Feed-forward neural networks

• We have been considering neural networks that use firing rates, rather than spike trains. (‘rate model’)

• Synaptic input is the firing rate of the input neuron times a synaptic weight \( w \).

\[ I_s = w u \]

• The output firing rate is some non-linear function of the synaptic input.

\[ v = F[I_s] = F[w u] \]
Feed-forward network

- Implements an arbitrary matrix transformation

\[ \vec{v} = W \vec{u} \]
Recurrent networks

• Today we will consider the case where there are also connections between different neurons in the output layer.

• Develop an intuition for how recurrent networks respond to their inputs.

• Examine computations performed by recurrent networks (amplifier, integrator, sequence generation, short term memory).

• Use all the powerful linear algebra tools we have developed!
Learning Objectives for Lecture 18

• Mathematical description of recurrent networks

• Dynamics in simple autapse networks

• Dynamics in fully recurrent networks

• Recurrent networks for storing memories

• Recurrent networks for decision making (winner-take-all)
Learning Objectives for Lecture 18

• Mathematical description of recurrent networks

• Dynamics in simple autapse network

• Dynamics in fully recurrent networks

• Recurrent networks for storing memories

• Recurrent networks for decision making (winner-take-all)
Time dependence

• The steady state firing rate of our output neuron looks like this...

\[ v_\infty = F[I_s] = F[w u] \]

• But neurons don’t respond instantaneously to current inputs

Synaptic delays
Dendritic propagation
Membrane time constant

input neuron

input spike

\[ \hat{I}_{syn}(t) \]

post-synaptic current

output neuron
Time dependence

• We model the firing rate of our model neuron as follows:

$$\tau_n \frac{dv}{dt} = -v + v_\infty$$

• We will look at how networks respond to changes in their inputs

$$v_\infty(t) = F[wu(t)]$$
Time dependence

- We can incorporate time-dependence into our general feed-forward network...

\[
\tau_n \frac{d\vec{v}}{dt} = -\vec{v} + \vec{v}_\infty
\]

\[
\vec{v}_\infty = F\left[ W \vec{u} \right]
\]

- The time dependence is really boring in a feedforward network, but it is extremely important in RNNs.
Recurrent networks

- We will now consider the case where there are connections between different neurons in the output layer.

- Two kinds of input
  - Feed-forward input: $W \vec{u}$
  - Recurrent input: $M \vec{v}$

\[
\tau_n \frac{d\vec{v}}{dt} = -\vec{v} + F\left[W \vec{u} + M \vec{v}\right]
\]
Recurrent networks

• We will now consider the case where there are connections between different neurons in the output layer

\[
W \tilde{u} = \begin{bmatrix}
w_{11} & w_{12} & w_{13} & w_{14} \\
w_{21} & w_{22} & w_{23} & w_{24} \\
w_{31} & w_{32} & w_{33} & w_{34}
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4
\end{bmatrix}
\]

\[
M \tilde{v} = \begin{bmatrix}
m_{11} & m_{12} & m_{13} \\
m_{21} & m_{22} & m_{23} \\
m_{31} & m_{32} & m_{33}
\end{bmatrix}
\begin{bmatrix}
v_1 \\
v_2 \\
v_3
\end{bmatrix}
\]

\[
\tau_n \frac{d\tilde{v}}{dt} = -\tilde{v} + F \left[ W \tilde{u} + M \tilde{v} \right]
\]
Recurrent networks

- We will simplify this equation to focus on the recurrent network.

- Rather than writing the input as a vector of input firing rates, write a vector of effective inputs to each output neuron.

\[ \tau_n \frac{d\tilde{v}}{dt} = -\tilde{v} + F[\tilde{h} + M \tilde{v}] \]

\[ \tilde{h} = W \tilde{u} \]
Recurrent networks

• We will start by analyzing the case with linear neurons

\[ \tau_n \frac{d\vec{v}}{dt} = -\vec{v} + F[\vec{h} + M \vec{v}] \]

• For linear neurons

\[ F(\vec{x}) = \vec{x} \]

Thus...

\[ \tau_n \frac{d\vec{v}}{dt} = -\vec{v} + M \vec{v} + \vec{h} \]

This is a system of coupled equations!
Learning Objectives for Lecture 18

• Mathematical description of recurrent networks

• Dynamics in simple autapse network

• Dynamics in fully recurrent networks

• Recurrent networks for storing memories

• Recurrent networks for decision making (winner-take-all)
Recurrent networks

• Consider the case that $M$ is a diagonal matrix

$$M = \begin{pmatrix}
  \lambda_1 & 0 & 0 \\
  0 & \lambda_2 & 0 \\
  0 & 0 & \lambda_3 \\
\end{pmatrix}$$
Recurrent networks

- Note that if \( M \) is a diagonal matrix

\[
M = \Lambda = \begin{pmatrix}
\lambda_1 & 0 \\
0 & \lambda_2 \\
0 & \lambda_3 \\
\end{pmatrix}
\]

\[
\tau_n \frac{d\vec{v}}{dt} = -\vec{v} + \Lambda \vec{v} + \vec{h}
\]

\[
\tau_n \frac{dv_i}{dt} = -v_i + \lambda_i v_i + h_i
\]

We have \( n \) independent equations – each neuron acts independently of all the others
Recurrent networks

- Rewrite our equation:
  \[ \tau_n \frac{dv_a}{dt} = -v_a + \lambda_a v_a + h_a \]

- There are three cases to consider
  \[ \tau_n \frac{dv_a}{dt} = -(1 - \lambda_a) v_a + h_a \]
  \[
  \begin{aligned}
  > 0 & \quad = 0 \\
  < 0 & 
  \end{aligned}
  \]

- Start with the case that: \( \lambda_a < 1 \)
  \[ \frac{\tau_n}{1 - \lambda_a} \frac{dv_a}{dt} = -v_a + \frac{h_a}{1 - \lambda_a} \]
  \[ \tau_a \frac{dv_a}{dt} = -v_a + v_{a,\infty} \]

A solution we’ve seen before!
\[ v_a(t) = v_{a,\infty} + (v_0 - v_{a,\infty}) e^{-t/\tau_a} \]

Exponential relaxation
Recurrent networks

- Positive (excitatory) feedback acts to amplify the steady state activity of each neuron by an amount that depends on the strength of the feedback!

\[ v_{a,\infty} = \frac{h_a}{1 - \lambda_a} \]

\[ \tau_a = \frac{\tau_n}{1 - \lambda_a} \]

- Positive feedback amplifies the response and slows the time-constant of the response
Recurrent networks

- Negative (inhibitory) feedback acts to suppress the steady state activity of a neuron by an amount that depends on the strength of the feedback.

\[ h_a \]

- Negative feedback suppresses the response and speeds the time-constant of the response

\[ v_{a,\infty} = \frac{h_a}{1 - \lambda_a} \]

\[ \tau_a = \frac{\tau_n}{1 - \lambda_a} \]
Recurrent networks

- If $\lambda < 1$, the activity always relaxes back to zero when the input is removed.
Recurrent networks

• How do we represent the response of a network of neurons.

State-space trajectories

• Cool computations
  – Amplifier
  – Integrator
  – Memory
  – Sequence generator
State-space trajectories
Learning Objectives for Lecture 18

• Mathematical description of recurrent networks
• Dynamics in simple autapse network
• Dynamics in fully recurrent networks
• Recurrent networks for storing memories
• Recurrent networks for decision making (winner-take-all)
Recurrent networks

- Now let’s look at the more general case of recurrent connectivity.

\[
M = \begin{pmatrix}
m_{11} & m_{12} \\
m_{21} & m_{22}
\end{pmatrix}
\]
Recurrent networks

• We saw how the behavior of a recurrent network is extremely simple to describe if $M$ is diagonal.

• So let’s make $M$ diagonal! Rewrite $M$ as follows

$$M = \Phi \Lambda \Phi^T$$

where $\Lambda$ is a diagonal matrix.
Recurrent networks

• How do we write $M$ as $\Phi \Lambda \Phi^T$?

• Solve the eigenvalue equation $M \Phi = \Phi \Lambda$

  o The diagonal elements of $\Lambda$ are the eigenvalues of $M$

  $$\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \\ 0 & \lambda_3 & \ddots \end{pmatrix}$$

  o The columns of $\Phi$ are the eigenvectors of $M$

$$\Phi = \begin{bmatrix} \hat{f}_1 | \hat{f}_2 | \hat{f}_3 | \cdots | \hat{f}_n \end{bmatrix}$$

• Remember that...

$$M \hat{f}_\alpha = \lambda_\alpha \hat{f}_\alpha$$
Recurrent networks

\[ M \Phi = \Phi \Lambda \quad M \hat{f}_\mu = \lambda_\mu \hat{f}_\mu \]

- If \( M \) is a symmetric matrix, then …
  - the eigenvalues are real
  - \( \Phi \) is a rotation matrix. The eigenvectors give us an orthogonal basis set:

\[ \hat{f}_i \cdot \hat{f}_j = \delta_{ij} \]

\[ \Phi^T \Phi = I \]
Recurrent networks

- Now we are going to write our vector of output firing rates \( \vec{v} \) in this new basis.

  Project \( \vec{v} \) onto each of the new basis vectors.

\[
c_\alpha = \vec{v} \cdot \hat{f}_\alpha
\]

- Express \( \vec{v} \) as a linear combination of basis vectors

\[
\vec{v} = c_1 \hat{f}_1 + c_2 \hat{f}_2 + c_3 \hat{f}_3 + \ldots
\]
Recurrent networks

• Of course $\vec{v}$ is a function of time, so we have to write...

$$\vec{v}(t) = c_1(t) \hat{f}_1 + c_2(t) \hat{f}_2 + c_3(t) \hat{f}_3 + \ldots$$

or

$$\vec{v}(t) = \sum_{i=1}^{n} c_i(t) \hat{f}_i$$

where

$$c_\alpha(t) = \vec{v}(t) \cdot \hat{f}_\alpha$$

• In matrix notation, we write this change-of-basis as

$$\vec{v} = \Phi \vec{c} \quad \vec{c} = \Phi^T \vec{v}$$
Recurrent networks

• Let’s rewrite our network equation in this new basis set...

\[ \tau_n \frac{d\bar{v}}{dt} = -\bar{v} + M\bar{v} + \bar{h} \quad \bar{v} = \Phi\bar{c} \]

\[ \tau_n \Phi \frac{d\bar{c}}{dt} = -\Phi\bar{c} + M\Phi\bar{c} + \bar{h} \]

• But we have chosen a basis set \( \Phi \) such that

\[ M\Phi = \Phi\Lambda \]

• Thus...

\[ \tau_n \Phi \frac{d\bar{c}}{dt} = -\Phi\bar{c} + \Phi\Lambda\bar{c} + \bar{h} \]
Recurrent networks

\[ \tau_n \Phi \frac{d\tilde{c}}{dt} = -\Phi \tilde{c} + \Phi \Lambda \tilde{c} + \tilde{h} \]

• Multiply both sides from the left by \( \Phi^T \)

\[ \tau_n \Phi^T \Phi \frac{d\tilde{c}}{dt} = -\Phi^T \Phi \tilde{c} + \Phi^T \Phi \Lambda \tilde{c} + \Phi^T \tilde{h} \]

\[ \tau_n \frac{d\tilde{c}}{dt} = -\tilde{c} + \Lambda \tilde{c} + \tilde{h}_f \quad \tilde{h}_f = \Phi^T \tilde{h} \]

• But this is just our original network equation with a diagonal weight matrix!
Recurrent networks

• We can rewrite the equation for our network as \( n \) independent equations for \( n \) independent ‘modes’ of the network.

• We can think of this transformation as making a new network with only autapses.

\[
\rho h \cdot \hat{f}_1 \quad \rho h \cdot \hat{f}_2
\]

• The activities \( c_\alpha (t) \) of our network modes represent activity of linear combinations of neurons in our original network.
Recurrent networks

Let’s find the steady-state solution of our system of equations...

\[
\tau_n \frac{dc}{dt} = -\tilde{c} + \Lambda \tilde{c} + \Phi^T \hat{h}
\]

\[
\tau_n \frac{dv}{dt} = -\tilde{v} + M \tilde{v} + \hat{h}
\]

\[
\tau_n \frac{d\bar{c}}{dt} = -I \tilde{c} + \Lambda \tilde{c} + \Phi^T \hat{h}
\]

\[
\tilde{v} = \Phi \tilde{c}
\]

\[
\tau_n \frac{d\bar{c}}{dt} = -(I - \Lambda) \tilde{c} + \Phi^T \hat{h}
\]

\[
0 = -(I - \Lambda) \tilde{c}_\infty + \Phi^T \hat{h}
\]

\[
(I - \Lambda) \tilde{c}_\infty = \Phi^T \hat{h}
\]

\[
\tilde{c}_\infty = (I - \Lambda)^{-1} \Phi^T \hat{h}
\]

\[
\Phi \tilde{c}_\infty = \Phi (I - \Lambda)^{-1} \Phi^T \hat{h}
\]

\[
\tilde{v}_\infty = \Phi \tilde{c}_\infty
\]

\[
\tilde{v}_\infty = \Phi (I - \Lambda)^{-1} \Phi^T \hat{h}
\]
Recurrence networks

- The steady-state solution (with input vector $\vec{h}$) is:

$$\vec{v}_\infty = \Phi(I - \Lambda)^{-1}\Phi^T \vec{h}$$

- So what happens if our input is parallel to one of the eigenvectors?

$$\vec{v}_\infty = G \vec{h}$$

- Then, in steady state, the output will be parallel to the input!
Recurrent networks

- If our input vector is parallel to one of the eigenvectors, then our steady-state output will be parallel to the input.

- In this case, our input activates only one mode of the network, and no other mode.

- The response of the network to inputs along each of the eigenvectors (modes) is amplified or suppressed by a gain factor

\[ g_\mu = \frac{1}{1 - \lambda_\mu} \]

- The time constant of the response is increased or decreased by the same factor

\[ \tau_\mu = \frac{\tau_n}{1 - \lambda_\mu} \]
Recurrent networks

- Now let’s look at a case where two output neurons are connected to each other by mutual excitation.

What is the weight matrix?

\[ M = \begin{pmatrix} 0 & 0.8 \\ 0.8 & 0 \end{pmatrix} \]

\[ M \Phi = \Phi \Lambda \]

\[ \Phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \]

\[ \Lambda = \begin{pmatrix} 0.8 & 0 \\ 0 & -0.8 \end{pmatrix} \]

\[ \hat{f}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \]

\[ \hat{f}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \]

\[ g_1 = 5 \]

\[ g_2 = 1.8^{-1} \]
Recurrent networks

- If the input is parallel to the eigenvectors, then only one mode is excited.

\[
\hat{f}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
g_1 = 5 \\
\hat{f}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \\
g_2 = 1.8^{-1}
\]
Recurrent networks

- If the input is not parallel to an eigenvector, we break the input into a component along each mode

\[
\vec{h} = (\vec{h} \cdot \hat{f}_1) \hat{f}_1 + (\vec{h} \cdot \hat{f}_2) \hat{f}_2
\]

\[
\hat{f}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad \hat{f}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}
\]

\[
\lambda_1 = 0.8, \quad \lambda_2 = -0.8
\]
Recurrent networks

- Two output neurons are connected to each other by mutual excitation.

\[
\begin{align*}
\hat{f}_1 &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
g_1 &= 5 \\
\hat{f}_2 &= \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \\
g_2 &= 1.8^{-1}
\end{align*}
\]
Recurrent networks

- Now let’s look at a case where two output neurons are connected to each other by mutual inhibition.

What is the weight matrix?

\[ M = \begin{pmatrix} 0 & -0.8 \\ -0.8 & 0 \end{pmatrix} \]

\[ M \Phi = \Phi \Lambda \]

\[ \Phi = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \]

\[ \Lambda = \begin{pmatrix} -0.8 & 0 \\ 0 & 0.8 \end{pmatrix} \]

\[ \hat{f}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \]

\[ \hat{f}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \]

\[ g_1 = 1.8^{-1} \]

\[ g_2 = 5 \]
Recurrent networks

- Two output neurons are connected to each other by mutual inhibition.

\[ \hat{f}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \]

\[ \hat{f}_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix} \]

\[ g_2 = 5 \]

\[ g_i = 1.8^{-1} \]
Learning Objectives for Lecture 18

• Mathematical description of recurrent networks

• Dynamics in simple autapse network

• Dynamics in fully recurrent networks

• Recurrent networks for storing memories

• Recurrent networks for decision making (winner-take-all)
Recurrent networks

• We have described the case where \( \lambda < 1 \).

What happens when \( \lambda = 1 \)?

\[
\tau_n \frac{dc_\alpha}{dt} = -(1 - \lambda\alpha)c_\alpha + \hat{f}_\alpha \cdot \vec{h}(t)
\]

\[
\tau_n \frac{dc_1}{dt} = \hat{f}_1 \cdot \vec{h}(t) = h_{f_1}(t)
\]

\[
c_1(t) = c_1(0) + \frac{1}{\tau_n} \int_0^t h_{f_1}(\tau) d\tau
\]

Integrator!
Recurrent networks

• What happens when $\lambda > 1$?

\[ \tau_n \frac{dc_1}{dt} = -(1 - \lambda_1)c_1 + \hat{f}_1 \cdot \bar{h}(t) \]

\[ \tau_n \frac{dc_1}{dt} = (\lambda_1 - 1)c_1 + \hat{f}_1 \cdot \bar{h}(t) \]

$> 0$

Exponential growth!
Recurrent networks

• The behavior of the network depends critically on $\lambda$

$\lambda < 1$
Exponential relaxation

$\lambda = 1$
Integration

$\lambda > 1$
Exponential growth

With zero input...
relaxation back to zero

With zero input...
persistent activity!

MEMORY!
Recurrent networks

- Networks with $\lambda \geq 1$ have memory!

$$\tau_n \frac{dc_1}{dt} = (\lambda_1 - 1)c_1 + h_{f1}(t)$$

$$\tau_n \frac{dc_1}{dt} = c_1 \quad c_1(t) = 0$$

- With zero input, zero is an ‘unstable fixed point’ of the network
Recurrent networks

• Add a saturating activation function $F(x)$

$$v = F(I)$$
Recurrent networks

- Saturating activation function plus eigenvalues greater than 1 lead to stable states other than zero!
Recurrent networks

- Two-neuron network that has two attractors

\[ v_1 = F(I) \]

\[ v_2 = F(I) \]

\[ h_1 \quad 2 \]

\[ h_2 \quad -2 \]
Learning Objectives for Lecture 18

• Mathematical description of recurrent networks

• Dynamics in simple autapse network

• Dynamics in fully recurrent networks

• Recurrent networks for storing memories

• Recurrent networks for decision making (winner-take-all)
Winner-take-all network

- Implements decision making

Network will remain in attractor 1 if \( h_1 > h_2 \)

Network will remain in attractor 2 if \( h_2 > h_1 \)
Winner-take-all network

- Implements decision making
Learning Objectives for Lecture 18

• Mathematical description of recurrent networks

• Dynamics in simple autapse network

• Dynamics in fully recurrent networks

• Recurrent networks for storing memories

• Recurrent networks for decision making (winner-take-all)
Recurrent networks

• Networks with many attractors…
Hopfield networks

• Networks with many attractors…

\[2^n \text{ possible states !}\]
9.40 Introduction to Neural Computation
Spring 2018

For information about citing these materials or our Terms of Use, visit: https://ocw.mit.edu/terms.