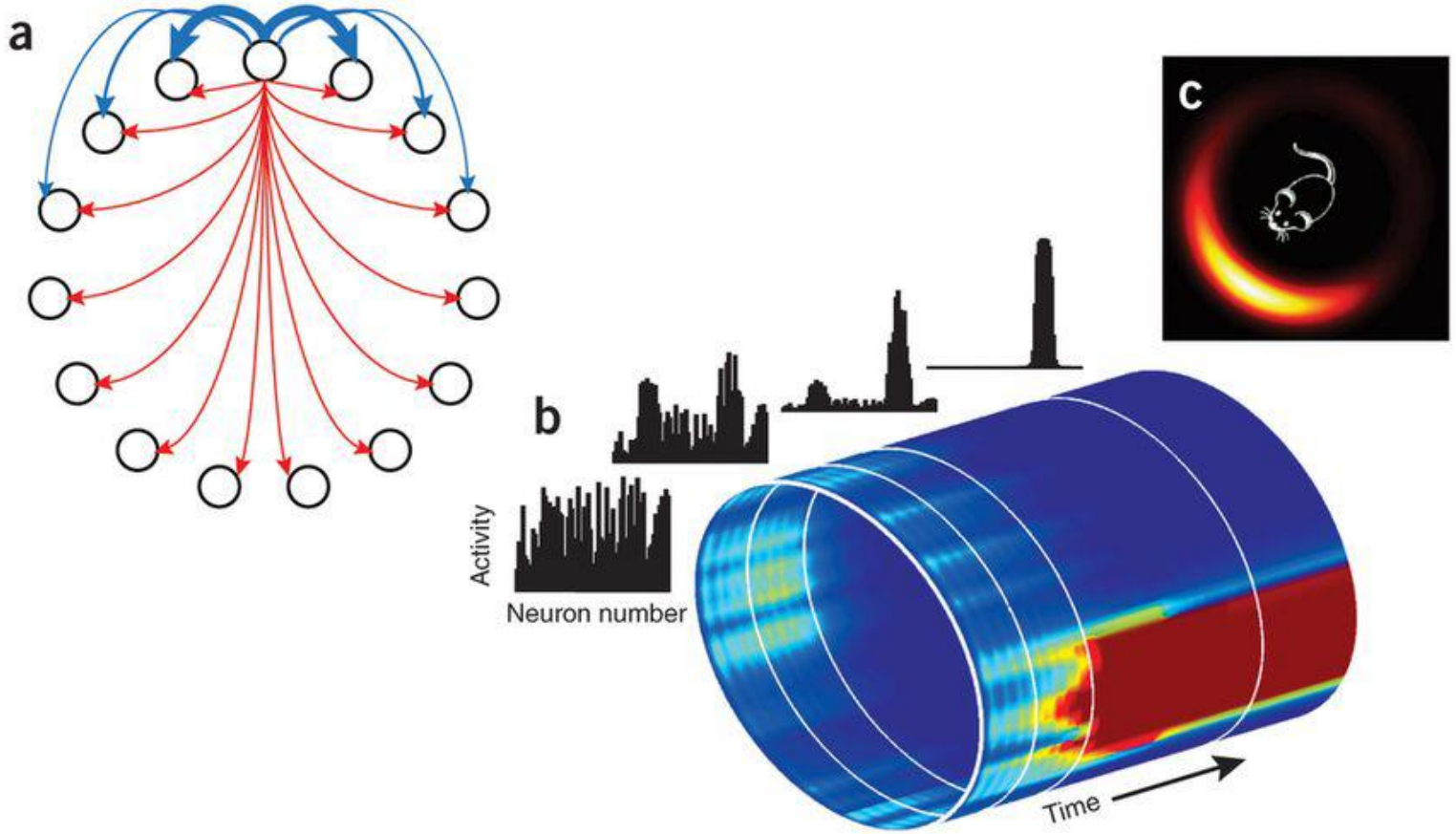
Linear Dynamic System: Fixed point(s)

- What's the dimensionality of the space of fixed points?
  - Dimensionality of the kernel space of $A = M - I$
  - Number of eigenvector with $\lambda = 1$ in M
- How many dimension do we need to represent **head direction**?
  - 1d?
  - 2d?
  - What's the dimensionality of the eigenspace with $\lambda = 1$?





Continuous Ring Attractor in Theory



Spectral Analysis of Recurrent Matrix

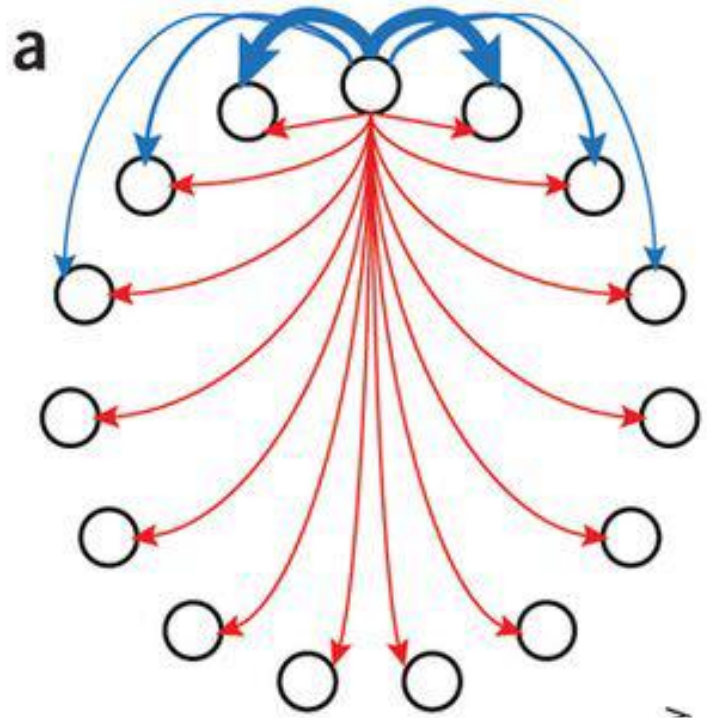
- Recurrent connection matrix
  - $\theta_i = (i-1)\,\Delta\theta, i = 1,2,\ldots N$
  - $M[i,j] \propto \cos(\theta_i - \theta_j)$

- All but 2 eigenvalues are zero.
  - Null space dim $N-2$
  - 2d eigenspace with the same value L

- This spectrum could be normalized to construct a ring attractor!

$$\frac{dv}{dt} = -v + \frac{1}{L}Mv$$

  - $M_{ij} = \cos(\theta_i - \theta_j)$

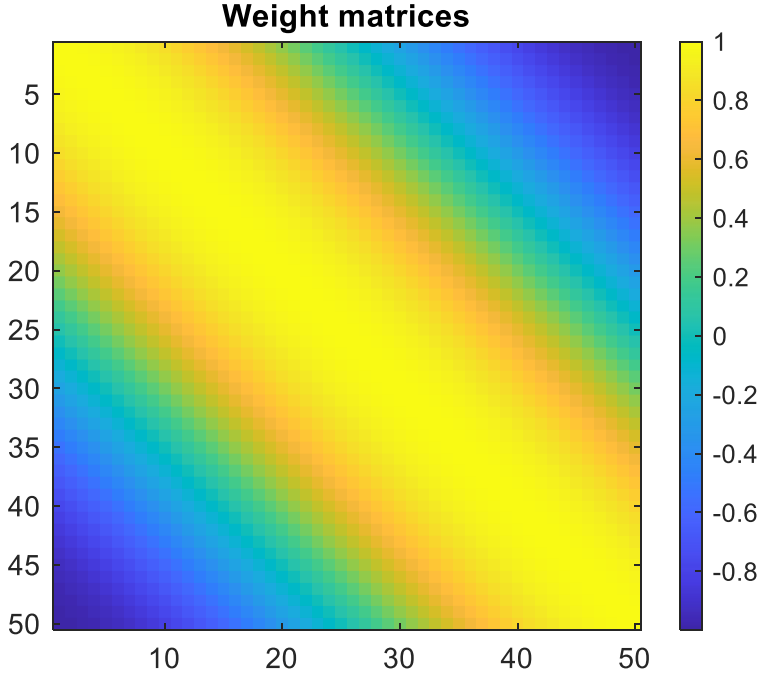


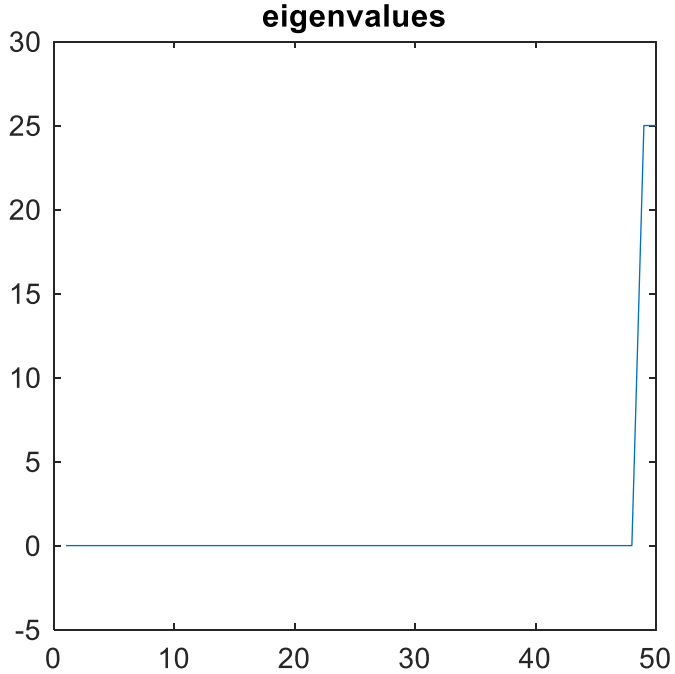
Eigen Structure of a Linear Ring Attractor Model

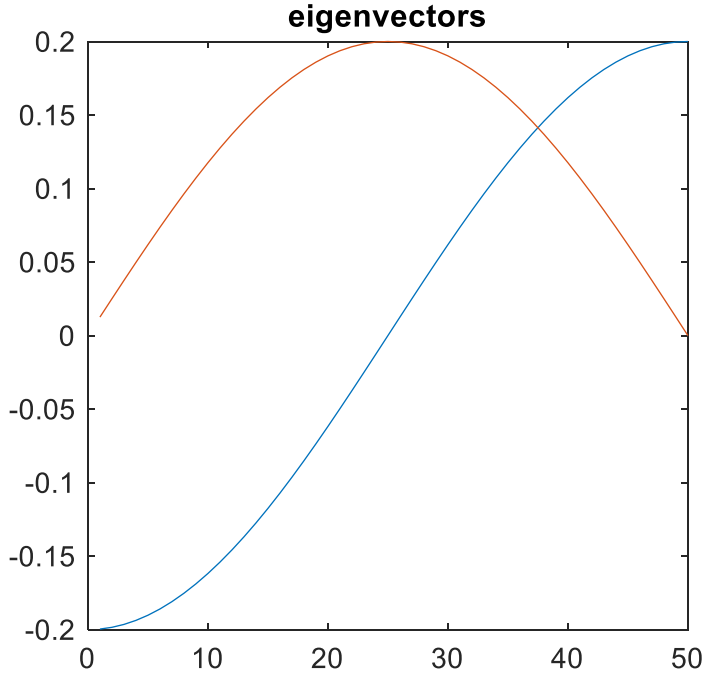
```
%Matlab simulation code
prefang = 0:pi/50:pi-0.0001;
angdiff = prefang - prefang';
wmat = cos(angdiff);
[evc,eva] = eig(wmat);
```

- Eigenvectors form a 2d space
  - sin and cos basis for the circular attractor.
  - Any $\sin(x+\phi)$ like bump could be a stable state.

- Is there a limit of how high a bump could be in this linear RNN? Will they be equally stable?
  - Yes







[Neuro120_tutorials/RingAttractorEigs.m at master · Animadversio/Neuro120_tutorials]

The only two non-zero eigenvector form a sin cos basis of the ring.

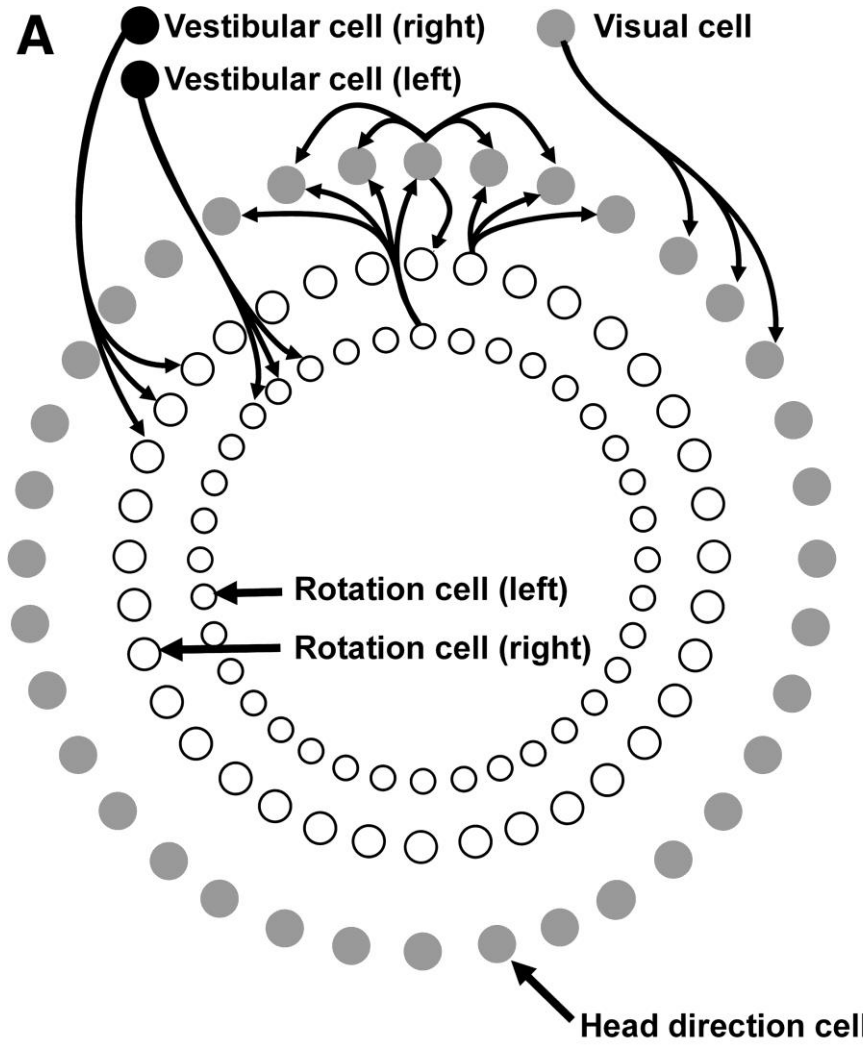
How Ring Attractor **Memorize the State** of Head Direction?

How Ring Attractor **Track the Change** in Head Direction?

How to design a circuit that takes in turning information and move the bump

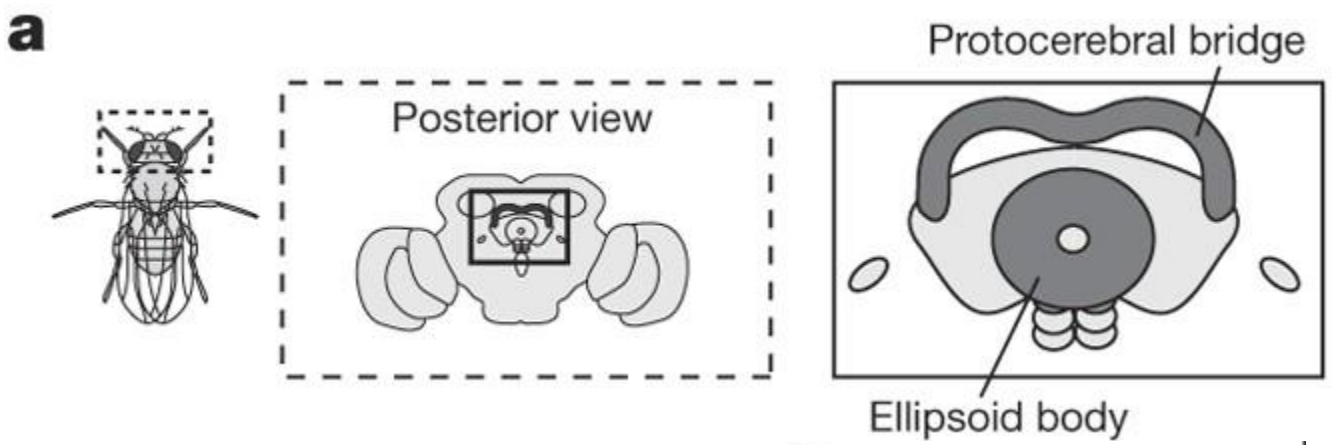
How the bump move when the animal turn: Model for Mouse

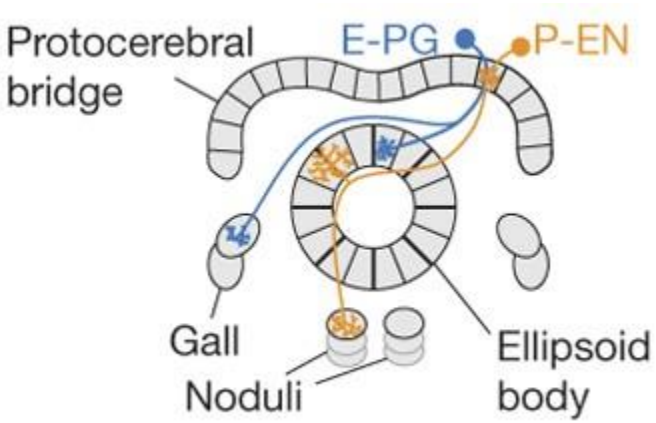
- Vestibular system sense acceleration
  - Push / pull the bump of activity along the attractor.
- Visual system find anchors for head direction.
  - Pin down certain directions.



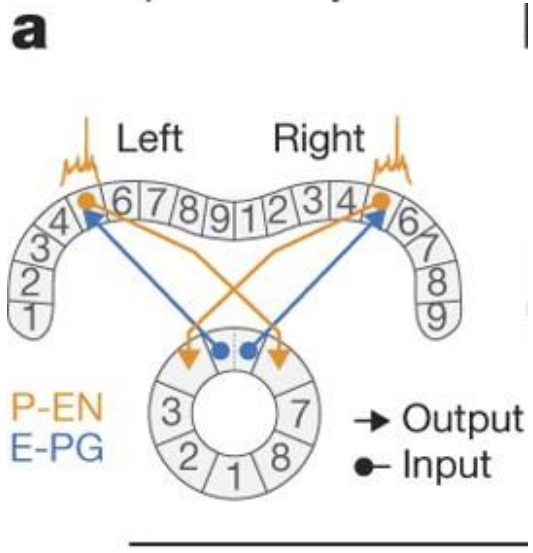
Skaggs, W. E., Knierim, J. J., Kudrimoti, H. S. & McNaughton, B. L. A model of the neural basis of the rat's sense of direction. *Adv. Neural Inf. Process. Syst.* **7**, 173–180 (1995) https://doi.org/10.1152/jn.00501.2017

How the bump move when the animal turn:
Circuit Anatomy for Fly

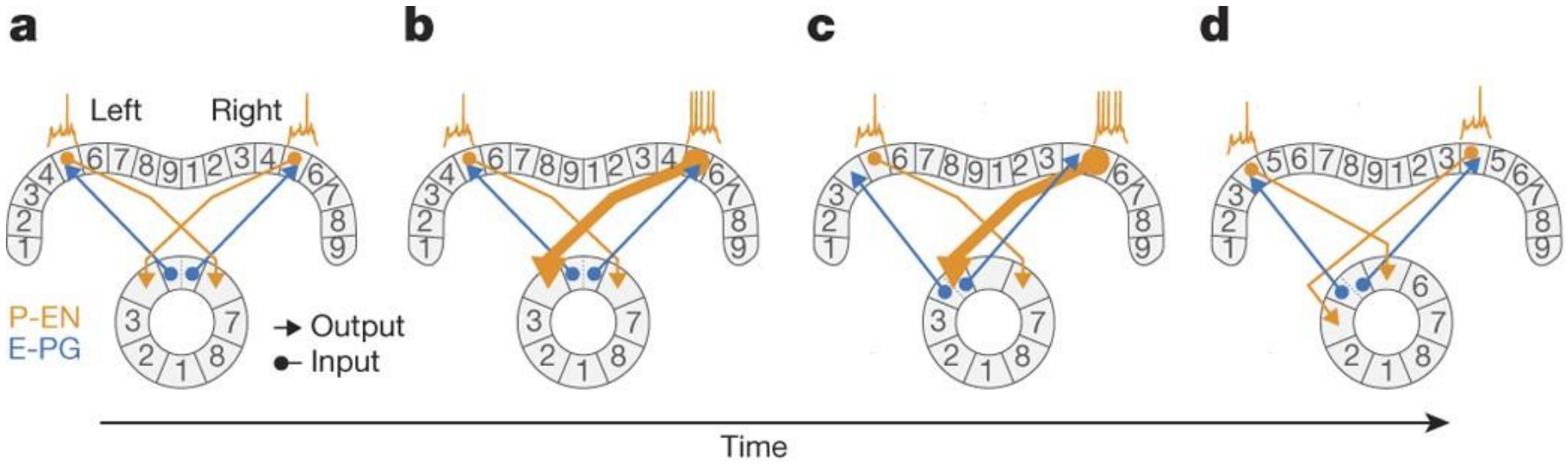




Anatomy of projections

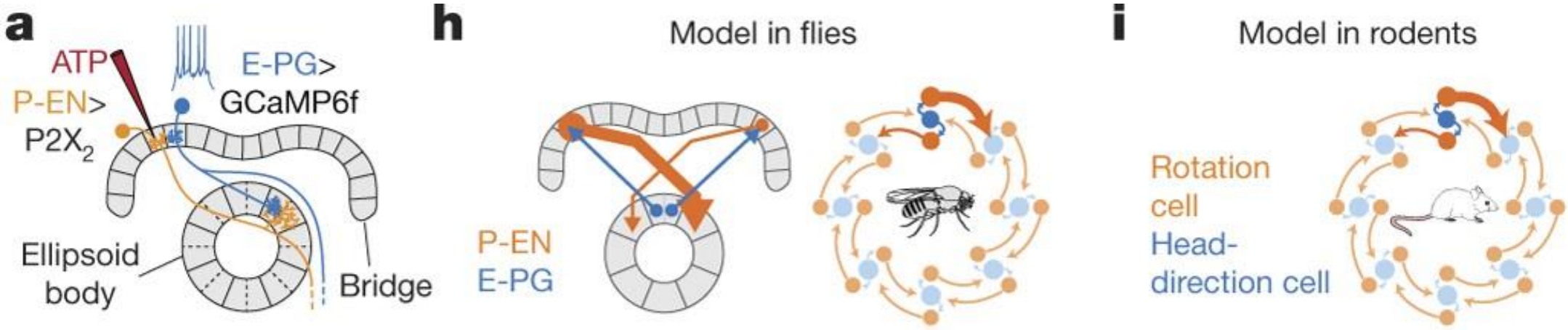


How the bump move when the animal turn: Model and Data for Fly

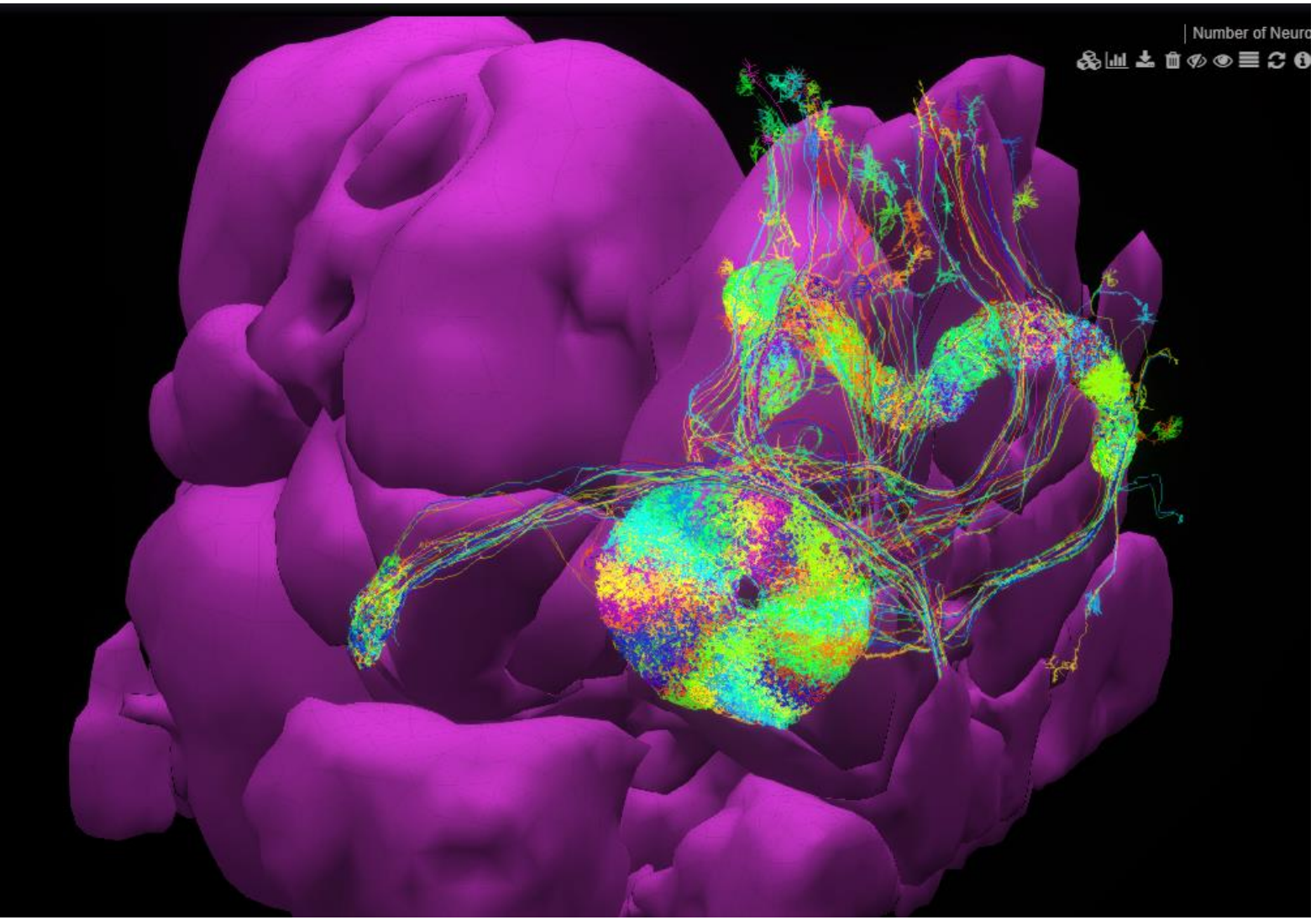


A neural circuit architecture for angular integration in Drosophila | Nature 2017

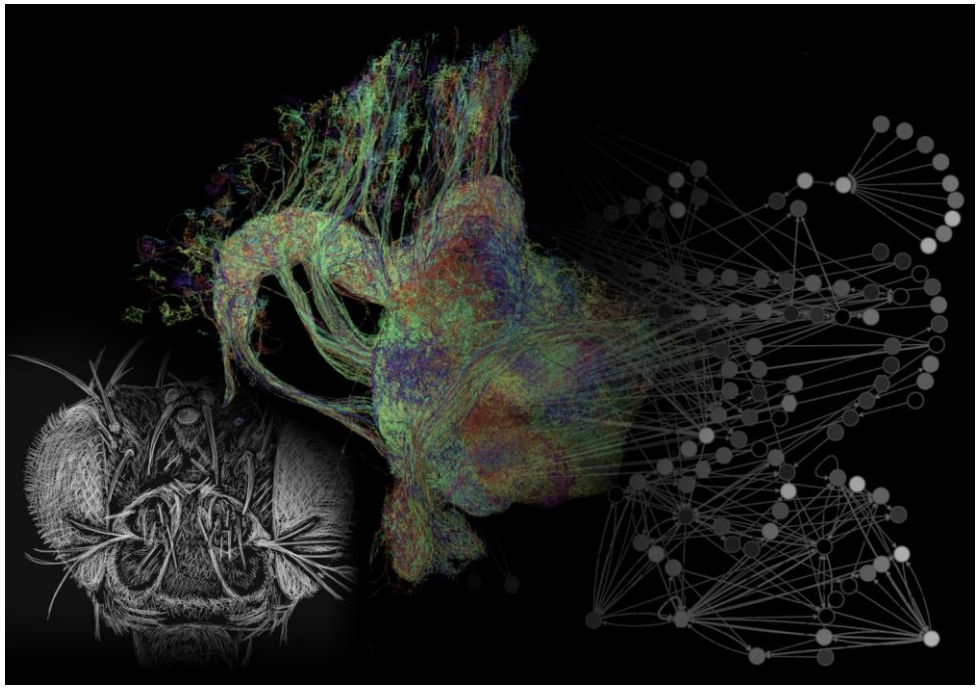
Striking Consistence between Model and Reality



This Circuit is Breathtakingly Beautiful.



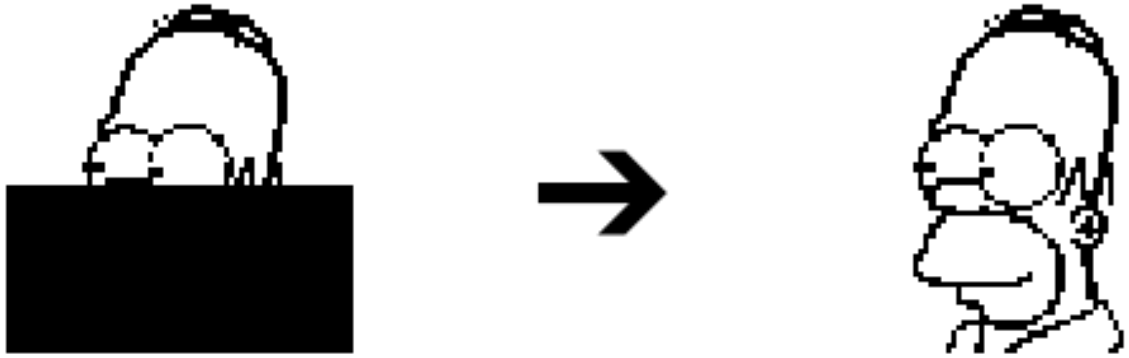
NeuroNLP.Hemibrain (fruitflybrain.org)



Hopfield Network

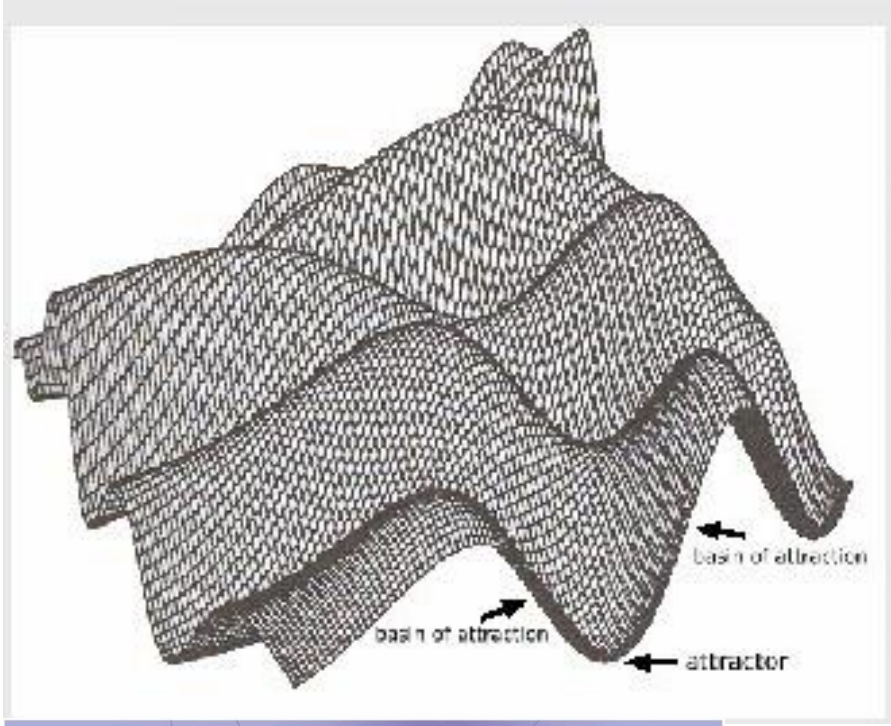
Desirable Properties of Long-Term Memory Model

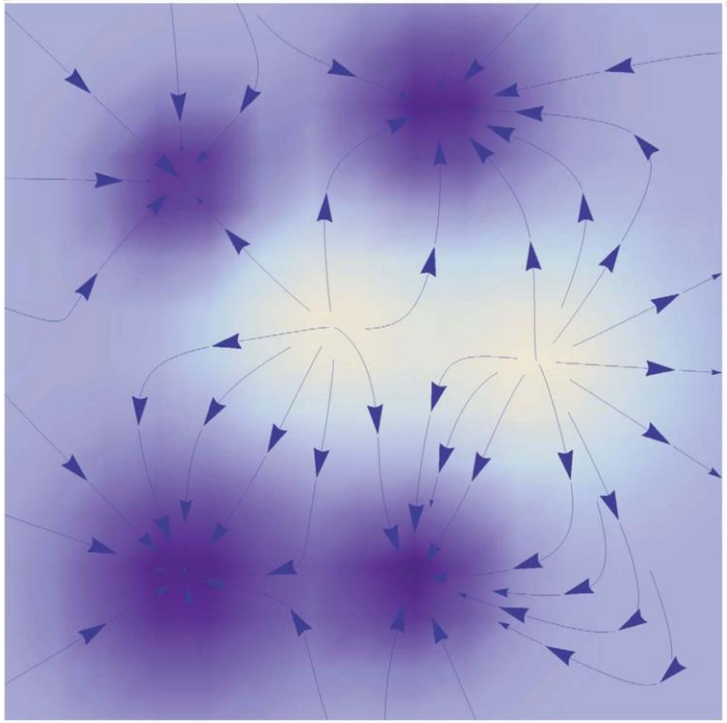
- Content addressable
  - Part of pattern -> Whole pattern
  - Could recover from corruption, noise resistant.
  - Auto-associative



Insights of Hopfield Network

- Memorized patterns as attractors states.
- Memory retrieval as converging to attractor.
  - The dynamics / update rule follows an energy landscape.
  - Memory stores at the maximum / minimum





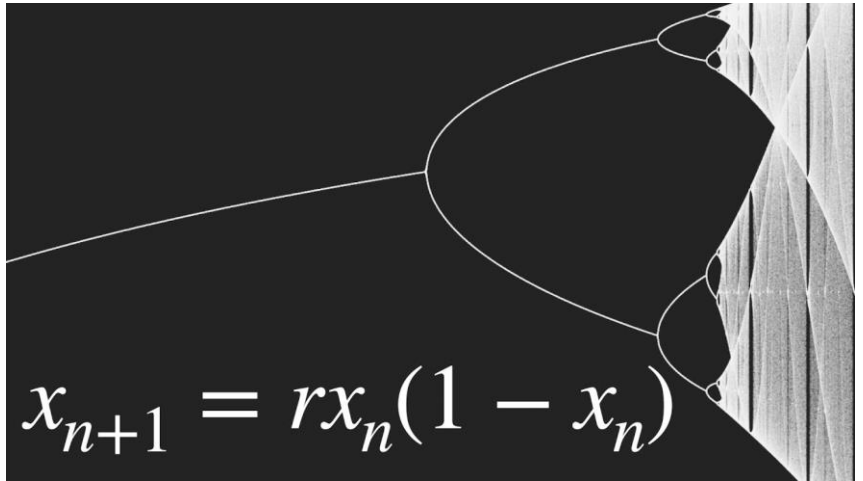
Discrete vs Continuous Dynamic System

Continuous System

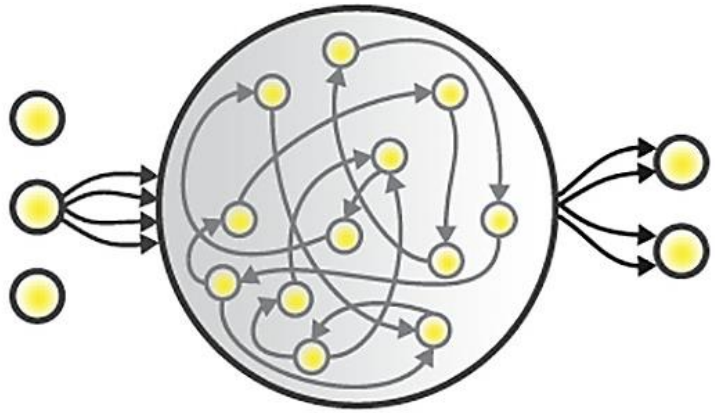
- A differential equation
$$\frac{dx}{dt} = f(x)$$
- Progress by integration
$$x(t) = x(0) + \int f\big(x(\tau)\big)d\tau$$
- Could be approximated by discrete system
$$x[t+1] = x[t] + f(x[t])$$

Discrete System

- A mapping
$$X[t+1] = F(X[t])$$
- Progress by iteratively applying the map
$$X[t+1] = F(F(F(\dots F(X[0]))))$$



Hopfield Network vs RNN



Recurrent neural network

$$\frac{dv}{dt} = -v + F(Mv)$$

- Continuous time
- Neuron:
  - Continuous firing rate
  - Nonlinear function.
- Connectivity

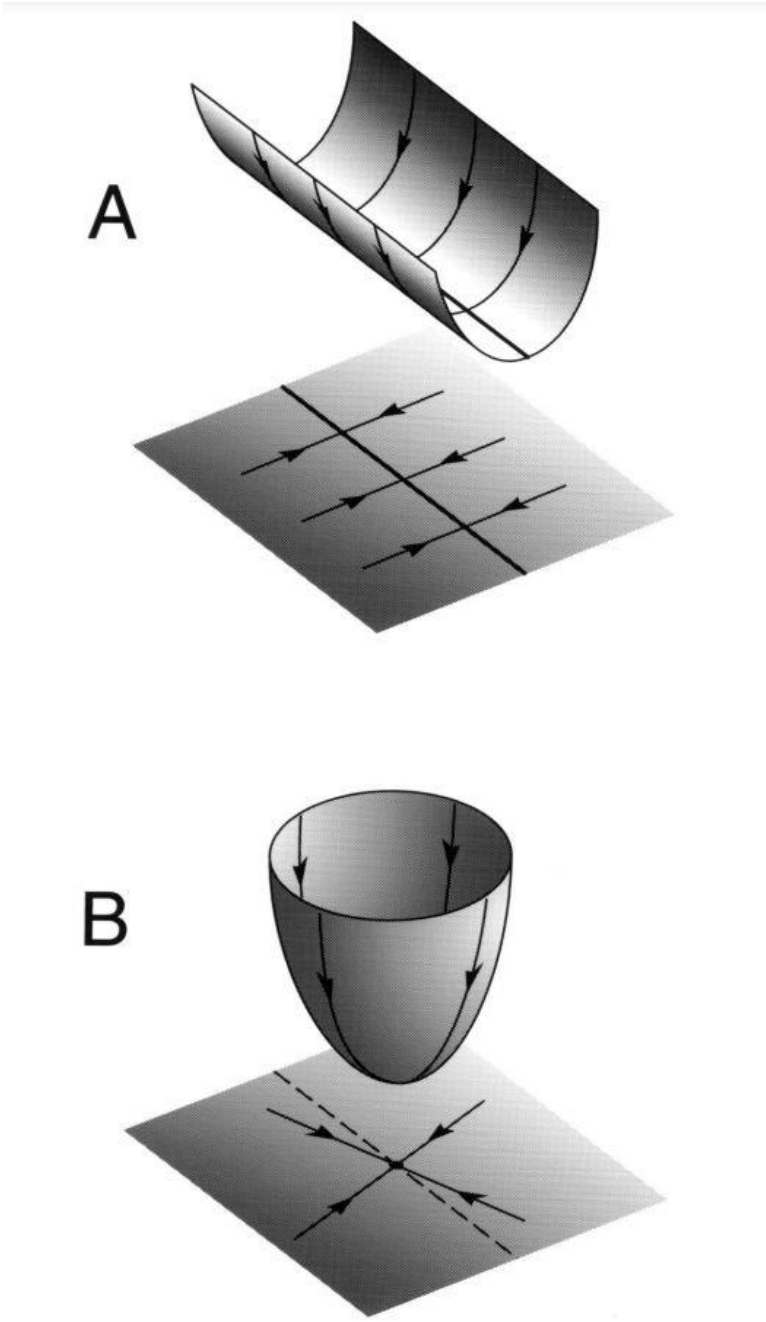
Hopfield network

$$v[t+1] = F(Mv[t])$$

- Discrete time
- Neuron:
  - Binary state, fire or not (1,-1)
  - Threshold function
- Connectivity
  - Symmetric $W_{ij} = W_{ji}$ (or no?)
  - No self connection $W_{ii} = 0$

Could we model Long-Term Memory by Linear RNN?

- Or can Linear RNN store patterns in its attractors?
  - No, recall that linear RNN has 1 or infinite stable state.
  - All the stable fix points are linearly connected.
  - You don't want a memory to interpolate two.



How to Learn / Build a Hopfield Network

How to set a state as attractor / stable fixed point?

Energy Function of a Hopfield Network

$$E(v) = -v^T M v$$

- Energy change with turning off / on a neuron

$$\Delta E = E(v[k] = 1) - E(v[k] = -1)$$
$$= -4 \sum_j M[k,j]v[j] + 4M[k,k]$$
$$= -4 \sum_j M[k,j]v[j]$$

Recall the dynamics of Hopfield network

$$v[k] = F\left(\sum_j M[k,j]v[j]\right)$$

- Neuron will activate / inactivate to make this energy *lower*.

$$v[k] = \begin{cases} -1, & \Delta E > 0 \\ 1, & \Delta E < 0 \end{cases}$$

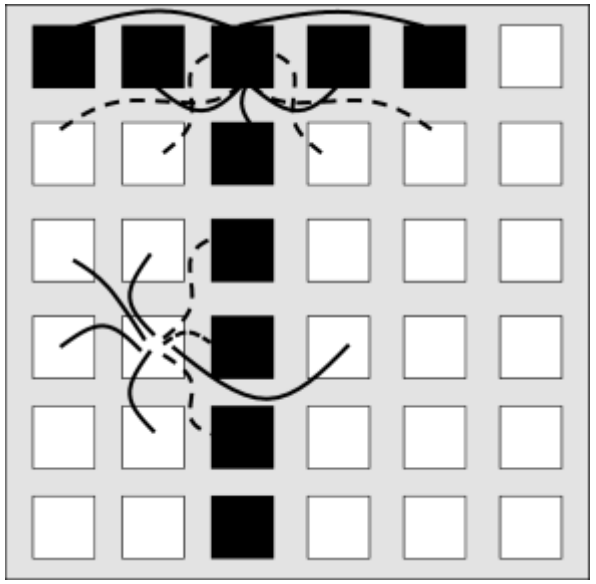
Memory patterns x need to have highest Energy!

$$E(x) = x^T M x$$

Demo: Manual Encoding a Single Memory

- If we have a target pattern x
- We can design a recurrent matrix.

$$M = xx^T$$



- What is the eigenvalue and vector of this matrix?
  - 0, N
  - x is eigenvector (not normalized)

- How to interpret it ?

$$M_{ij} = x[i]x[j] = \begin{cases} 1 & \text{if } x[i] = x[j] \\ -1 & \text{if } x[i] \neq x[j] \end{cases}$$

  - Neurons having same values in the pattern exciting each other.
  - Neurons with opposite values in the pattern inhibiting each other.

Demo: Manual Encoding a Single Memory

- If we have a target pattern x
- We can design a recurrent matrix.

$$M = xx^T$$

- Then given any initial state $v[0]$

$$v[1] = F(Mv[0])$$
$$= F\big((x^T v[0])x\big)$$

$$v[1] = \begin{cases} x & \text{if } x^T v[0] > 0 \\ -x & \text{if } x^T v[0] < 0 \end{cases}$$

Given a single memory, any initial state will converge to that memory or it's antipodes memory !

A good start

Demo: Encoding Multiple Memories

- If we have multiple target pattern $\{x_i\}$
- We can still design a recurrent matrix.

$$M = \sum_i x_i x_i^T$$

- Then given any initial state $v[0]$

$$v[1] = F(Mv[0])$$

$$= F\left(\sum_i \left(x_i^T v[0]\right) x_i\right)$$

$$= F\left(\sum_i c_i x_i\right), c_i = x_i^T v[0]$$

  - c_i quantifies similarity of input with a pattern x_i
  - The more similar a pattern is the more weight it has on the outcome.

$$M = \sum_i x_i x_i^T - I$$

General lesson

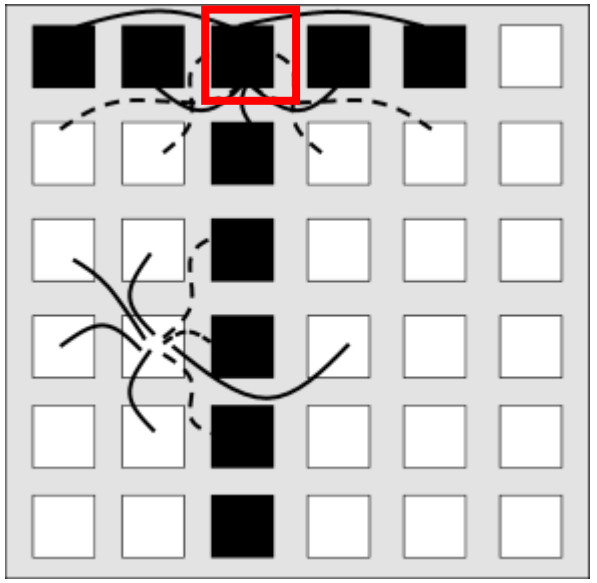
- Non-correlated memory patterns are easier to store, less likely to interfere.
- Similar / correlated memory pattern can interfere and create amalgamates.
- Not necessarily converge to the good memory.

Hopfield network as an Array of Perceptrons

- In Hopfield Network
  - Each unit is using the activation of other units to predict / classify its own value, same as perceptron!

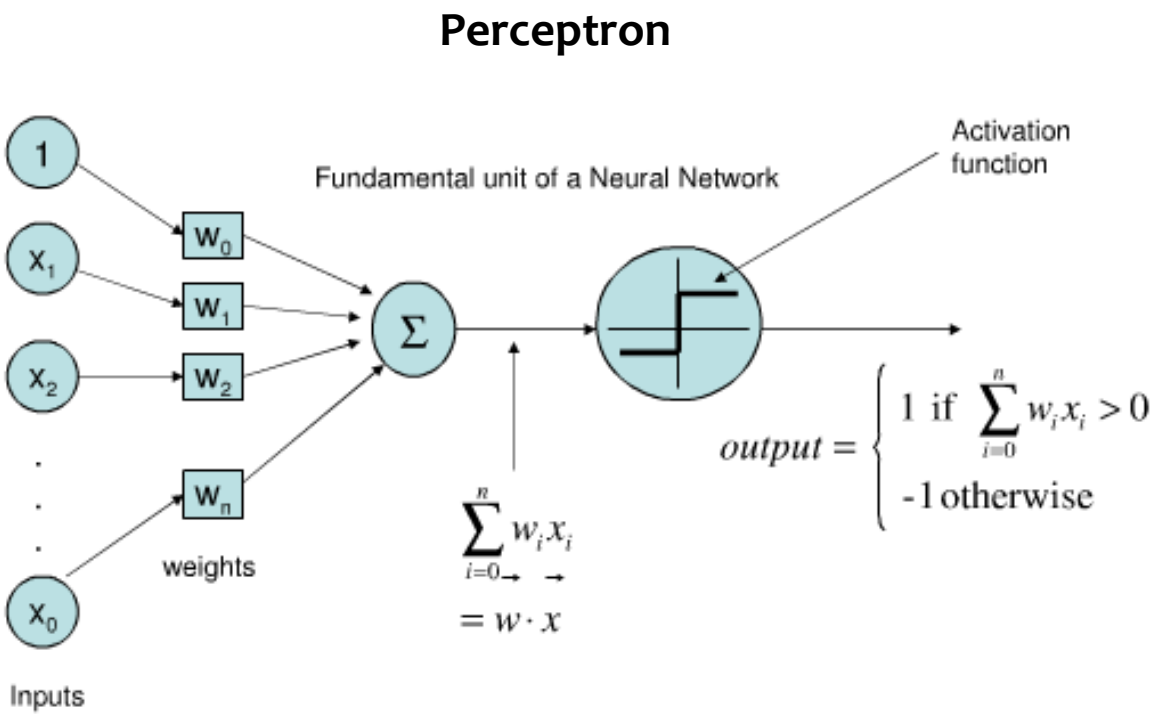
$$v[i] = F(M[i,:]v)$$

  - All the patterns form its training data. $\{x_i\}$
  - Training Objective:
    - All the units need to correctly predict its value on all the patterns.



This philosophy is similar to the modern transformer architecture. Deeply related.

Digression: How perceptron learn?



- If we want a perceptron to learn a mapping $\{x_i, y_i\}, y_i \in \{-1,1\}$

$$y = f(w^T x)$$

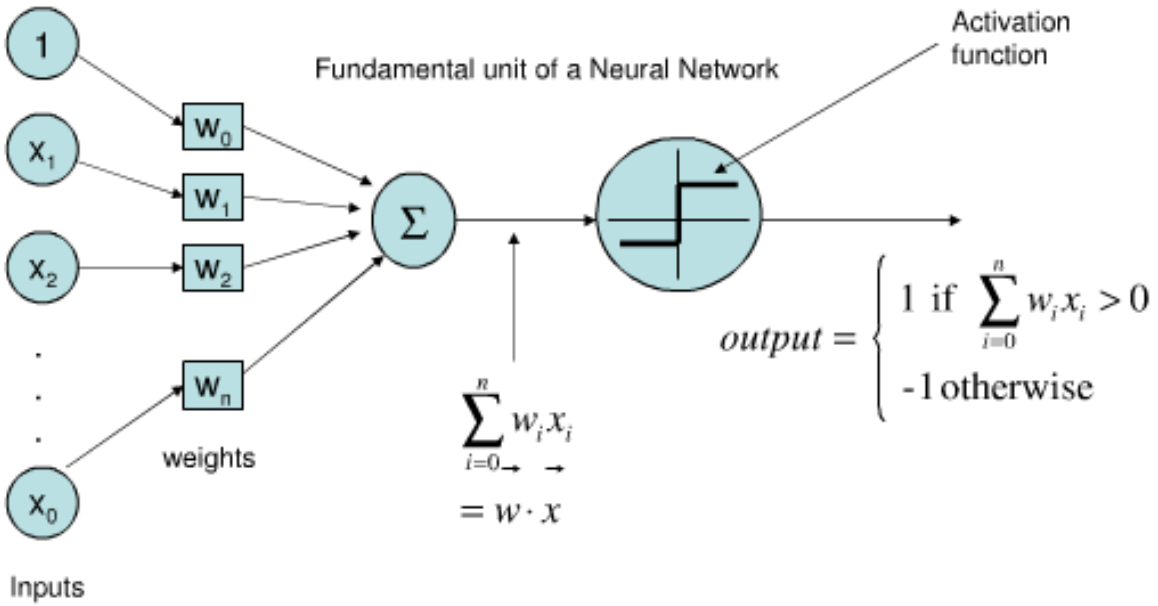
- How should we set the weights?

Perceptron Algorithm

Initialize w
while not converge
    for i = 1 : N
        If $w^T x_i > 0$ and $y_i < 0$
            $w \leftarrow w - \eta x_i$
        If $w^T x_i < 0$ and $y_i > 0$
            $w \leftarrow w + \eta x_i$

Digression: How perceptron learn?

Perceptron

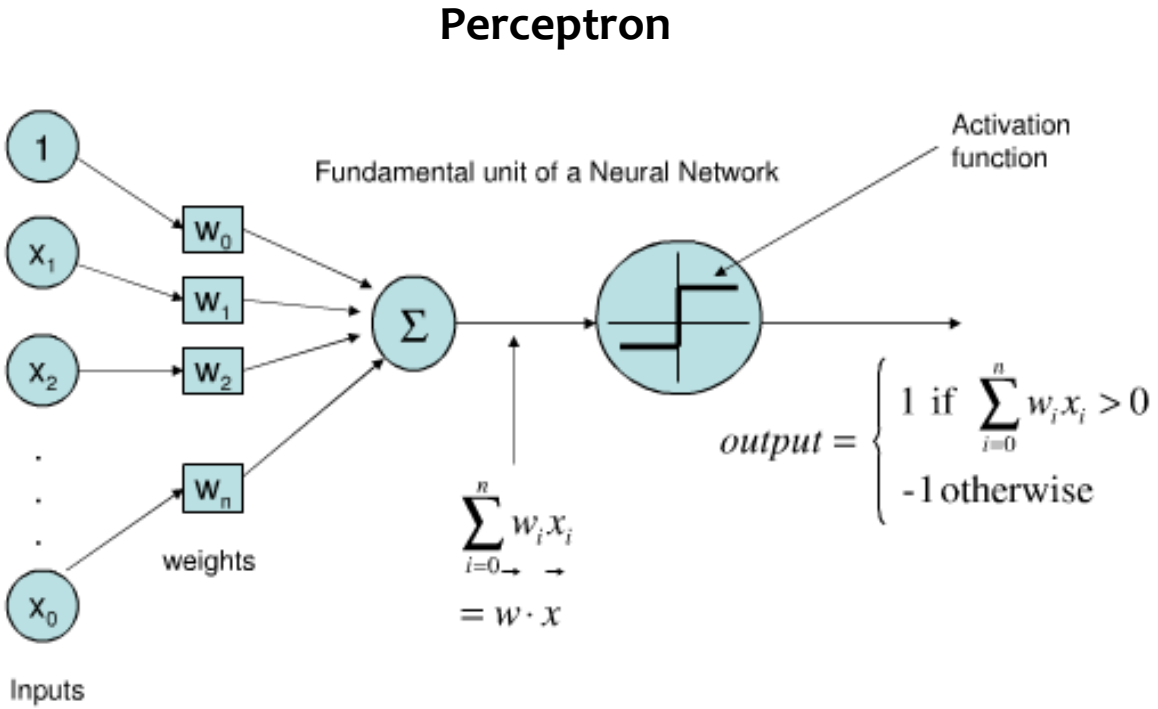


- This algorithm can be written more concisely

Perceptron Algorithm

Initialize w
while not converged
    for i = 1:N
        If $y_i \cdot (w^T x_i) < 0$
            $w \leftarrow w + \eta y_i x_i$

Digression: How perceptron learn?



- This algorithm can be written in a vectorized fashion
  - K datapoints, N neurons
  - $X \in \mathbb{R}^{K \times N}, \boldsymbol{y} \in \mathbb{R}^{K \times 1}, w \in \mathbb{R}^{N \times 1}$
  - $x_i = X[i,:], y_i = \boldsymbol{y}[i]$

Perceptron Algorithm (Vectorized)

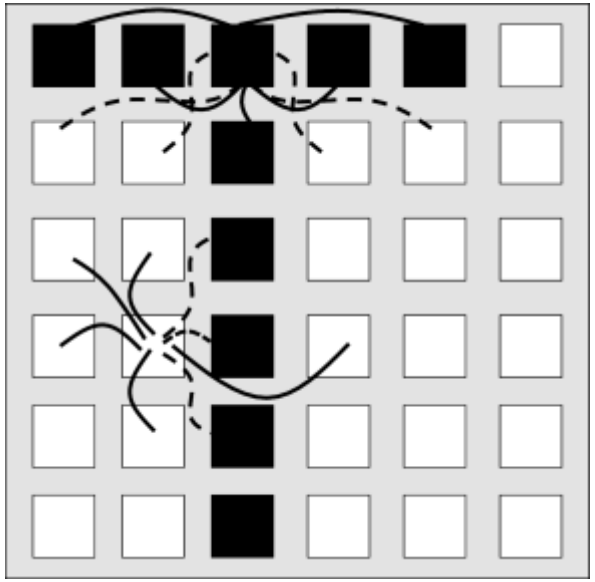
Initialize w
while not converged
Compare prediction Xw and target y
Find errored samples
$$\text{mask} = y \odot (Xw) < 0$$
Update the weights by these samples
$$w^T \leftarrow w^T + \eta(y \odot \text{mask})^T X$$

$\odot$ denotes element wise multiplication.
0 could be substitute by positive number to create a larger margin.

Hopfield network as an Array of Perceptrons

- Similar to perceptron, for each neuron
  - K datapoints / patterns, N neurons
  - $X \in \mathbb{R}^{K \times N}, \boldsymbol{y} \in \mathbb{R}^{K \times 1}, w \in \mathbb{R}^{N \times 1}$
  - $x_i = X[i,:], y_i = \boldsymbol{y}[i]$

- Only difference is to learn N perceptrons at the same time
  - The output is the same as input.
  - $X \in \mathbb{R}^{K \times N}, y = X \in \mathbb{R}^{K \times N}, M \in \mathbb{R}^{N \times N}$



Perceptron Learning Algorithm for Hopfield Network

- Generalized Perceptron learning rule
  - Target: Making the target pattern fixed points
$$v = F(Mv)$$
  - In HW4, pattern is encoded in $0, 1$
  - Input strength before nonlinearity.
$$\text{input} = Mv - 1/2$$
  - Input strength on the correct direction.
$$(\text{input} \odot (2v - 1))$$
  - Find errored unit with a margin
$$\text{errunit} = \text{input} \odot (2v - 1) < \kappa$$
  - Update errored units associated with their input.
$$\Delta M = [\text{errunit} \odot (2v - 1)] \cdot v^T$$
$$M \leftarrow M + \eta \Delta M$$

Ways to Design Hopfield Network Weights

Error Driven Learning

- Methods
  - Generalized Perceptron Learning
  - Backpropagation
- Cons:
  - Takes longer time to train. Harder to implement. More tuning.
  - No analytical solution.
- Pros
  - Better in storage and retrieval.

Prescription / Hand-Design

$$M = \sum x_i x_i^T$$

- Pros:
  - Easy to implement, one-shot setup.
  - Analytically simple, some theoretical results.
- Cons:
  - Not very good to store practical data.

Hopfield Network is classic and not dead…

- Some modern variations of Hopfield Network
  - [1606.01164] Dense Associative Memory for Pattern Recognition Dmitry Krotov, John J Hopfield
  - [2008.06996] Large Associative Memory Problem in Neurobiology and Machine Learning Dmitry Krotov, John Hopfield
  - [2008.02217] Hopfield Networks is All You Need
  - Hopfield Networks is All You Need | hopfield-layers (ml-jku.github.io)

- Major development:
  - The RNN interpretation is not as important as the energy interpretation.
  - Modern Hopfield network, discard the form of RNN but add variations to the energy function $E(v)$

$$E(v) = \sum_i F(x_i^T v)$$