Introduction to Neural Networks
U. Minn. Psy 5038

Daniel Kersten

Problem Set 1

Problem Set 1

Exercise 1

1A. Let **w** be a weight vector representing a template pattern. Let {**x**} be a collection of pattern vectors all of unit length. Show theoretically that the cross-correlator gives maximum response to the pattern which matches the form of the template pattern. Recall that the response is determined by the dot-product of the input vector with the weight vector.

1B. Now write a *Mathematica* program to demonstrate this property of cross-correlators. Use the **Table** function to fill a 32x32 matrix **R** with random numbers. Use the built-in function **Random[].** Then define a function **normalize[x]** that takes as input a vector **x**, and returns a normalized version of **x**. Use the **Table** function again to turn **R** into a matrix **R2** whose rows are normalized to unit length. Calculate the matrix product of **R2** with the 8th row of **R2**. Use **ListPlot** to show that the maximum of the product occurs at element 8. Make several more plots using other rows of **R2**, and show the maximum always occurs at the row that matches the input vector.

Exercise 2

Use a set of rules to define a semi-linear "squashing" function, **limit[x]**, which is:

-1 for x < -1;

x for 1 >=x >= -1;

1 for x> 1.

Plot **limit[x]** from x = -2 to 2.

Exercise 3

Using *Mathematica*'s ability to find derivatives of functions, define a function **dsquash[]** to be equal to the derivative of the logistic function:

```
squash[r_] := 1/(1 + Exp[-r]);
```

Plot dsquash from r = -2 to 2.

Mathematica Hint: You can't just define a function dsquash[x_]:=D[squash, etc..]. But there are (at least three ways of doing it).

1) You can use the function Evaluate[ ] to do the define dsquash all in one line.

2) Alternatively, you may wish to use the *Mathematica* rule for replacing a variable with a value in an expression. This would also enable you to define the derivative function all on one line.

3) Otherwise, a brute-force method is to compute the derivative, copy it, and then turn that copied cell into an **input cell** type. (Use **Cell menu>Convert To**).

Later on, when we study back-propagation networks we will need to use the derivative of the non-linear squashing function in our derviation of a learning rule for neural networks. For this reason, it is useful to have a squashing function that has a closed form solution for the derivative.

Exercise 4

There are neurons in the primary visual cortex of mammals called "simple cells". One model for these cells is a linear cross-corelator followed by a thresholding non-linearity (e.g. the half-wave rectification of a diode). The receptive field weights of this cross-correlator typically show a "center-surround" organization. In one dimension, a much reduced model weight vector could look like this:

```
w = {-2,-1,6,-1,-2};
```

Define a threshold function **thresh[s]** that is zero for **s** less than zero, and equal to **s** for values of **s** greater than or equal to 0.

Use the above weight vector **w**, and your **thresh[]** function to model the response of a simple cell. What is the response of your cell to an input **x**:

a) **x = {-1,-.5,3,-.5,-1}**

or

b) **x = {2, 1, 0, 1, 2} ?**

Exercise 5 (Requires material in Lecture notes 4 or 5)

Express the vector:

```
h = {1,2,3,4,5,6,7,8} ;
```

as a linear sum of normalized Walsh vectors (feel free to copy and paste code from Lecture 4 or 5). Plot the "spectrum" of **h**. In particular use **ListPlot** to show the spectrum, which consists of the eight values of the projections of **h** onto the 8 Walsh functions. Verify your answer by reconstructing **h** from the projections.