Introduction to Neural Networks
U. Minn. Psy 5038

Lateral inhibition

Introduction

Last time

- **Developed a "structure-less, continuous signal, and discrete time" generic neuron model and and from there built a network.**

- **Basic linear algebra review. Motivated linear algebra concepts from neural networks.**

Today

We are going to look at an explanation of a human perceptual phenomenon called Mach bands, that involves a linear approximation based on a real neural network. The model is a good fit to the data. This is an example of neural filtering found in early visual coding. We will study two types of network that may account for Mach bands: 1) feedforward; 2) feedback. The feedback system will provide our first example of a dynamical system. The system we will look at was developed as a model of the neural processing in the horseshoe crab (*limulus*) compound eye. Despite the (apparent) enormous difference between your visual system and that of the horseshoe crab, our visual system shares a fundamental image processing function with that of this lowly crustacean (and virtually all other animals that have image-based image components).

- **An application of a simple linear model for visual spatial filtering**

- **Add some dynamics for visual spatial filtering**

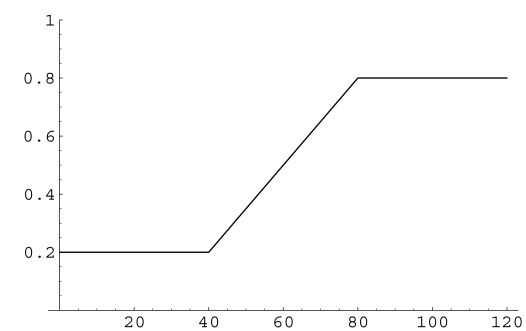
- **Winner-take-all network: Add a threshold non-linearity**

Mach bands & perception

Ernst Mach was an Austrian physicist and philospher. In addition to being well-known today for a unit of speed, he is also known for several visual illusions. One illusion is called "Mach bands". Let's make some.

In[170]:=
```
Clear[y];
low = 0.2; hi = 0.8;
y[x_] := low /; x<40
y[x_] :=
    ((hi-low)/40) x + (low-(hi-low)) /; x>=40 && x<80
y[x_] := hi /; x>=80
```

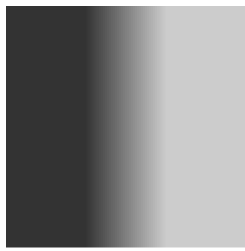
In[175]:=
```
Plot[y[x],{x,0,120},PlotRange->{0,1}];
```



In[176]:=
```
size = 120;
e := Table[y[i],{i,1,size}];
```

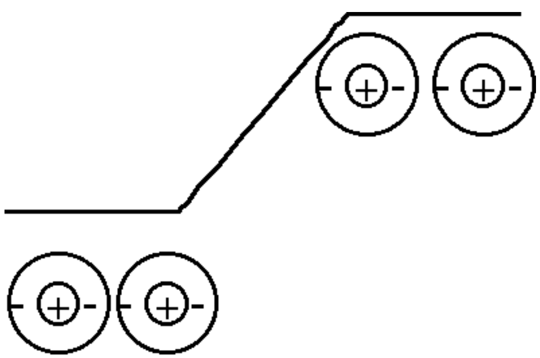
Let's make a 2D gray-level picture displayed with **ListDensityPlot** to experience the Mach bands for ourselves. **PlotRange** allows us to scale the brightness.

In[178]:=
```
e1 = e;
picture = Table[e1,{i,1,60}];
ListDensityPlot[picture,Frame->False,Mesh->False,
                PlotRange->{0,1}];
```



What Mach noticed was that the left knee of the ramp looked too dark, and the right knee looked too bright. Objective light intensity did not predict apparent brightness.

■ Mach's explanation



Neural basis?

Limulus (horseshoe crab)--Hartline, who won the 1967 Nobel prize for this work that began in the 30's.

Frog - Barlow

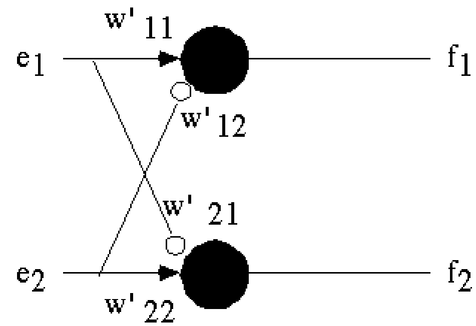
Cat --Kuffler

Feedforward model

Two types of models: feedforward and feedback (in our context, "recurrent lateral inhibition")

y = w' .**e**

where **e** is a vector representing the input intensities, w' is a suitably chosen set of weights (i.e. excitatory center and inhibitory surround as shown in the above figure), and **y** is the output.



■ *Mathematica* implementation

Because the stimulus is effectively one-dimensional, we'll simulate the response in one dimension.

Let the receptive field for one output unit be represented by 5 weights, with a center value of 6, and surround values of -1:

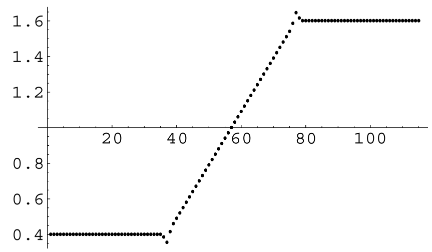
In[181]:=
```
wp = Table[0,{i,1,Length[e]}];
wp[[1]]=-1; wp[[2]]=-1; wp[[3]]=6; wp[[4]]=-1; wp[[5]]=-1;
```

Now assume that all units have the same weights, and calculate the response at each point by shifting the weight filter **wp** right one by one, and taking the dot product with the input pattern **e**, each time:

In[183]:=
```
response = Table[RotateRight[wp,i].e,{i,1,115}];
```

This way we can mimic the response we want:

In[185]:=
```
ListPlot[response];
```



There is neurophysiological evidence for an implementation of lateral inhibition via feedback or *recurrent lateral inhibition*.

Feedback model: Recurrent lateral inhibition

■ Dynamical systems: difference equation for one neuron

State of neuron output **f** at discrete time k.

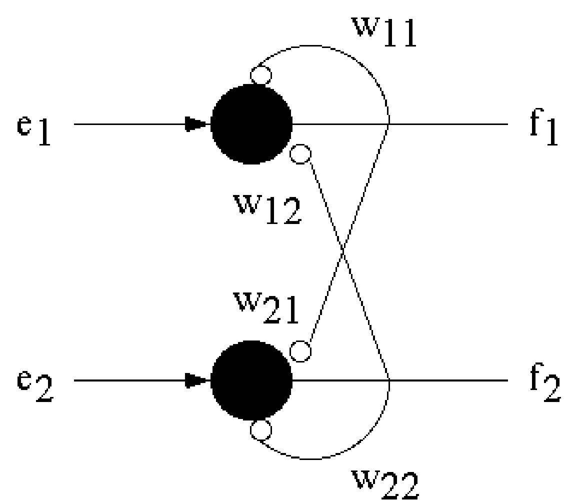
One neuron

$$f[k+1] = e[k] + w\, f[k] \quad (1)$$

Suppose the initial state f[0] is known and e[k] is zero, can you find an expression for f[k]?

■ Dynamical systems: Coupled difference equations for interconnected neurons

Now let's study a two neuron system. The formalism will extend naturally to higher dimensions. To keep this simpler, we won't specify weights for the inputs e, but we will specify weights for the newly added feedback connections:



Let **e** be the input activity vector to the neurons, **f** is the n-dimensional state vector representing output activity and **W** is a fixed nxn weight matrix. Then for a two neuron network we have:

$$\begin{aligned} f_1[k+1] &= e_1[k] + w_{12}\, f_2[k] + w_{11}\, f_1[k] \\ f_2[k+1] &= e_2[k] + w_{21}\, f_1[k] + w_{22}\, f_2[k] \end{aligned} \quad (2)$$

or in terms of vectors and matrices

$$\begin{pmatrix} f_1[k+1] \\ f_2[k+1] \end{pmatrix} = \begin{pmatrix} e_1[k] \\ e_2[k] \end{pmatrix} + \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \begin{pmatrix} f_1[k] \\ f_2[k] \end{pmatrix} \quad (3)$$

or in summation notation:

$$f_i[k+1] = e_i[k] + \sum_j w_{ij} \cdot f_j[k] \quad (4)$$

or in concise vector-matrix (and *Mathematica*) notation:

$$\mathbf{f}[k+1] = \mathbf{e}[k] + \mathbf{W} \,.\, \mathbf{f}[k] \quad (5)$$

where $\mathbf{W} = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}$

This equation is an example of a simple dynamical system. As you might imagine, the state of dynamical system typically changes with time (i.e. iteration k). Are there solutions for which the state does not change with time? It there are these solutions are called steady state solutions.

In contrast to the way we set up the weights for the feedforward matrix (which included the forward excitatory weights), we are going to assume later that all of these weights are inhibitory (because we are modeling lateral inhibition). The postive contributions, if any, will come from the input **e**.

■ Steady state solution for a discrete system

A steady-state solution simply means that the state vector **f** doesn't change with time:

$$\mathbf{f}[k+1] = \mathbf{f}[k] \tag{6}$$

or in vector and *Mathematica* notation:

$$\mathbf{f} = \mathbf{e} + \mathbf{W}.\mathbf{f}$$

where we drop the index k. Note that by expressing **f** in terms of **e**, this is equivalent to another linear matrix equation,the feedforward solution:

$$\mathbf{f} = \mathbf{W}'.\mathbf{e},$$

where

$$\mathbf{W}' = (\mathbf{I} - \mathbf{W})^{-1}$$

The **-1** exponent means the inverse of the matrix in brackets. **I** is the identity matrix.

We will review more later on how to manipulate matrices, find the inverse of a matrix, etc..

■ Dynamical system -- coupled differential equations ("limulus" equations)

What if time is not modeled in discrete clocked chunks? The theory for coupled discrete equations

$$\mathbf{f}[k+1] = \mathbf{e}[k] + \mathbf{W}.\mathbf{f}[k] \tag{7}$$

nicely parallels the theory for continuous differential equations where time varies continuously:

$$\frac{d\mathbf{f}}{dt} = \mathbf{e}[t] + \mathbf{W}''.\mathbf{f}[t] \tag{8}$$

(***W″*** is not necessarily the same matrix as ***W*** .) If you want to learn more about dynamical systems, see Luenberger (1979).

Continous time seems a more reasonable assumption for a neural network of visual processing, so we model a dynamical system for lateral inhbition with feedback.

Let **e**(t) be the input activity to the neurons, **f**(t) is the n-dimensional state vector representing output activity now as a function of time. **W** is a fixed nxn weight matrix. The equation in the previous section is the steady state solution to the following differential equation:

$$\frac{d\mathbf{f}}{dt} = \mathbf{e}[t] + \mathbf{W}.\mathbf{f}[t] - \mathbf{f}[t] \tag{9}$$

(You can see this by noting that as before, "steady state" just means that the values of **f(t)** are not changing with time, i.e. **df/dt = 0**). We are going to develop a solution to this set of equations using a discrete-time approximation.

The state vector **f** at time t+Δt (ε = Δt) can be approximated as:

$$\mathbf{f}(t+\Delta t) \cong \mathbf{f}(t) + \varepsilon[\mathbf{e}(t) + \mathbf{W}.\mathbf{f}(t) - \mathbf{f}(t)]$$

We will fix or "clamp" the input **e**, start with arbitrary position of the state vector **f**, and model how the state vector evolves through time. We'll ask whether it seeks a stable state for which **f**(t) is no longer changing with time, **f**(t + Δt) ≅ **f**(t),

i.e. when **d**f/dt = 0. In the limit as Δt (or ε) approaches zero, the solution is given by the steady state solution of the previous section. But neural systems take time to process their information and for the discrete time approximation, the system may not necessarily evolve to the steady state solution.

Simulation of recurrent lateral inhibition

First we will initialize parameters for the number of neurons (**size**), the space constant of the lateral inhibitory field (**spaceconstant**), the maximum strength of the inhibitory weights (**maxstrength**), the number of iterations (**iterations**), and **ϵ**:

■ The input stimulus

In[1]:=
```
size = 30;
spaceconstant =5;
maxstrength = 0.05;
iterations = 10;
ϵ = .3;
```

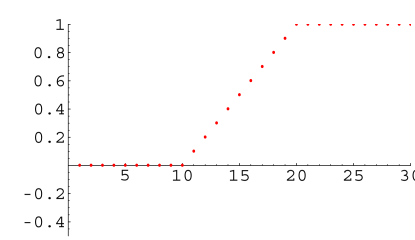
In[6]:=
```
e   = Join[Table[0,{i,N[size/3]}],Table[i/N[size/3],
        {i,N[size/3]}], Table[1,{i,N[size/3]}]];
g0  = ListPlot[e, PlotRange -> {{0,30},{-0.5,1.0}},PlotStyle-
>{RGBColor[1,0,0]},
        DisplayFunction -> Identity];
picture = Table[e,{i,1,30}];
```

We've stored the graphic **g0** of the input for later use. The option

DisplayFunction -> Identity prevents the display. We can turn it on later with:

DisplayFunction -> $DisplayFunction.

In[9]:=
```
Show[g0, DisplayFunction -> $DisplayFunction];
```



■ Initializing the state vector and specifying the weights

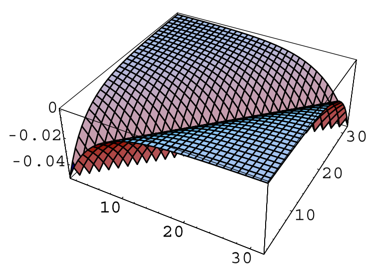
Now we'll initialize the starting values of the output **f** to be random real numbers between 0 and 1, drawn from a uniform distribution.

In[10]:=
```
f = Table[Random[],{size}];
```

Now let's set up synaptic weights which are negative, but become weaker the further they get from the neuron. We assume that the weights drop off exponentially away from each neuron:

In[48]:=
```
W =
Table[N[-maxstrength Exp[-Abs[i-j]/spaceconstant],1],
        {i,size},{j,size}];
```

In[49]:=
```
ListPlot3D[W];
```



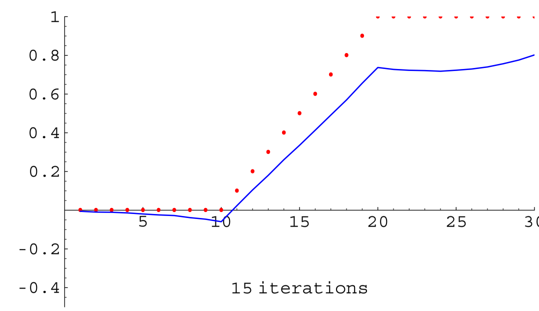
■ Simulating the response

We are going to use the *Mathematica* function **Nest[]** to iterate through the limulus equations. `Nest[`*f*`,` *expr*`,` *n*`]` gives an expression with *f* applied *n* times to *expr*. For example, if we have defined a function **T[]**, **Nest[T,x,4]** produces as output **T[T[T[T[x]]]]**.

Let's express our discrete approximation for the limulus dynamical system in terms of a function, **T**, which will get applied repeatedly to itself with Nest:

In[15]:=
```
T[f_] := f + ϵ (e + W.f - f);
```


In[16]:=
```
iterations = 15;
g1 = ListPlot[Nest[T, f, iterations],PlotJoined->True,
        PlotRange -> {{0,30},{0,1.0}},PlotStyle->{RGBColor[0,0,1]},
        DisplayFunction -> Identity];
Show[g0,g1, Graphics[Text[iterations "iterations",
        {size/2,-0.4}]],
        DisplayFunction -> $DisplayFunction];
```



Exercises: Explore the parameter space

The effect of ϵ, strength of inhibition, and number of iterations

■ Define a function with inputs: ϵ, maxstrength and iterations, and outputs: a plot of response

We can use the **Module[ ]** function to define a routine with local variables and a set of other functions to define limulus[ϵ_,maxstrength_,iterations_]:

In[19]:=
```
limulus[ϵ_, maxstrength_, iterations_] := Module[{f, W},
  W = Table[
    N[-maxstrength Exp[-Abs[i - j] / spaceconstant], 1], {i, size}, {j, size}];
  f = Table[Random[], {size}];
  T[f_] := f + ϵ (e + W.f - f);
  g1 = ListPlot[Nest[T, f, iterations],
    PlotJoined → True, PlotRange → {{0, 30}, {0, 1.0}},
    PlotStyle → {RGBColor[0, 0, 1]}, DisplayFunction → Identity];
Show[g0, g1, Graphics[Text[iterations "iterations", {size / 2, -0.4}]],
   DisplayFunction → $DisplayFunction];
 ];

(*Note that this function isn't "clean"--
  although f and W are local variables,it relies on earlier
  global definitions of size,g0,spaceconstant, and e.*)
```

In[20]:=
```
limulus[.3, .05, 15];
```


For maxstrength = 0.05, ϵ = .3, run limulus[.3,.05, iteration] for iteration values = 1, 3, 9, 27

What does the steady state response look like if the inhibition is small (i.e. small maxstrength)?

What does the steady state response look like if the inhibition is large?

What if the iteration step-size, ϵ, is large (e.g. 2)

Neural networks as dynamical systems

We've explored a simple linear neural network that is a good model of limulus processing, and seems to provide a possible explanation for human perception of Mach bands. Real neural networks typically have non-linearities. There is no general theory of non-linear systems of difference or differential equations. But the exploration of this linear set does lead us to ask questions which are quite general about dynamical systems:

What does the trajectory in state-space look like?

Does it go to a stable point?

How many stable points or "attractors" are there?

There are non-linear systems which show more interesting behavior in which one sees:

Stable orbits

Chaotic trajectories in state-space

"Strange" attractors

We will return to some of these questions later when we study Hopfield networks.

Recurrent lateral inhibition & Winner-take-all (WTA)

Sometimes one would like to have a network that takes in a range of inputs, but as output would like the neuron with biggest value to remain high, while all others are suppressed. In other words, we want the network to make a decision. The limulus equation can be set up to act as such a "winner-take-all" network. We will remove self-inhibition by setting all the diagonal elements of **W** to zero. We will also add a non-linear thresholding function ("rectification") to set negative values to zero, and we will increase the spatial extent of the inhibition.

- **Make a rectifying threshold function**

In[81]:=
```
thresh[x_] := N[If[x < 0.0, 0.0, x]];
SetAttributes[thresh, Listable];
```

- **Make a "tepee" stimulus and intialize the neural starting values**

In[155]:=
```
size = 32;
e   = Join[Table[0,{i,N[size/4]}],
        Table[i/N[size/4],{i,N[size/4]}],
        Table[(N[size/4]-i)/N[size/4],{i,N[size/4]}],
        Table[0,{i,N[size/4]}]];
g0  = ListPlot[e, PlotRange ->
{{0,size},{-1,2.0}},PlotStyle→{RGBColor[1,0,0]},
        DisplayFunction -> Identity];
```

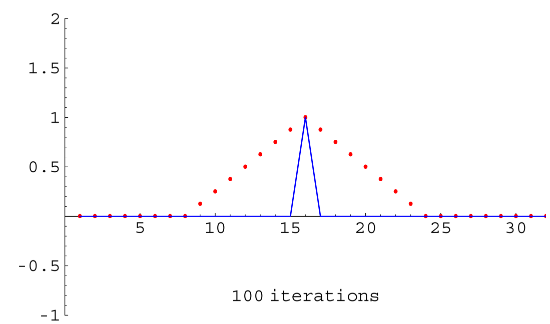
- **Define winnertakeall[ ] as for limulus[ ], but with no self-inhibition:**

In[158]:=
```
winnertakeall[ϵ_, maxstrength_,
   iterations_, spaceconstant_] := Module[{f, W},
   W = Table[N[-maxstrength Exp[-Abs[i - j] / spaceconstant], 1],
     {i, size}, {j, size}];
   For[i = 1, i <= size, i++, W[[i, i]] = 0.0];
   f = Table[Random[], {size}];
   T[f_] := thresh[f + ϵ (e + W.f - f)];
   g1 = ListPlot[Nest[T, f, iterations],
     PlotJoined → True, PlotRange → {{0, size}, {-1, 2.0}},
     PlotStyle → {RGBColor[0, 0, 1]}, DisplayFunction → Identity];
Show[g0, g1, Graphics[Text[iterations "iterations", {size / 2, -0.8}]],
    DisplayFunction → $DisplayFunction];
  ];

(*Note that this function isn't "clean"-- although f and W are local
  variables,it relies on earlier global definitions of size,g0,and e. *)
```


Use ListPlot3D[W] to see the modified structure of the weight matrix

Run simulation: Find a set of parameters that will select the maximum response and suppress the rest



If we think of the number of iterations to steady-state as "reaction time", does this neural network for making decisions? How sensitive is its function to the choice of parameters?

If you are having a hard time finding a good set of parameters, select the cell below, then go to **Cell->Cell Properties->Cell Open**, and then run it.

Next time

■ **Review matrices. Representations of neural network weights**

References

Anderson, J. A. (1995). An Introduction to Neural Networks . Cambridge, MA: MIT Press. (Chapter 4.)

Hartline, H. K., & Knight, B. W., Jr. (1974). The processing of visual information in a simple retina. Ann N Y Acad Sci, 231(1), 12-8.

Knill, D. C., & Kersten, D. (1991). Apparent surface curvature affects lightness perception. Nature, 351, 228-230.http://gandalf.psych.umn.edu/~kersten/kersten-lab/demos/lightness.html

Luenberger, D.G. (1979). Introduction to dynamic systems : theory, models, and applications. (pp. xiv, 446). New York: Wiley.

Ratliff, F., Knight, B. W., Jr., Dodge, F. A., Jr., & Hartline, H. K. (1974). Fourier analysis of dynamics of excitation and inhibition in the eye of Limulus: amplitude, phase and distance. Vision Res, 14(11), 1155-68.

Appendix

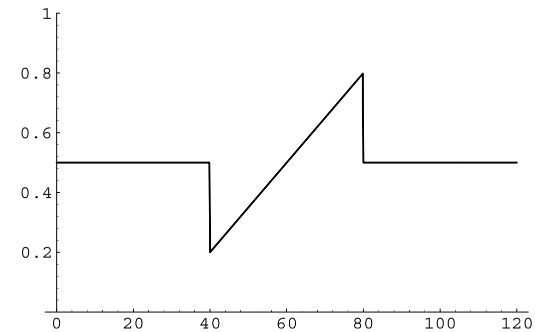
Exercise: Make a gray-level image of the horizontal luminance pattern shown below.

Does the left uniform gray appear to be the same lightness as the right patch? Can you explain what you see in terms of lateral inhibition?

```
Clear[y];
low = 0.2; hi = 0.8;
left = 0.5; right = 0.5;
y[x_] := left /; x<40
y[x_] :=
    ((hi-low)/40) x + (low-(hi-low)) /; x>=40 && x<80
y[x_] := right /; x>=80
```



```
Plot[y[x],{x,0,120},PlotRange->{0,1}];
```



Exercise: Hermann grid

Below is the Hermann Grid. Notice the phantom dark spots where the white lines cross. Can you explain what you see in terms of lateral inhibition?

```
width = 5; gap = 1; nsquares = 6;
```

```
hermann = Flatten[Table[{Rectangle[{x, y}, {x + width, y + width}]},
{x, 0, (width + gap) * (nsquares - 1), width + gap},
    {y, 0, (width + gap) * (nsquares - 1), width + gap}], 1];
```

```
Show[Graphics[hermann, AspectRatio -> 1]];
```