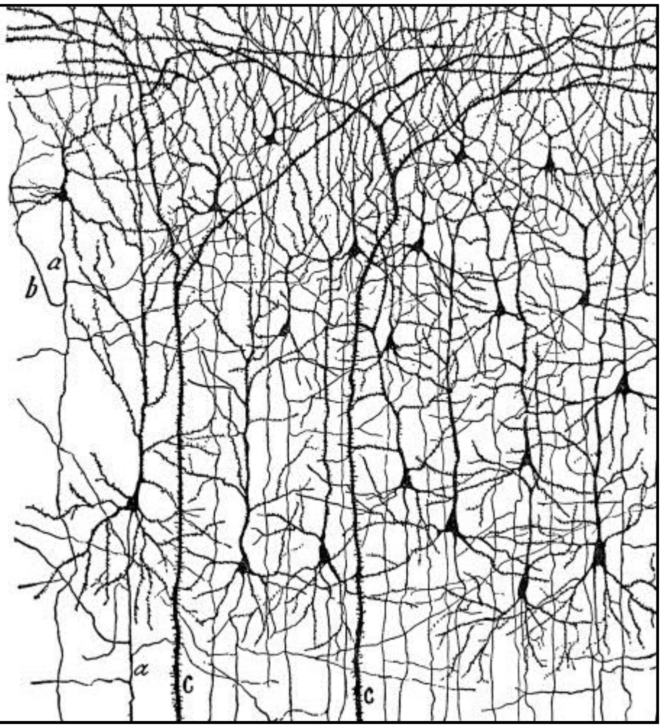
CSE/NEUBEH 528

Networks of Neurons

(Chapter 7)

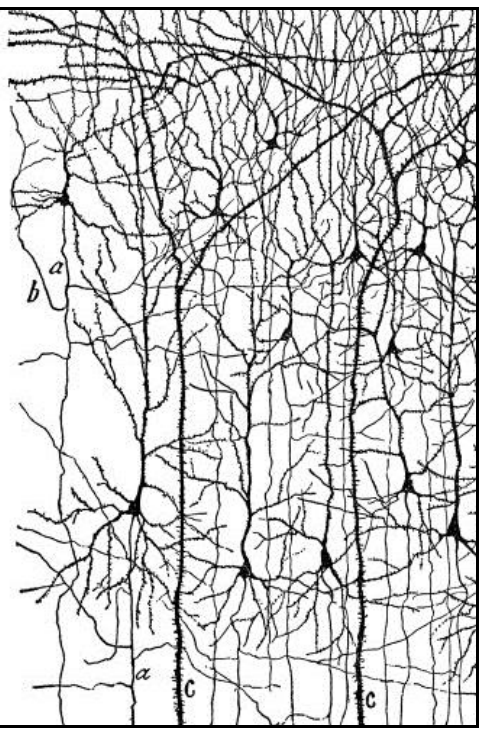


R. Rao, 528: Lecture 9

Drawing by Ramón y Cajal

Today's Agenda

- Computation in Networks of Neurons
  - Feedforward Networks: What can they do?
  - Recurrent Networks: What more can they do?



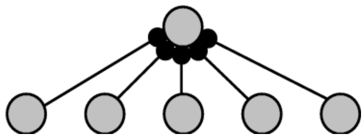
R. Rao, 528: Lecture 9

Flashback Firing-Rate-Based Network Model

output v

weights $\mathbf{w}$

input $\mathbf{u}$



F is the "activation function"

Output firing rate changes like this: $\tau_r \frac{dv}{dt} = -v + F(I_s(t))$

Input current changes like this: $\tau_s \frac{dI_s}{dt} = -I_s + \mathbf{w} \cdot \mathbf{u}$

What happens when:

$\tau_s << \tau_r$?

$\tau_r << \tau_s$?

Static input?

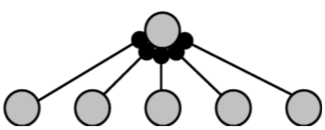
What if there are multiple output neurons?

Single Output

Scalar v

Vector $\mathbf{w}$

Vector $\mathbf{u}$



Input Vector

$$\tau \frac{dv}{dt} = -v + F(\mathbf{w} \cdot \mathbf{u})$$

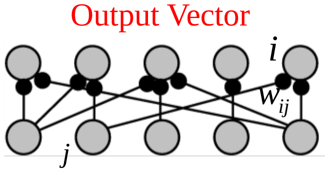
(Assuming relatively fast synapses, $I_s = \mathbf{w} \cdot \mathbf{u}$ at each t)

Output Vector

Vector $\mathbf{v}$

Matrix W

Vector $\mathbf{u}$



Input Vector

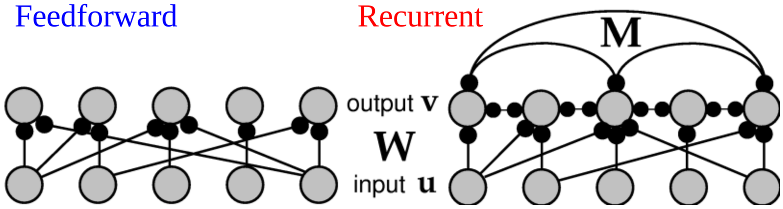
$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + F(W\mathbf{u})$$

$\mathbf{v} : N \times 1$ vector; $\mathbf{u} : K \times 1$ vector

$W : N \times K$ matrix; F : pointwise function

General Equation for Modeling Networks

Feedforward Recurrent



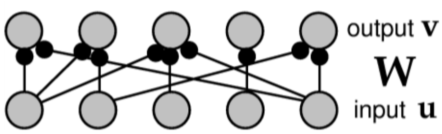
$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + F(\mathrm{W}\mathbf{u} + \mathrm{M}\mathbf{v})$$

Output Decay Input Feedback

For feedforward networks, M = matrix of zeros

Image Source: Dayan & Abbott textbook

Example: Linear Feedforward Network



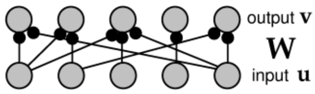
Dynamics: $\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + \mathrm{W}\mathbf{u}$

Steady State (set $d\mathbf{v}/dt$ to 0): $\mathbf{v}_{ss} = \mathrm{W}\mathbf{u}$

$$\mathrm{W} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}$$

What is $\mathbf{v}_{ss}$?

Linear Feedforward Network



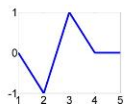
$$\mathbf{v}_{ss} = W\mathbf{u} = \underset{6\times 5}{\begin{bmatrix} 1 & 0 & 0 & 0 & -1 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix}} \underset{5\times 1}{\begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix}} = \underset{6\times 1}{\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}}$$

What is the network doing?

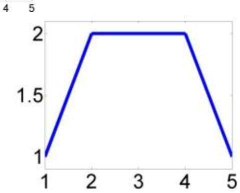
Network is performing Linear Filtering for Edge Detection

$\text{Filter} = [0 \quad -1 \quad 1 \quad 0 \quad 0]$

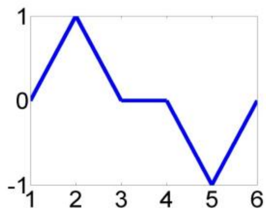
(and shifted versions in W)



$\text{Input } \mathbf{u} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \end{bmatrix} \quad \text{Output } \mathbf{v} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}$



Input **u**



Output **v**

Example of Edge Detection in a 2D Image

Input **u** Output **v**

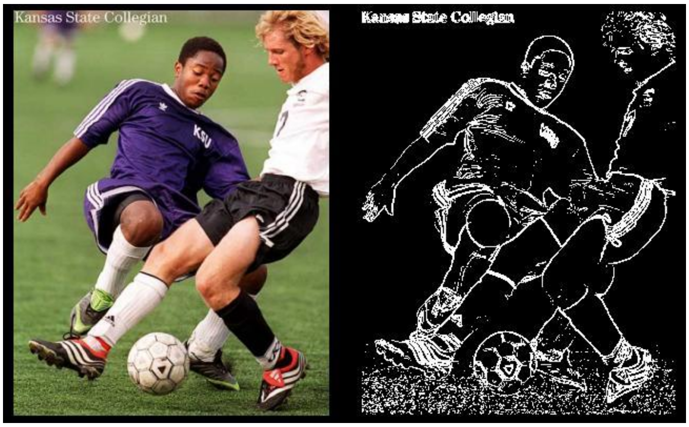
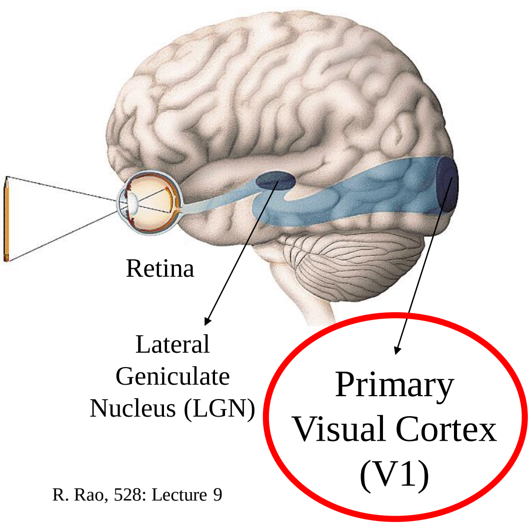
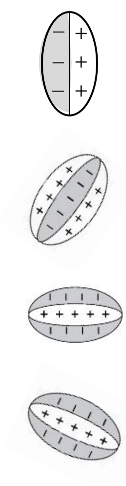


Image from http://www.alexandria.nu/ai/blog/entry.asp?E=51

Edge detectors in the brain

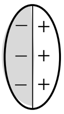




Examples of receptive fields in primary visual cortex (V1)

The Brain can do Calculus!

V1 neurons are basically computing derivatives!



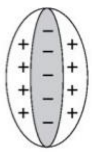
One row of W

$$[0 \quad -1 \quad 1 \quad 0 \quad 0]$$

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$\text{Discrete approximation} \approx f(x+1) - f(x)$$

$$W\mathbf{u} = \mathbf{u}(x+1) - \mathbf{u}(x)$$

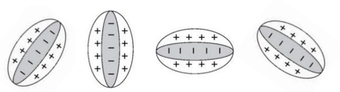


$$[0 \quad 1 \quad -2 \quad 1 \quad 0]$$

$$\frac{d^2 f}{dx^2} = \lim_{h \to 0} \frac{f'(x+h) - f'(x)}{h}$$

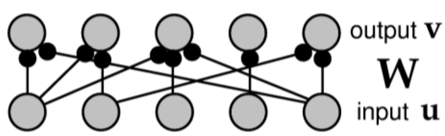
$$\text{Disc. approx.} \approx \big(f(x+1) - f(x)\big) - \big(f(x) - f(x-1)\big)$$

$$= f(x+1) - 2f(x) + f(x-1)$$

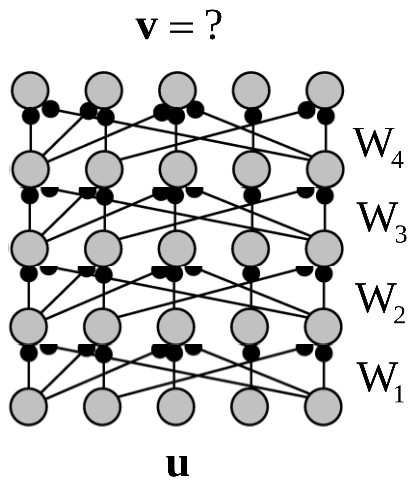


Linear filtering with W**u** is fine but what about W**u**-sing more than 2 layers of neurons?

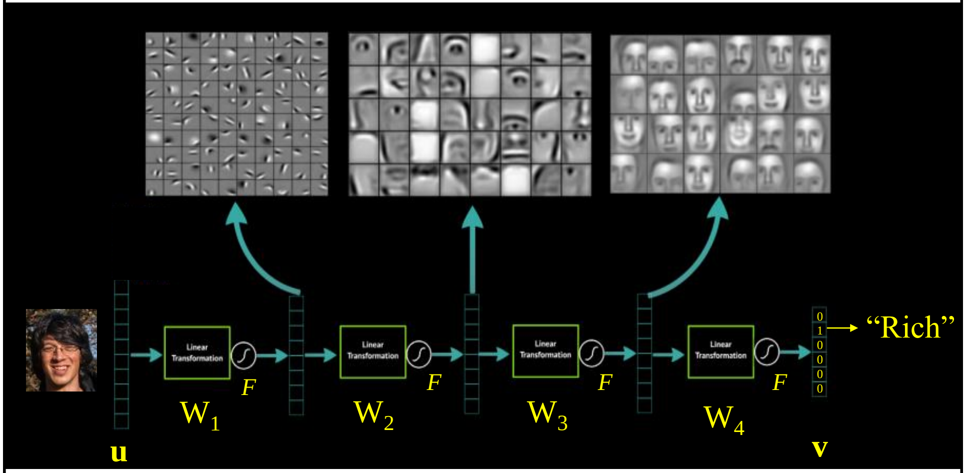
$$\mathbf{v} = W\mathbf{u}$$



Linear Multilayer Feedforward Network



Deep (Nonlinear) Feedforward Networks

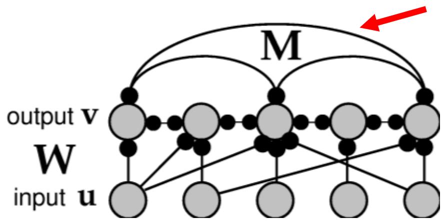


$$\mathbf{v} = F(W_4F(W_3F(W_2F(W_1\mathbf{u}))))$$

How do get the W's? Answer: Stay tuned…

Figure adapted from https://www.datarobot.com/blog/a-primer-on-deep-learning/

Recurrent Neural Networks

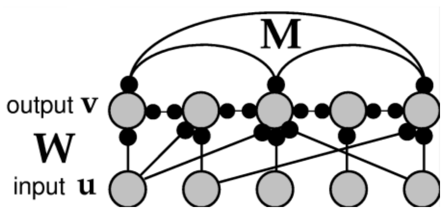


$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + F(\mathrm{W}\mathbf{u} + \mathrm{M}\mathbf{v})$$

Output Decay Input Feedback

Image Source: Dayan & Abbott textbook

What can a Linear Recurrent Network do?



$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + \underbrace{\mathrm{W}\mathbf{u}}_{\mathbf{h}} + \mathrm{M}\mathbf{v}$$

$\mathbf{u} : K \times 1$ vector

$\mathrm{W} : N \times K$ matrix

$\mathbf{v}, \mathbf{h} : N \times 1$ vectors

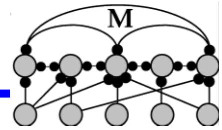
$\mathrm{M} : N \times N$ matrix

Want to find out how **v**(t) behaves for different M

How?

Image Source: Dayan & Abbott textbook

Eigenvectors to the rescue!



$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + \mathbf{h} + M\mathbf{v}$$

- Idea: Use *eigenvectors* of M to *solve differential equation* for **v**
- Suppose $N \times N$ matrix M is symmetric
- M has N *orthogonal* eigenvectors $\mathbf{e}_i$ and N eigenvalues λ_i which satisfy:

$$M\mathbf{e}_i = \lambda_i \mathbf{e}_i$$

Using Eigenvectors to Solve for Network Output $\mathbf{v}(t)$

- We can represent output vector $\mathbf{v}(t)$ using eigenvectors of M:

$$\mathbf{v}(t) = \sum_{i=1}^{N} c_i(t)\mathbf{e}_i$$

- Substituting above in the diff. equation for **v**: $\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + \mathbf{h} + M\mathbf{v}$

  using $M\mathbf{e}_i = \lambda_i \mathbf{e}_i$ and orthonormality of $\mathbf{e}_i$, we can solve for c_i

  (and therefore $\mathbf{v}(t)$!):

$$c_i(t) = \frac{\mathbf{h}\cdot\mathbf{e}_i}{1-\lambda_i}\left(1-\exp\left(\frac{-t(1-\lambda_i)}{\tau}\right)\right) + c_i(0)\exp\left(\frac{-t(1-\lambda_i)}{\tau}\right)$$

(For full derivation, see Lecture Notes on course website)

Eigenvalues determine Network Stability!

$$\mathbf{v}(t) = \sum_{i=1}^{N} c_i(t)\mathbf{e}_i \qquad c_i(t) = \frac{\mathbf{h}\cdot\mathbf{e}_i}{1-\lambda_i}\left(1-\exp\left(\frac{-t(1-\lambda_i)}{\tau}\right)\right) + c_i(0)\exp\left(\frac{-t(1-\lambda_i)}{\tau}\right)$$

If any $\lambda_i > 1$ (e.g., 2), $\mathbf{v}(t)$ explodes $\Rightarrow$ network is unstable!

If all $\lambda_i < 1$, network is stable and $\mathbf{v}(t)$ converges to steady state value:

$$\mathbf{v}_{ss} = \sum_i \frac{\mathbf{h}\cdot\mathbf{e}_i}{1-\lambda_i}\mathbf{e}_i$$

Amplification of Inputs in a Recurrent Network

$$\mathbf{v}_{ss} = \sum_i \frac{\mathbf{h}\cdot\mathbf{e}_i}{1-\lambda_i}\mathbf{e}_i$$

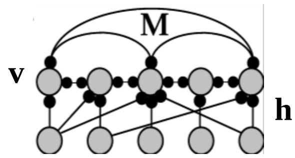
If all $\lambda_i < 1$ and one λ_i (say λ_1) is close to 1 with others much smaller:

$$\mathbf{v}_{ss} \approx \frac{\mathbf{h}\cdot\mathbf{e}_1}{1-\lambda_1}\mathbf{e}_1$$

Amplification of input projection by a factor of $\frac{1}{1-\lambda_1}$

E.g., $\lambda_1 = 0.9,\ \frac{1}{1-\lambda_1} = 10$

Example of a Linear Recurrent Network

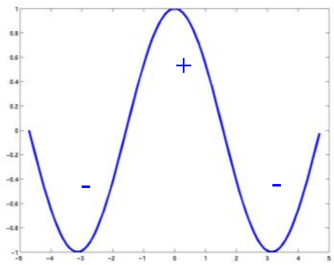


Each output neuron codes for an angle between -180 to +180 degrees

Recurrent connections M = cosine function of relative angle

$$M(\theta,\theta') \propto \cos(\theta - \theta')$$

Excitation nearby,
Inhibition further away



$(\theta - \theta')$

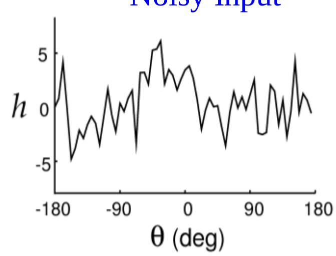
Is M symmetric? $M(\theta, \theta') = M(\theta', \theta)$?

Amplification in the Linear Recurrent Network

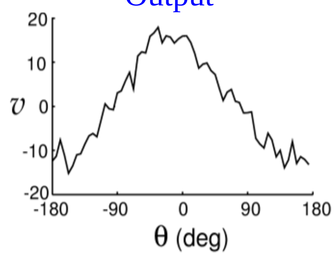
$M(\theta,\theta') \propto \cos(\theta - \theta')$, all eigenvalues = 0 except $\lambda_1 = 0.9$

$$\text{Amplification } \mathbf{v}_{ss} \approx \frac{(\mathbf{h}\cdot\mathbf{e}_1)\mathbf{e}_1}{1-\lambda_1} = 10 \times (\mathbf{h}\cdot\mathbf{e}_1)\mathbf{e}_1$$

Noisy Input



Output



Preferred angle of neuron

(From section 7.4 in Dayan & Abbott textbook)

Memory in Linear Recurrent Networks

$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + \mathbf{h} + M\mathbf{v} \qquad \mathbf{v}(t) = \sum_{i=1}^{N} c_i(t)\mathbf{e}_i$$

Suppose $\lambda_1 = 1$ and all other $\lambda_i < 1$. Then, $\tau \frac{dc_1}{dt} = \mathbf{h} \cdot \mathbf{e}_1$

If input $\mathbf{h}$ is turned on and then off, can show that even after $\mathbf{h} = 0$:

$$\mathbf{v}(t) = \sum_i c_i(t)\mathbf{e}_i$$

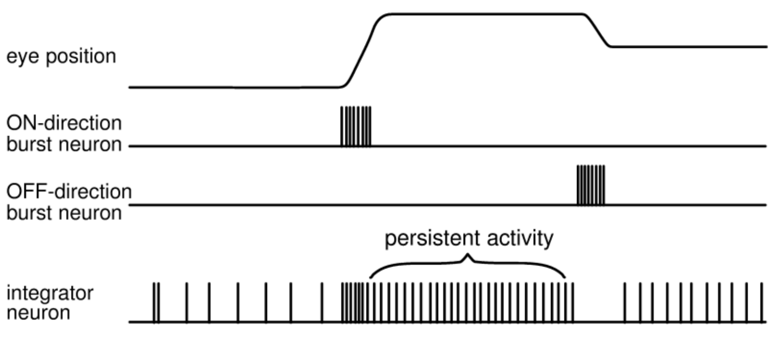
$$\approx c_1\mathbf{e}_1 = \frac{\mathbf{e}_1}{\tau}\int_0^t \mathbf{h}(t') \cdot \mathbf{e}_1 dt'$$

Sustained activity without any input!

Networks keeps a memory of integral of past input

(For full derivation, see Lecture Notes on course website)

The Brain can do Calculus (Part II: Integration)*



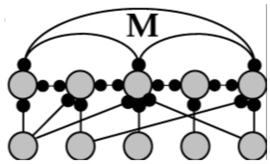
Input: Bursts of spikes from brain stem oculomotor neurons
Output: Memory of eye position in medial vestibular nucleus

*For "Part I: Differentiation," see earlier slide

(Image: Dayan & Abbott based on (Seung et al., 2000))

Nonlinear Recurrent Networks

Input vector **h**



Output vector **v**

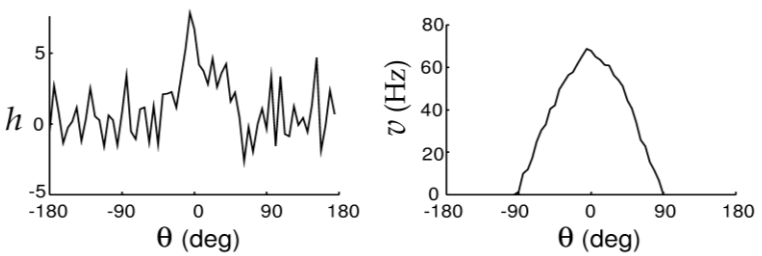
Example: Rectification nonlinearity:
$F(x) = [x]^+ = x$ if $x > 0$ and 0 o.w.

$$\tau \frac{d\mathbf{v}}{dt} = -\mathbf{v} + F(\mathbf{h} + \mathrm{M}\mathbf{v})$$

Output Decay Input Recurrent Feedback

Nonlinear Recurrent Network performs **Amplification**

Input Output

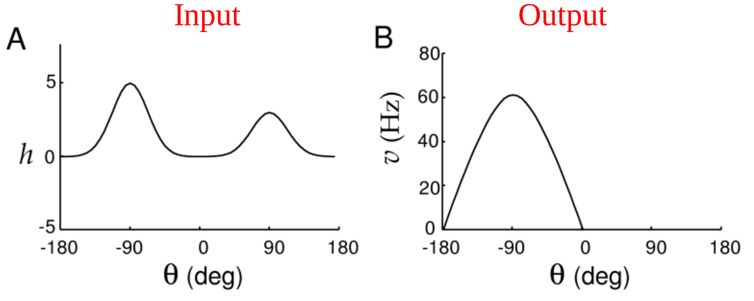


As before, recurrent connections $M(\theta, \theta') \propto \cos(\theta - \theta')$

All eigenvalues $= 0$ but $\lambda_1 = 1.9$ (yet stable due to rectification)

Image Source: Dayan & Abbott textbook

Same Nonlinear Network performs **Selective "Attention"**

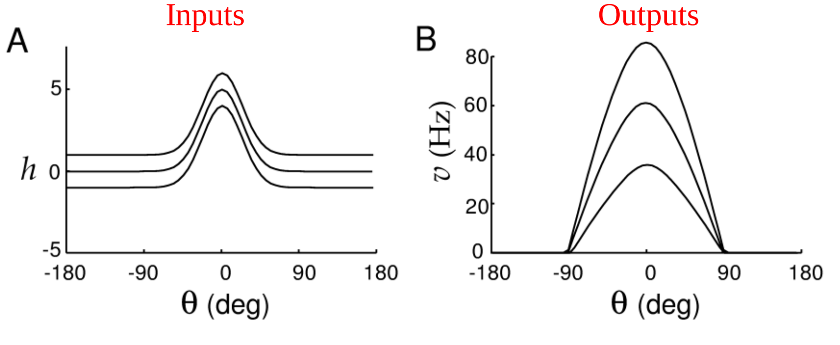


Network performs "Winner-Takes-All" input selection

R. Rao, 528: Lecture 9

Image Source: Dayan & Abbott textbook

Gain Modulation in the Nonlinear Network

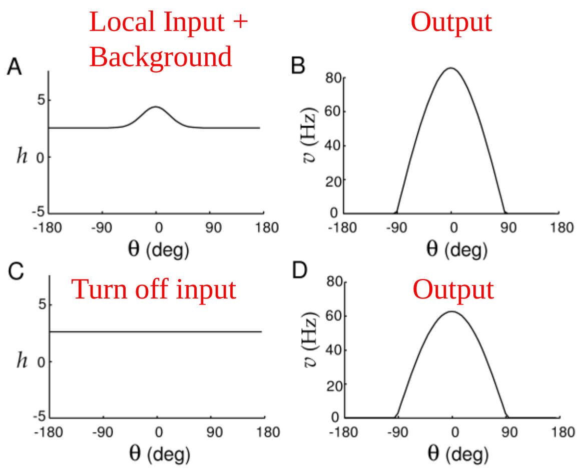


Adding a constant amount to the input *h* **multiplies** the output

R. Rao, 528: Lecture 9

Image Source: Dayan & Abbott textbook

Memory in the Nonlinear Network



Network maintains a *memory of previous activity* when input is turned off.

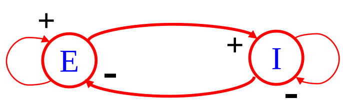
Similar to "short-term memory" or "working memory" in prefrontal cortex

Memory is maintained by recurrent activity

Image Source: Dayan & Abbott textbook

What about Non-Symmetric Recurrent Networks?

- Example: Network of Excitatory (E) and Inhibitory (I) Neurons
  - Connections can't be symmetric: Why?



10 ms → τ_E

$$\tau_E \frac{dv_E}{dt} = -v_E + [M_{EE} v_E + M_{EI} v_I - \gamma_E]^+$$

($M_{EE} = 1.25$, $M_{EI} = -1$, $\gamma_E = -10$)

$$\tau_I \frac{dv_I}{dt} = -v_I + [M_{II} v_I + M_{IE} v_E - \gamma_I]^+$$

($M_{II} = 0$, $M_{IE} = 1$, $\gamma_I = 10$)

Parameter we will vary to study the network (τ_I)

How do we analyze the dynamic behavior of such a network?

Stability Analysis

$$\frac{dv_E}{dt} = \frac{-v_E + [M_{EE}v_E + M_{EI}v_I - \gamma_E]^+}{\tau_E}$$

$$\frac{dv_I}{dt} = \frac{-v_I + [M_{II}v_I + M_{IE}v_E - \gamma_I]^+}{\tau_I}$$

Take derivatives of **right hand side** with respect to both v_E and v_I

Stability Matrix (aka the "Jacobian" Matrix):

$$J = \begin{bmatrix} \frac{(M_{EE}-1)}{\tau_E} & \frac{M_{EI}}{\tau_E} \\ \frac{M_{IE}}{\tau_I} & \frac{(M_{II}-1)}{\tau_I} \end{bmatrix}$$

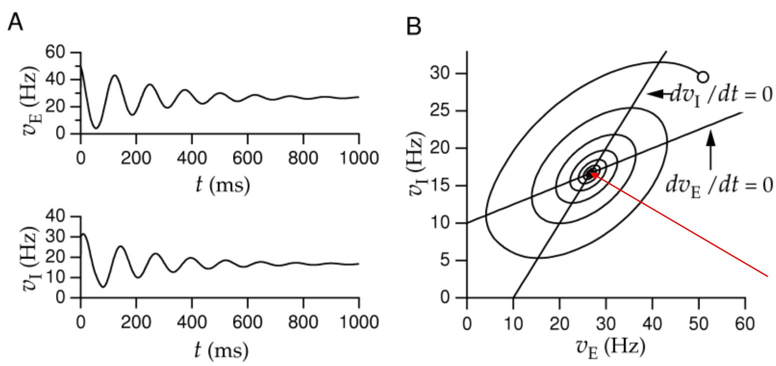
1.25 -1 10 ms 1 0

- Eigenvalues of J can have real and imaginary parts
- These eigenvalues determine dynamics of the nonlinear network near a fixed point

(For all the gory details of this stability analysis, see Lecture Notes on course website)

Damped Oscillations in the Network

Choose τ_I = 30 ms (makes real part of eigenvalues negative)



Stable Fixed Point

Image Source: Dayan & Abbott textbook

Unstable Behavior and Limit Cycle

Choose $\tau_I = 50$ ms (makes real part of eigenvalues positive)

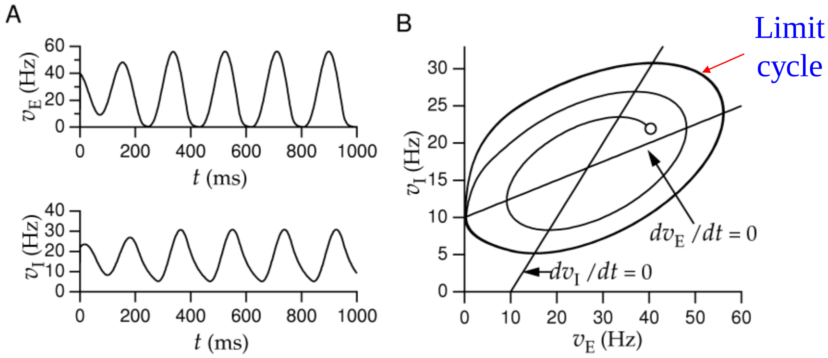
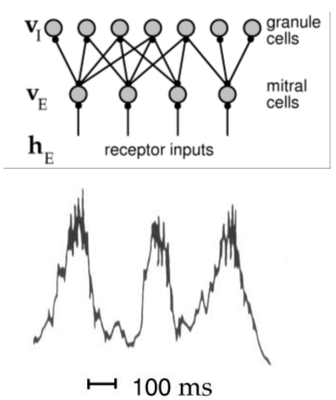


Image Source: Dayan & Abbott textbook

Oscillatory Activity in Real Networks



Activity in rabbit (or wabbit) olfactory bulb during 3 sniffs

(see Chapter 7 in textbook for details)

That’s all folks!
Next Class: Guest lecture by Prof. Eric Shea-Brown

- Things to do:
  - Start reading Chapter 8 in D & A
  - Homework #3 due Sunday Feb 19
  - Finalize a final project topic and partner(s)
    - Email Raj, Adrienne and Rich your topic and partners, or ask to be assigned to a team