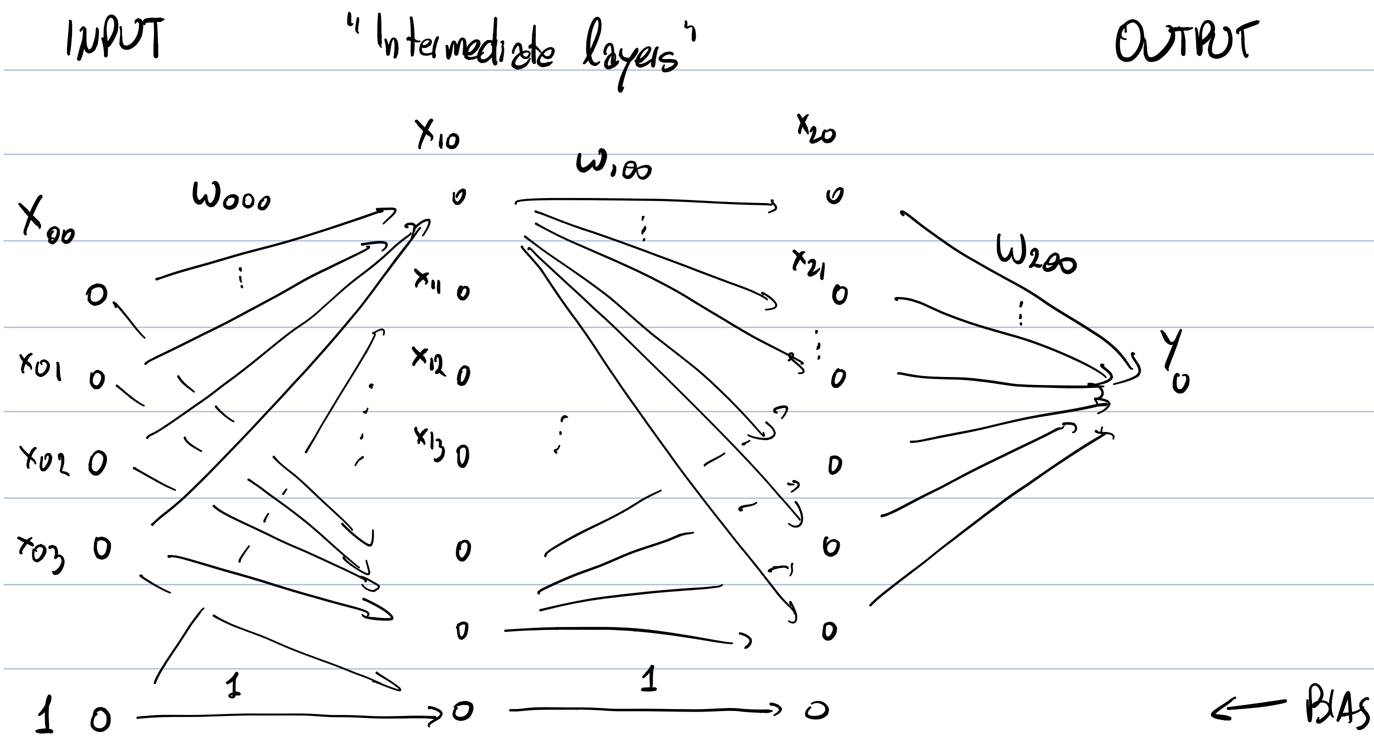
NEURAL NETWORKS

CIML, ch. 10

Why "neural networks"?



x_{00}, etc. are "neurons". They "fire" if a combination of their inputs is "large enough". In the first formulations of neural networks,

$$x_{10} = \tanh(w_{000} \cdot x_{00} + w_{001} \cdot x_{01} + w_{002} \cdot x_{02} \cdots)$$

$$x_{11} = \tanh(w_{010} \cdot x_{00} + w_{011} \cdot x_{01} + w_{012} \cdot x_{02} \cdots)$$

$$\vdots$$

$$x_{1n} = \tanh(w_{0n0} \cdot x_{00} + w_{0n1} \cdot x_{01} + w_{0n2} \cdot x_{02} \cdots)$$

X_{0*} is the vector of activations at layer 0.
W_{0**} is the weight matrix at layer 1 ($0 \rightsquigarrow 1$).
tanh is the "link function" or "non-linearity".

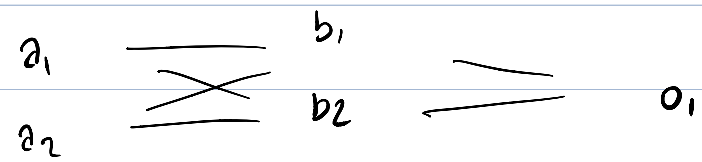
↑ originally had bio inspiration, turns out which link fn is less important than having link fn (why?)

Thm: 2-layer networks are universal (!!!)

"Proof" of much weaker thm:

n-layer network can emit XOR of B^n inputs

BASE CASE



$b_1 = \tanh(\infty \cdot a_1 - \infty \cdot a_2) \leftarrow a_1 \wedge \sim a_2$

$b_2 = \tanh(\infty \cdot a_2 - \infty \cdot a_1) \leftarrow a_2 \wedge \sim a_1$

$o_1 = \tanh(\infty \cdot a_1 + \infty \cdot a_2) \leftarrow b_1 \vee b_2 =$

$(a_1 \wedge \sim a_2) \vee (a_2 \wedge \sim a_1) = a_1 \text{ XOR } a_2$

INDUCTION STEP Next layer, do $b_3 = a_3$ and b_3 XOR $o_1 = o_2$, etc.

"2-layer can emit XOR" is also "easy"

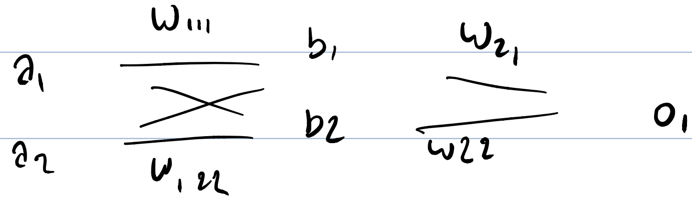
Why NNs? Because they can approximate any function!

Just guess the architecture + non-linearities! Easy! :)

Deep Learning: the dark art of using gradient descent to find the weights of a multi-layer neural network that will minimize some loss.

Problem: we will need gradients of complicated functions!

E.g. what's the gradient of $f(a_1, a_2) = o_1$?



$$o_1 = \tanh(w_{21} \cdot b_1 + w_{22} \cdot b_2)$$
$$b_1 = \tanh(w_{111} \cdot a_1 + w_{121} \cdot a_2)$$
$$b_2 = \tanh(w_{112} \cdot a_1 + w_{121} \cdot a_2)$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$\frac{d \tanh(x)}{dx} = 1 - \tanh(x)^2$$

Ugh...

COMPUTING (WITH) DERIVATIVES

How do we get computers to evaluate derivatives for us?

Numerical Differentiation: $f'(x) \approx \dfrac{f(x+\Delta) - f(x)}{\Delta}$

PROBLEM 1) Subtracting two large values similar to one another is a terrible idea when using floating point numbers: <u>catastrophic cancellation</u>.

PROBLEM 2) When evaluating a gradient of $f: \mathbb{R}^n \to \mathbb{R}$, requires $O(n)$ evaluations

Symbolic Differentiation

Idea: The chain rule is great, let's write a compiler that reads programs that evaluate f, and outputs programs that evaluate f'!

$$d(x+y) = dx + dy$$
$$d(x^k) = k\,x^{(k-1)}\,dx$$
$$d(\log x) = 1/x \; dx$$
$$d \sin x = \cos x \cdot dx$$

Note that expressions inside d on right are simpler than on left. (Process terminates!)

$$d \sin(\sin x) = \cos(\sin x) \cdot (d \sin x)$$
$$= \cos(\sin x) \cdot \cos x$$

$$d \sin(\sin(\sin x)) = \quad \cdots \quad ?$$

PROBLEM: Evaluating symbolic derivative might take <u>much</u> longer.

AUTOMATIC DIFFERENTIATION

Idea: Evaluate function at value and its derivative at the same time. "DUAL NUMBERS"

1) Decide which variable you're taking derivative of
2) Evaluate expression, and its derivative. QED (Really!)

$f(x,y) = xy + \sin x$ Let's compute $\frac{\partial f}{\partial x}(3,0)$

How do we usually evaluate functions? Bottom-up

$f(3,0) = xy + \sin x$ $\frac{\partial f}{\partial x}(3,0) = \frac{\partial}{\partial x}(xy) + \frac{\partial}{\partial x}(\sin x)$ (at $(3,0)$)

$xy\,(3,0) = x\,y$ $\frac{\partial (xy)}{\partial x}(3,0) = x\frac{\partial y}{\partial x} + y\frac{\partial x}{\partial x}$ (at $(3,0)$)

$x\,(3,0) = 3$ $\partial x/\partial x\,(3,0) = 1$

$y\,(3,0) = 0$ $\partial y/\partial x\,(3,0) = 0$

Same for $\sin x$

How do we implement this on computers?

1) Operator overloading!

2) Your own interpreter

This is "forward-mode AD"

PROBLEM: Still needs $O(n)$ evaluations for gradient.

Next class: "Reverse-mode AD"

REVERSE-MODE AD

- Let's state the chain rule slightly differently: (so it works for multiple variables)

$$\frac{\partial s}{\partial u} = \sum_f \frac{\partial s}{\partial f} \cdot \frac{\partial f}{\partial u}$$

(In other words, if s depends on u through multiple variables, the chain rule sums over all of them).

Now let's write our $f(x,y) = \sin x + xy$ as a computation graph:

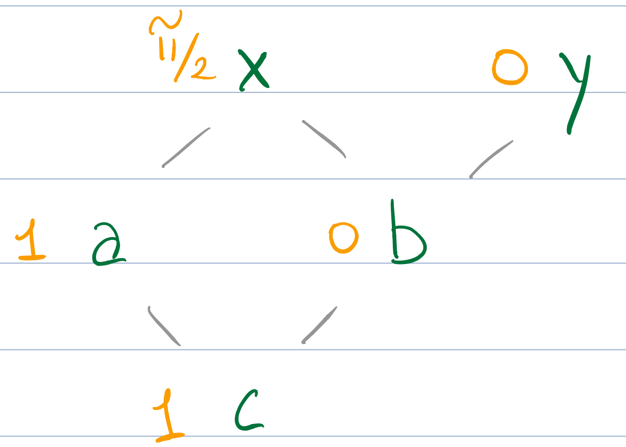
$a = \sin x$ $\qquad$ $da = \cos x \, dx$

$b = x y$ $\qquad$ $db = y \, dx + x \, dy$

$c = a + b$ $\qquad$ $dc = da + db$

$f(x,y) = c$

$f(\tilde{\pi}/2, 0)$:



Forward-mode AD recap

We choose to take derivative wrt x:

$f(\tilde{\pi}/2, 0) =$

$\tilde{\pi}/2 \quad x \quad \frac{dx}{dx} = 1 \qquad 0 \quad y \quad \frac{dy}{dx} = 0$

$1 \quad a \quad \frac{da}{dx} = 0 \qquad 0 \quad b \quad \frac{db}{dx} = 0$

$1 \quad c \quad \frac{dc}{dx} = 0$

If we change variable to y, we must recompute the derivative values:

$f(\tilde{\pi}/2, 0) =$

$\tilde{\pi}/2 \quad x \quad \frac{dx}{dy} = 0 \qquad 0 \quad y \quad \frac{dy}{dy} = 1$

$1 \quad a \quad \frac{da}{dy} = 0 \qquad 0 \quad b \quad \frac{db}{dy} = \tilde{\pi}/2$

$1 \quad c \quad \frac{dc}{dy} = \tilde{\pi}/2$

Note that the only value we wanted were $\frac{dc}{dx}$ and $\frac{dc}{dy}$.
In FW AD, we choose the denominator of the derivative and set the numerator to be the differential of the cell.
What if we flipped this, choosing the numerator, and setting the denominator to be the differential of the cell?

FORWARD-MODE

$x \quad \frac{dx}{dy} =$ $\qquad$ $y \quad \frac{dy}{dy} =$

$a \quad \frac{da}{dy} =$ $\qquad$ $b \quad \frac{db}{dy} =$

$c \quad \frac{dc}{dy} =$

REVERSE-MODE

$x \quad \frac{dc}{dx} =$ $\qquad$ $y \quad \frac{dc}{dy} =$

$a \quad \frac{dc}{da} =$ $\qquad$ $b \quad \frac{dc}{db} =$

$c \quad \frac{dc}{dc} =$

What's our variable of interest in this case? c!

$\pi/2 \quad x \quad \frac{dc}{dx} = ?$ $\qquad$ $0 \quad y \quad \frac{dc}{dy} = ?$

$1 \quad a \quad \frac{dc}{da} = ?$ $\qquad$ $0 \quad b \quad \frac{dc}{db} = ?$

$1 \quad c \quad \frac{dc}{dc} = 1$

$a = \sin x \qquad da = \cos x \, dx$

$b = x y \qquad db = y \, dx + x \, dy$

$c = a + b \qquad dc = da + db$

$f(x,y) = c$

Now we proceed from c upwards, or backwards:
("back propagation")

$\tilde{\pi}/2 \quad x \quad \frac{ds}{dx} = 0+0 \qquad 0 \quad y \quad \frac{ds}{dy} = \tilde{\pi}/2$

$1 \quad a \quad \frac{ds}{da} = 1 \qquad 0 \quad b \quad \frac{ds}{db} = 1$

$1 \quad c \quad \frac{ds}{dc} = 1$

s

$$\frac{ds}{da} = \frac{ds}{dc} \cdot \frac{dc}{da} = 1 \qquad c = a + b \quad \frac{dc}{da} = 1$$

$$\frac{ds}{db} = \frac{ds}{dc} \cdot \frac{dc}{db} = 1 \qquad \frac{dc}{db} = 1$$

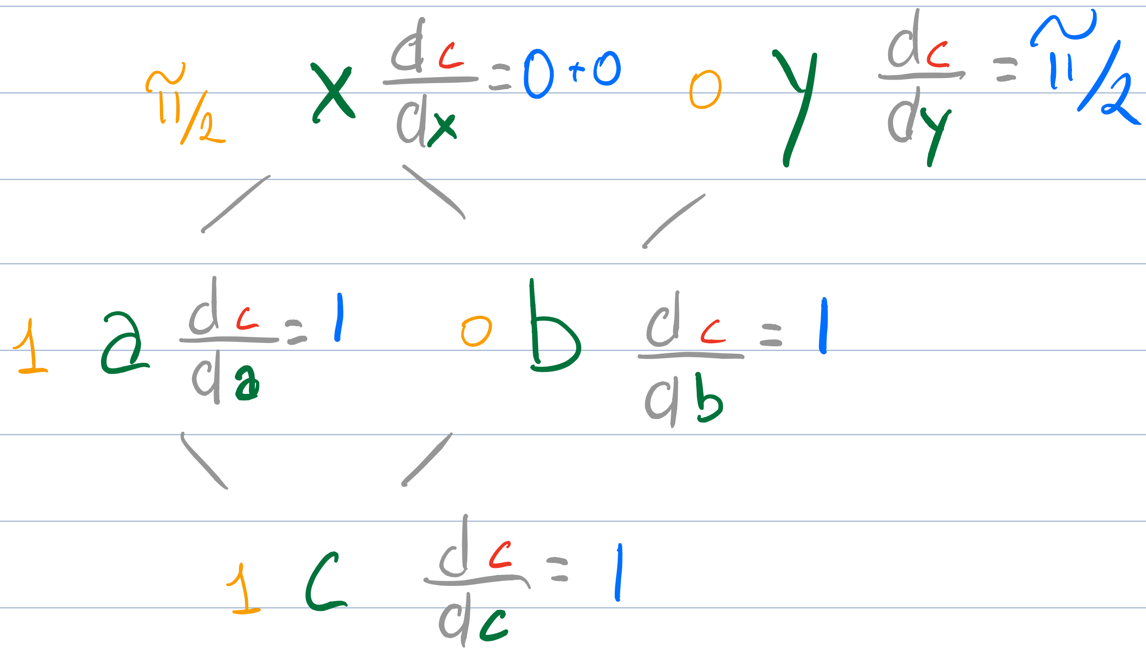
$$\frac{ds}{dx} = \frac{ds}{da}\,\frac{da}{dx} + \frac{ds}{db}\,\frac{db}{dx} \qquad a = \sin x \quad \frac{da}{dx} = \cos x$$

$$= 1 \cdot \cos \tilde{\pi}/2 + 1 \cdot 0 \qquad b = xy \quad \frac{db}{dx} = y$$

$$= 0$$

$$\frac{ds}{dy} =$$

Slightly different algorithm: bottom node "propagates upwards" terms of the chain rule it is responsible for:



$dc = da + db = 1\,da + 1\,db$ "Add 1 to $\frac{dc}{da}$, Add 1 to $\frac{dc}{db}$"

$da = \cos x \; dx = 0\,dx$ "Add 0 to $\frac{dc}{dx}$"

$db = x\,dy + y\,dx = \pi/2\,dy + 0\,dx$ "Add $\pi/2$ to $\frac{dc}{dy}$, Add 0 to $\frac{dc}{dx}$"

Two terms of the chain rule are added at different times

Note that the value you send up is multiplied by the derivative of the cell (it just turned out that those values were 1 in this example!)

Exercise: Evaluate the gradient of

$$f(x_1, x_2, y_1, y_2) = \log\left(1 + \exp\left(-(x_1 y_1 + x_2 y_2)\right)\right)$$

at

$x_1 = 1$ $\quad$ $y_1 = 0$

$x_2 = 1$ $\quad$ $y_2 = 0$