Peter Stanton

10/6/2017

CSS 485

Homework 1

1. [20 points] Consider a single-input neuron with a bias. We would like the output to be 0 for inputs less than 3 and 1 for inputs greater than or equal to 3.

a. What kind of transfer function is required?

A hard limit transfer function.

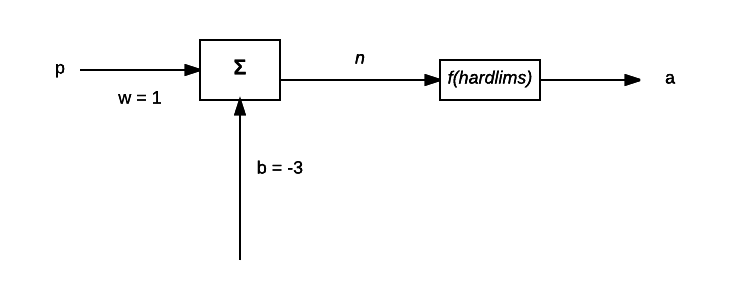
b. What weight and bias would you suggest? Is your bias in any way related to the weight? If yes, how?

a = f(wp+b)

output = f(weight \* input + bias)

We should have a bias of -3 and a weight of 1 so inputs of 3 or less do not exceed 0, and the argument for the hard limit transfer function does not exceed 0, yielding an output of 0. If we have an input greater than 3, we will have an input to the function greater than 0, generating a final output of 1. My bias is unrelated to the weight in this case.

c. Summarize your network by naming the transfer function and stating the bias and the weight. Draw a diagram of the network.



d. Use/write a Matlab function that implements your transfer function, along with a calling function (or script) that uses that transfer function, to implement your neuron design. Verify using Matlab that it performs as expected.

Write a matlab function, based on the inputs, outputs 0 if input is less than 3, or 1 if greater than or equal to 3.

function output = Homework1\_1d(in)

if in < 0

output = 0;

else

output = 1;

end

end

Diary:

in = 5

in =

5

Homework1\_1d(in)

ans =

1

in = -1

in =

-1

Homework1\_1d(in)

ans =

0

diary

2. [20 points] Implement a vector-valued network implementation function. This function should take an input vector, weight matrix, and a bias vector, producing a vector of outputs. In other words, this is a function which can be used to compute the output of multiple neurons simultaneously. (Likely, you will want to start with ensuring that your transfer function works for vector-valued inputs.) Use Matlab to demonstrate that it will work for a two-input, two-output network in which the inputs are binary (0 or 1) and the outputs are binary (0 or 1), with one output being the logical AND of its inputs and one output being the logical OR of its inputs. Note that you do not need to implement a learning algorithm; a little trial and error should allow you to set reasonable weight and bias values.

C = a + b;

Write a function in Matlab,

Function a = layer(p,w,b);

1-dimensional matrices are treated as vectors.

% I worked on this with Kate. For some reason my Matlab session was messed

% up. :(

function [n, and,or] = Homework1\_2(p,W,b)

%multiple neuron function

n = ((W\*p.') + b.');

and = all(n);

or = any(n);

end

 3. [20 points] Chapter 3 of the textbook designed three different neural networks to distinguish between apples and oranges, based on three sensor measurements (shape, texture, and weight). Suppose that we want to distinguish between bananas and pineapples:

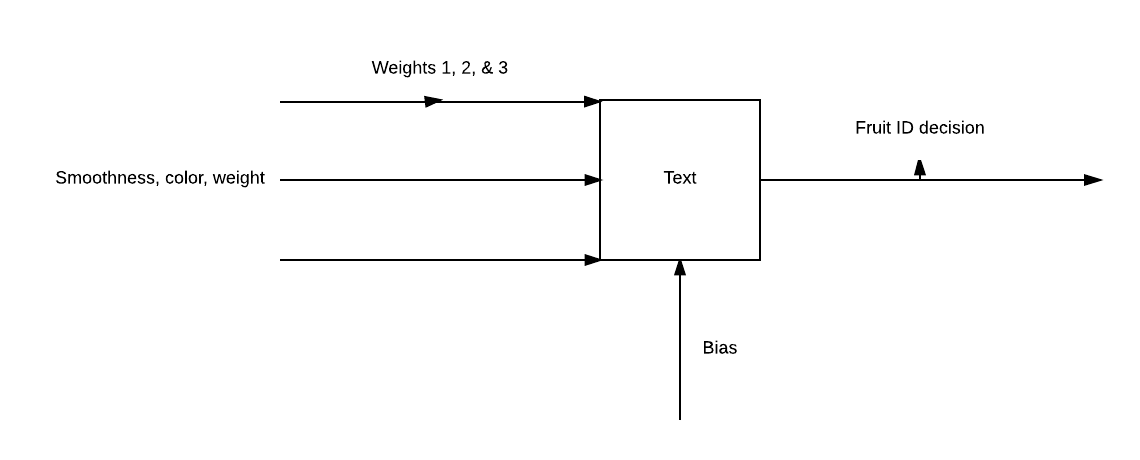
**p**1**p**2==⎡⎣−11−1⎤⎦(Banana)⎡⎣−1−11⎤⎦(Pineapple)

Design a perceptron to recognize these patterns. Explain what choices you needed to make. Present your weights and biases and use Matlab (likely, your code from question 2 will be useful) to illustrate that this network works.

We will have a 3-input, single layer neuron perceptron network. There will be a three-element weight matrix, a three-element input vector, and a bias. The input vector will contain values representing color, weight, smoothness, or whatever.

See page 65 of the textbook.

**Perceptron Design:**



Note: uses a symmetrical hard limit function.

Note that the decision boundaries are between [-1,1,-1], and [-1,-1,1].

function output = calcFruit(p,w,b)

n = ((w\*p.') + b.');

a = shardlim(n);

end

function a = shardlim(n)

%find decision boundary

%boundary is a diagonal line with slope of 1.

if n > 0

a = "Banana"

else

a = "Pineapple"

end

end