A Neural Network from scratch in JavaScript

Ian Channing
https://github.com/ianchanning/neural-network-js

February 12, 2019

My background

Maths and Computer Science at Imperial, London

JavaScript / React for Imec

All AI knowledge from online courses (In Andrew Ng we trust)

The beginning

What I cannot create, I do not understand.
— Richard Feynman [8] (1988)

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

Inspired / blatantly copied from

Funfunfunction NN playlist [3]

... but it's missing maths

deeplearning.ai week 2 [4]

... but code isn't open, filling in blanks

Neural Networks & Deep Learning course [5]

... but no code

Aim: respect the code & the maths

Get ourselves setup

Install VS Code https://code.visualstudio.com

Download & extract the zip
https://github.com/ianchanning/neural-network-js

Open `index.html`

Open Browser tools (F12)

Start the coding

```
<script src="neural-network.skeleton.js"></script>
<script>
  nn();
</script>
```

Wrap code inside a function to avoid evil global scope [9]

```
function nn() {
  // all your var belong to us
}
```

Skeleton

The outline of what we're going to produce

```
// data
function generator() {}

// SVG chart elements
function chart() {}

// perceptron / neuron
function neuron() {}

// generator + neuron + chart
function build() {}

// draw the chart to root `<div>`
function draw() {}
```

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
```

Slight digression (humour me)

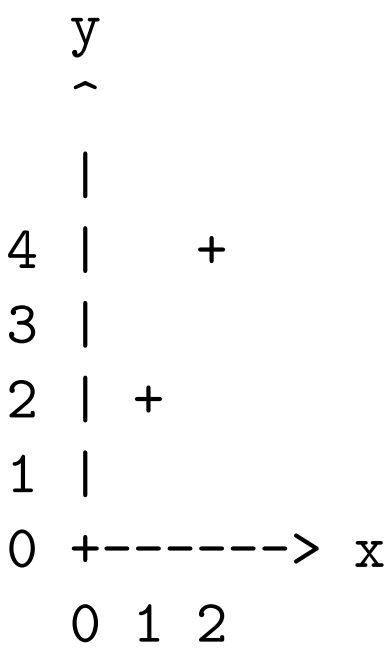
Code <3 Maths

JavaScript's `map` functions in maths

Reduce the gap between maths and code

Let's draw a graph

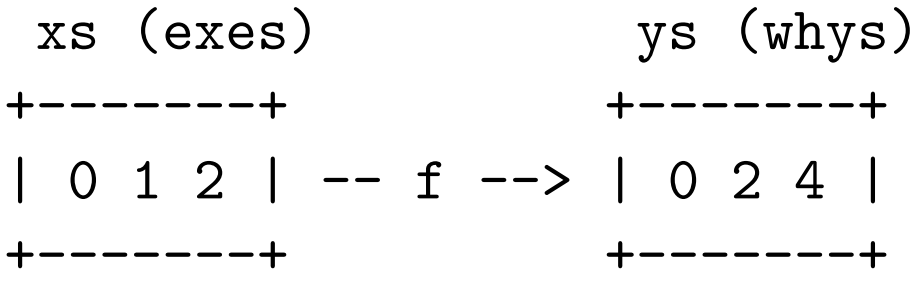
$$y = f(x) = 2x$$



Mathsy definitions

What's the mathsy name for:

I've got one 'set' and I want to go to another 'set' using f?

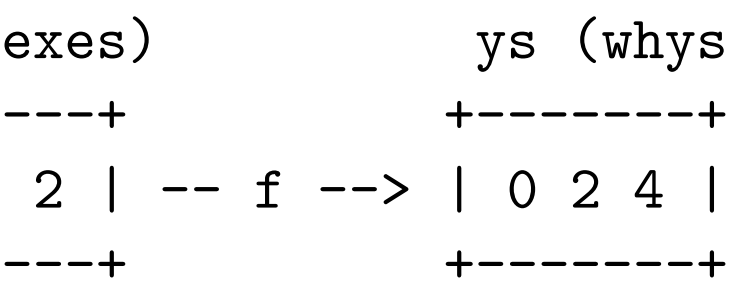


(This is actually University level maths - Set Theory)

Mathsy definitions

What's the mathsy name for:

I've got one 'set' and I want to go to another 'set' using f?



(This is actually University level maths - Set Theory)

Mapping! `f` 'maps' `0,1,2` on to `0,2,4`

$f(x)$ 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]
```

`map` is awesome. Kill all loops!

What's the point?

Our graph will be made up of `{x, y}` points.

One random point in JavaScript:

```
var point = {
  x: rand(0, 400),
  y: rand(0, 400)
};
```

I want to generate a set of random test values

Perhaps I should use a for loop? (never!)

Generate an empty array and use that to generate our new set.

```
function points(length) {
  return Array(length)
    .fill(0)
    .map(function(i) {
      return {x: rand(0, 400), y: rand(0, 400)};
    });
}
```

Mapping `[0,0,0] ---> [{x,y},{x,y},{x,y}]` (demo?)

Make `rand` and `points` available, functions are passed as values

```
return {rand, points};
```

I want to display these test values

Gonna need a graph mate, how does that SVG work again?

Should've read CSS-Trick's excellent guide on SVG Charts [10]

```
<svg
  version="1.1"
  xmlns="http://www.w3.org/2000/svg"
  height="400"
  width="400"
>
  <circle cx="90" cy="192" r="4"></circle>
</svg>
```

Brain shift required: `(0,0)` is top left

Putting that in JavaScript

```
function chart(height, width) {
  // <name xmlns="..."></name>
  function element(name) {
    var ns = "http://www.w3.org/2000/svg";
    return document.createElementNS(ns, name);
  }
  // <svg ...></svg>
  function svg() {
    // JS note: svg() can access element()
    // var s is private to svg()
    var s = element("svg");
    s.setAttribute("height", height);
    s.setAttribute("width", width);
    return s;
  }
}
```

I want to draw the circle

```
// centre is a point {x,y}
// <circle cx="0" cy="0" r="4" fill="blue"></circle>
function circle(centre, radius, colour) {
  var c = element("circle");
  c.setAttribute("cx", centre.x);
  c.setAttribute("cy", centre.y);
  c.setAttribute("r", radius);
  c.setAttribute("fill", colour);
  return c;
}
```

Make `svg` and `circle` available, functions are passed as values

```
return {svg, circle};
```

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

I smell a `map`. I want to map my test values onto the graph.

```
function build(generator, chart) {
  var mySvg = chart.svg();
  generator.points(100).map(function(point) {
    mySvg.appendChild(chart.circle(point, 4, "black"));
  });
  return mySvg;
}

function draw() {
  ...
  var svg = build(generator(), chart(400, 400));
  document.getElementById("root").appendChild(svg);
}
```

And... watch the magic

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 generate a set of random examples

More JavaScript maths

JavaScript's `reduce` function

$2 + 2 + 2$

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

```
 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 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

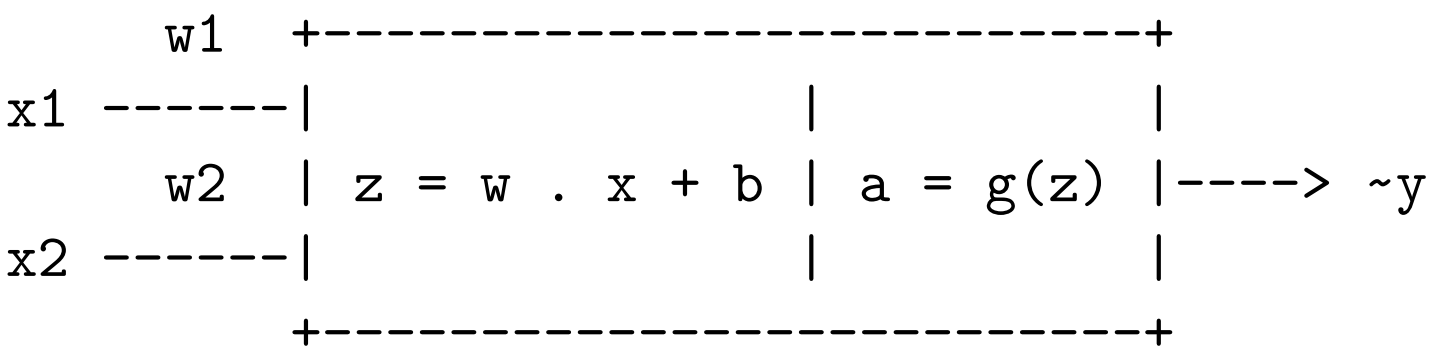
I want to visualise the functions we're going to use to improve the guesses

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

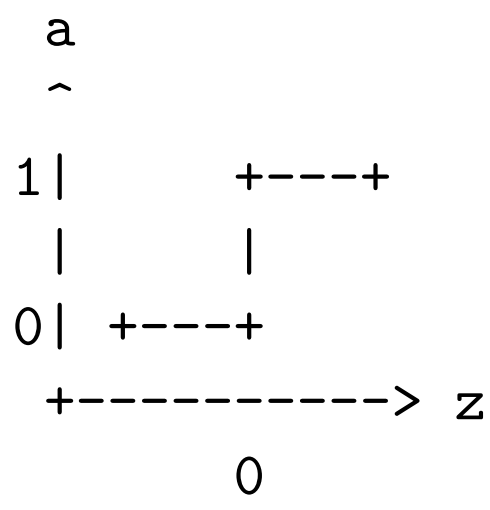
```
                | 0 if w . x <= threshold
activation =    |
                | 1 if w . x > threshold
```

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];}
```

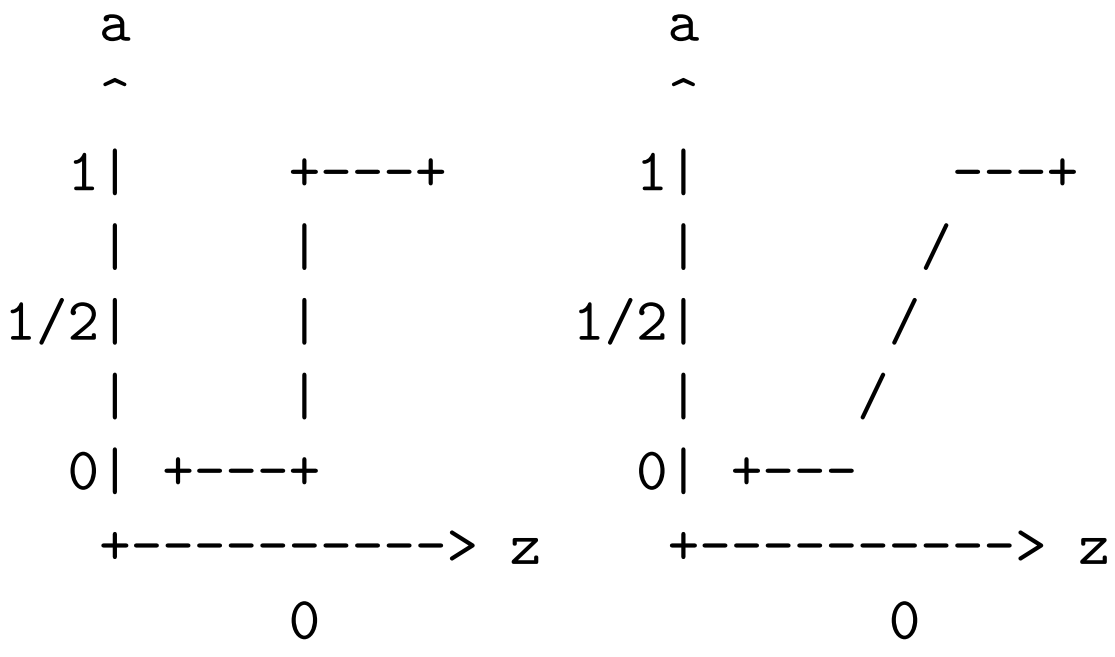
When scaling to a network change vectors to matrices (2D array)

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)*

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

> *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