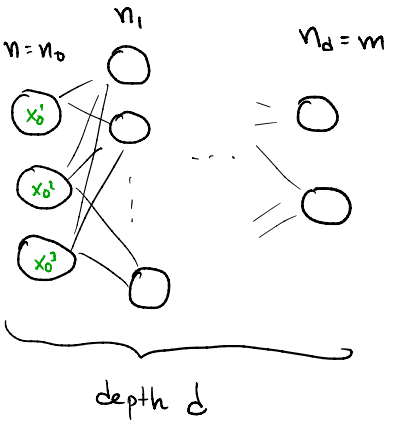
Lecture 1 ~ some comments



Mathematically, a DNN is a function $f: \mathbb{R}^n \to \mathbb{R}^m$

$x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$

defined recursively $x_0 = x$

$$x_1 = A(x_0 \cdot w_1 + b_1) \in \mathbb{R}^{n_1}$$

$$x_2 = A(x_1 \cdot w_2 + b_2) \in \mathbb{R}^{n_2}$$

$$\vdots$$

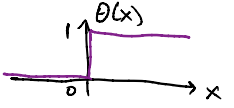
$$f(x) = x_d = A(x_{d-1} \cdot w_d + b_d) \in \mathbb{R}^{n_d} = \mathbb{R}^m$$

Observe: if A is trivial ($A(x) = x$), then f collapses to a single linear function

$$f(x) = x \cdot \vec{w} + \vec{b}$$ (for suitable $\vec{w}, \vec{b}$).

Boring! Linear fits.

A more interesting limit: $A(x) = \theta(x)$ step function



↝ "perceptron" networks from 50's & 60's

model for (iterative, inter-dependent) decision making. (See supplementary reading.)

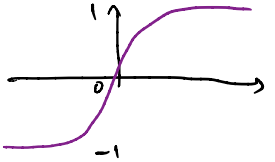
Problem w/ $A(x) = \theta(x)$: not differentiable (not even cts) $\rightsquigarrow$ difficult to <u>learn</u>

Learning: how to alter parameters w, b to improve the outcome of the model.

Rob: in much of modern ML, $A(x)$ is taken to be a differentiable function, w/ a nice derivative

e.g. $A(x) = \sigma(x) = \frac{1}{1-e^{-x}}$ or $A(x) = \tanh(x)$

$A'(x) = A(x)(1 - A(x))$



$A'(x) = 1 - A(x)^2$

and "outcome" is measured by a loss function $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$

$$\ell(x, y) = L(f(x), y) \quad \overset{\text{e.g.}}{=} \quad \frac{|f(x) - y|^2}{2}.$$

Aside: given a metric space W and a function $\ell : W \to \mathbb{R}$

one can define gradient flow on W.

In local coords w^i, metric $g_{ij}\, dw^i dw^j$ (or $g_{ij} := g\left(\frac{\partial}{\partial w^i}, \frac{\partial}{\partial w^j}\right)$)

the gradient flow ODE is $\boxed{\frac{\partial w^i}{\partial t} = -\, g^{ij} \frac{\partial \ell}{\partial w^j}}$.

With discrete steps, this becomes $\Delta w^i = -\, c\, g^{ij} \frac{\partial \ell}{\partial w^j}$.

(c: learning rate)

For a DNN, really $\ell : \mathbb{R}^n \times \underbrace{\mathbb{R}^{n n_1 + n_1 + n_1 n_2 + n_2 + \cdots + n_{d-1} n_d + n_d}}_{\text{space of } w,b \text{ params, assumed vec space w/ Euclidean metric}} \times \mathbb{R}^m \to \mathbb{R}$ $\qquad \ell(x; w, b; y) = \frac{|f_{w,b}(x) - y|^2}{2}$

$g^{ij} = \delta^{ij}$

$\ell(x,y) = L(f(x), y)$ $\qquad$ x, y are fixed, only care about w, b dependence

$$d\ell = \frac{\partial L}{\partial f(x)}\, df(x) = \frac{\partial L}{\partial f(x)} A'(x_{d-1} \cdot w_d + b_d)\,(x_{d-1} \cdot dw_d + db_d + dx_{d-1} \cdot w_d)$$

$\frac{\partial \ell}{\partial w_d} \sim \frac{\partial L}{\partial f} A'(x_{d-1} w_d + b_d)\, x_{d-1}$, $\frac{\partial \ell}{\partial b_d} \sim \frac{\partial L}{\partial f} A'(\cdots)$ (with appropriate index contractions, transposes, etc.)

$$= \frac{\partial L}{\partial f} A'(x_{d-1} w_d + b_d)\left(x_{d-1}\, dw_d + db_d + A'(x_{d-2} \cdot w_{d-1} + b_{d-1})(x_{d-2} \cdot dw_{d-1} + db_{d-1} + dx_{d-2} \cdot w_{d-1})\, w_d\right)$$

($\frac{\partial \ell}{\partial w_{d-1}}$, $\frac{\partial \ell}{\partial b_{d-1}}$; keep going)

$\rightsquigarrow$ "back-propagation" to compute components of the gradient.