A Neural Network for the JavaScript programmer

Ian Channing

November 25, 2018

The beginning

Young man, in mathematics you don't understand things. You just get used to them. — John Von Neumann

Let's generate some random data, visualize it and train a neuron to classify it

Inspired / blatantly copied from

Funfunfunction NN playlist ... but it's missing maths

deeplearning.ai week 2 ... but code isn't open, filling in blanks

NN & DL course ... but no code

So I want to try and give some respect to the code & the maths

I want it to display random values

Generate random test and training sets

```
function rand(min, max) {
  return Math.random() * (max - min) + min;
}
rand(1,3);
rand(0,400); // x, y range for our graph
```

Stretch (*) and shift (+)

```
rand(0,1) -->  rand(1,3)
```

```
+-----+
+-----+-----+          (Stretch by (3 - 1))
      +-----+-----+    (Shift by 1)
0     1     2     3
```

I want to generate a set of random test values

I want to generate a set of random examples

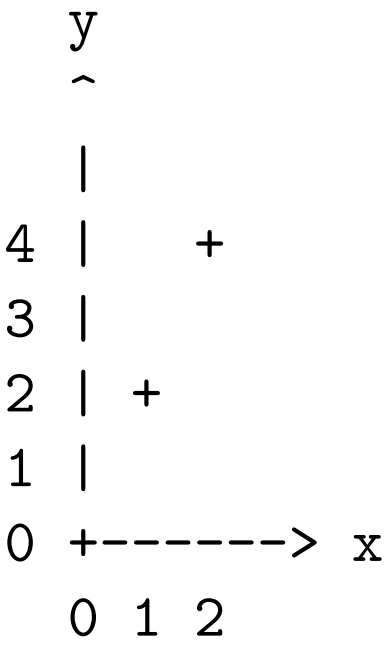
Slight digression (it'll be worth it in the long run)

JavaScript's `map` and `reduce` functions in maths

Reduce the gap between maths and code

$$y = f(x) = 2x$$

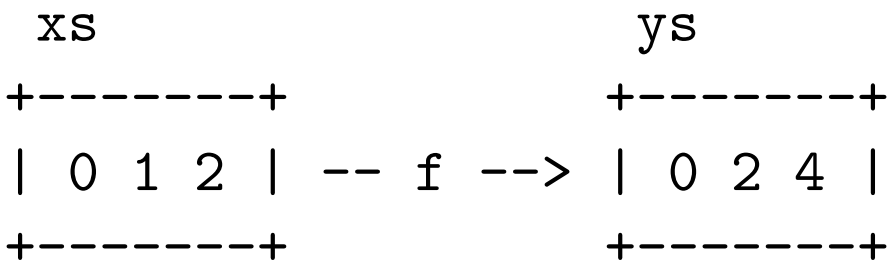
Let's draw a graph



Mathsy definitions

This is actually University level maths - Set Theory.

What's the mathsy name for:
I've got one 'set' and I want to go to another 'set'?



Mathsy definitions

Mapping!

$f(x)$ in JavaScript

$$y = f(x) = 2x$$

```
function f(x) {return 2 * x;}
var xs = [0,1,2];
var ys = xs.map(f); // [0,2,4]
```

$2 + 2 + 2$

$$y = \sum f(x) = \sum 2x$$

```
 x   2x Running total
 1   2  2
 1   2  4
+1  +2  6
--  --
 3   6
```

```
function sum(t, x) { return t + f(x); }
var xs = [1,1,1];
var y  = xs.reduce(sum, 0); // 6
```

I want to display these test values

I want to draw a circle

I want to draw the test values as circles on a graph

I want to separate these circles with a line

I want to colour the circles red or blue

I want to make the colour depend on which side of the line

I want to say whether my examples are red or blue

I want to make a guess based on x, y whether a circle is red or blue

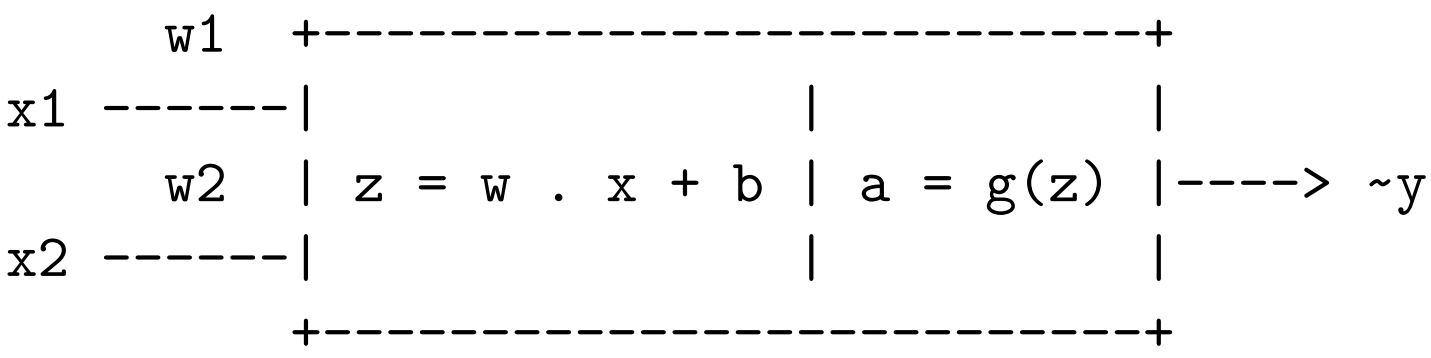
I want to visualise the functions we're going to use to improve the guesses

A neural network of one neuron

> *An Englishman, even if he is alone, forms an orderly queue of one - George Mikes*

Simplify network down to one neuron

Neurons act independently so can scale up process to a network



~y is our approx/guess of y, usually called $\hat{y}$ 'y hat'

g is our 'activation' function

Perceptron or neuron?

Originally called a perceptron

Changed to a neuron with the sigmoid activation function

For us:

1. Fully code perceptron
2. Iterate to a neuron

I want to describe a perceptron firing

Perceptron 'fires' when inputs reach a threshold

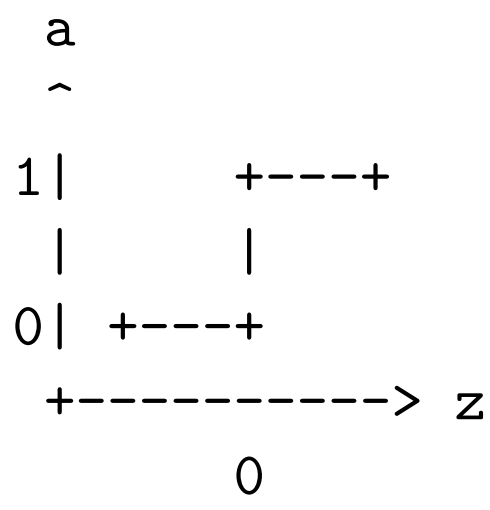
```
                | 0 if w . x <= threshold
activation =    |
                | 1 if w . x > threshold
```

Subtract threshold from both sides and call it 'bias'

```
bias = -threshold
```

```
                | 0 if w . x + bias <= 0
activation =    |
                | 1 if w . x + bias > 0
```

A bit confusing, let's see some code



if then, else...

```
// z = w . x + bias
// a = g(z)
function activation(z) {return (z <= 0) ? 0 : 1;}
// if (z <= 0) return 0; else return 1;
```

Easiest function you can write → basis for all AI

Someone somewhere is having a laugh

I want to multiply two single row / column matrices

Total the inputs using vector dot product / weighted sum

```
w . x = [w1 w2]|x1|
               |x2|
```

In code a vector is an array / list

```
function dot(w, x) {return w[0] * x[0] + w[1] * x[1];}
```

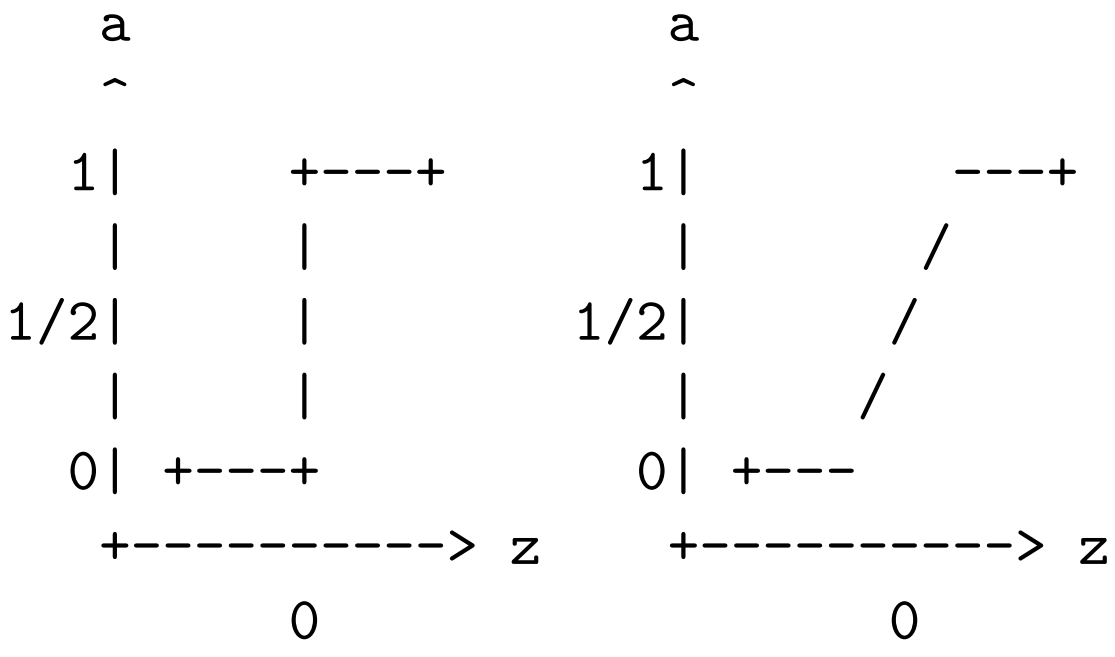
When scaling to a network change vectors to matrices (2D array)

I want to specify the cost function

todo...

Sigmoid neuron

Smooth curved perceptron



Todo ...

I want to specify the cost function

Let's meet the the cross entropy cost function.

The bit we use is the derivative for back-propagation in eqn (61)

$dC/dW_j = 1/n * \sum_x x_j(g(z) - y)$

To be continued...

I want to explain why we get bias/over-fitting

Here we loop around our examples just once. But for more complex problems we loop over the same examples thousands of times. When you say the same word a thousand times over you start to notice tiny details about the word that aren't relevant. e.g. conscience, that's actually con-science but that's totally irrelevant. Neural Networks have no other ideas about the world except for the examples we give them.

In summary

Generated random set of training and test data that we displayed on a graph for testing

Split the points on the graph using an abritrary line (why? back propagation needs linear separation)

Used a perceptron with an activation function and back propagation algorithm

Trained this perceptron to adjust two weights that then colour the test points depending on which side of the line

This is one step of gradient descent

'Improved' this with a sigmoid activation function and it's differentiated back propagation.

The end

> *implementing it myself from scratch was the most important — Andrej Karpathy talking to Andrew Ng (2018)*
> *What I cannot create, I do not understand. — Richard Feynman (1988)*
> *What you really want is to feel every element (and the connections between them) in your bones. — Michael Nielsen (2019)*

Inspired / blatantly copied from:

- Funfunfunction NN playlist
- deeplearning.ai week 2
- NN & DL course