Introduction to Neural Networks
U. Minn. Psy 5038
Spring, 1999
Daniel Kersten

Problem Set 2
(see Lecture 5 - Lateral inhibition)

Exercises

Exercise 1

This exercise is easiest if you copy and modify bits of code from Lecture 5 used to illustrate Mach bands. Make a **DensityPlot** of a 30x30 pixel step function (rather than the ramp function used in Lecture 5). As with the ramp in Lecture 5 illustrating Mach bands, the pattern should only change in the x-direction. Now, using the same parameters as were used for the ramp input in Lecture 5, find the response produced by the recurrent lateral inhibition equations.

Look at the brighter knee of the density plot. Do you perceive a bigger Mach band for the step than for the ramp?

Consider the model predictions. Does the model predict a bigger Mach band (at the bright knee) for the step than for the ramp?

Exercise 2

Use the function **NestList[]** (instead of the **Nest[]** used in Exercise 1) to produce a 12x30 matrix, called **responses** whose rows are the responses or state vectors for each of 12 sequence of iterations or states in Exercise 1. Although we can't view the dynamical trajectory in state space of a 30 dimensional state vector (30 neurons), you can extract the activities at each of the twelve states of just 2 neurons at locations 15, and 16. Do this using the matrix extraction rule which produces **submatrix** from matrix **m**:

submatrix = m[[Range[rowi, rowj], Range[columni,columnj]]];

Then use **ListPlot** to plot the twelve points. Do not use **PlotJoined->True**, but do use **PlotRange**, and **AxesOrigin** to present the state vector evolution clearly.

Exercise 3

Make two graphs to compare the response of the network to the step function with and without self-inhibition using the following initialization parameters:

```
size = 30;
spaceconstant =5;
maxstrength = 0.05;
iterations = ?;
epsilon = .3;
```

How many iterations are required before the network stabilizes with self-inhibition? Without self-inhibition?

Exercise 4

Given the above ramp input and weight matrix, find the steady-state solution to the limulus equation, by solving a set of simultaneous linear equations using the **Inverse[]** function that computes the inverse of a matrix. Plot the solution along with the input pattern **e**, as we did in the lecture. By the way, there is a *Mathematica* function, **IdentityMatrix[],** you might find useful.

This steady state solution can be viewed as a simple linear feedforward network. Use **ListPlot** to show that the effective receptive field of neurons in this feedforward network have a center-surround organization. Just show the plot for effective weights of neuron 15.