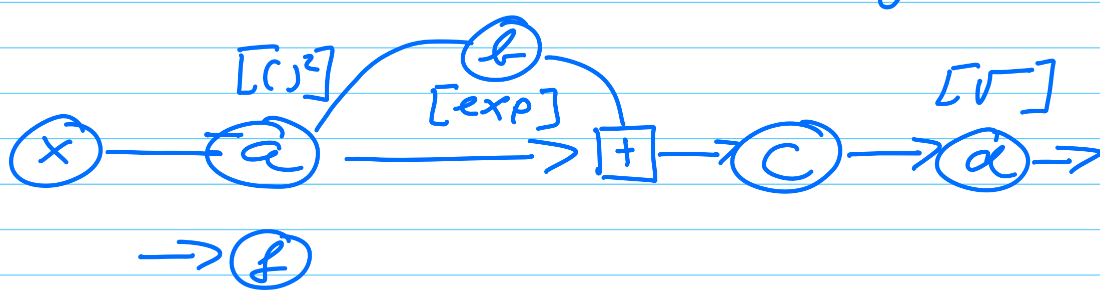
FYS 5429, FEB 8, 2023

Back prop & Autodiff

$$f(x) = \sqrt{x^2 + \exp(x^2)}$$

$$a = x^2 \quad b = \exp(a)$$

$$c = a + b \qquad d = \sqrt{c} = f(x)$$



Forward mode

$$\frac{da}{dx} = 2x \qquad \frac{db}{dx} = \frac{db}{da}\frac{da}{dx}$$

$$\frac{dc}{dx} = \left[\frac{dc}{da}\frac{da}{dx} + \frac{dc}{db}\frac{db}{dx}\right]$$

$$= \left[\frac{dc}{da}\frac{da}{dx} + \frac{dc}{db}\frac{db}{da}\frac{da}{dx}\right]$$

$$\frac{dd}{dx} = \frac{dd}{dc}\frac{dc}{dx} = \frac{df}{dx}$$

$$\frac{dd}{dc} = \frac{1}{2\sqrt{c}}$$

$$\frac{df}{dx} = \frac{1}{2\sqrt{c}}\frac{dc}{dx}$$

$$= \frac{x(1+b)}{\sqrt{c}}$$

c appears in $f(x) = \sqrt{c}$

precalculate $\sqrt{c}$

b

$\frac{df}{dx}$ Flops $= 3$?

Reverse mode

$$\frac{df}{dc} = \underbrace{\frac{df}{dd}}_{=1} \frac{dd}{dc} = \frac{dd}{dc} = \frac{1}{2\sqrt{c}}$$

$$\frac{df}{db} = \frac{df}{dc} \underbrace{\frac{dc}{db}}_{=1} = \frac{1}{2\sqrt{c}}$$

$$\frac{df}{da} = \frac{df}{db}\frac{db}{da} + \frac{df}{dc}\frac{dc}{da}$$

$$= \frac{1}{2\sqrt{c}}\left[1 + \exp(a)\right]$$

$$= \frac{1}{2\sqrt{c}}\left[1 + b\right]$$

$$\frac{df}{dx} = \frac{df}{da}\frac{da}{dx} = \frac{x(1+b)}{\sqrt{c}}$$

Autodiff is a formalization of this example

assume we have $x_1, \dots x_d$ input variables to $x_{d+1} \dots x_{D-1}$ as intermediate variables and x_D = output variable

$x_1 = x$ $\quad d = 1$ in our example

$x_2 = a \quad x_3 = b \quad x_4 = c$

$x_D = d = f$

for $i = d+1, D$

$$x_i = g_i(x_{pa(x_i)})$$

g_i are elementary functions and $x_{pa(x_i)}$ are the parent nodes of variable x_i in the graph

$$g_2 = (\cdot)^2 = a$$

$$g_3 = \exp(\cdot)$$

$$g_4 = c = a+b$$

$$g_5 = \sqrt{c} = d$$

$$\frac{\partial f}{\partial x_D} = 1 \qquad x_D = f$$

Reverse mode (Backprop)

$$\frac{\partial f}{\partial x_i'} = \sum_{\substack{x_j' \\ x_i' = \mathrm{Pa}(x_j)}} \frac{\partial f}{\partial x_j} \frac{\partial x_j'}{\partial x_i'}$$

Neural Network

$$\Theta = \{ W_1, b_1, W_2, b_2, \ldots W_L\ b_L \}$$

Define $C(\Theta)$

$$\frac{\partial C}{\partial W_L} = \delta^L \odot \underline{a^{L-1}}$$

output from previous layer

$$\delta^L = f'(z^L) \frac{\partial C}{\partial a^L}$$

$$\delta^L = \frac{\partial C}{\partial z^L}$$

$$\delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} f'(z_j^l)$$

$$w_{jk}^l \Leftarrow w_{jk}^l - \eta\, \delta_j^l a_k^{l-1}$$

$$b_j^l \Leftarrow b_j^l - \eta \frac{\partial C}{\partial b_j^l}$$

$$= b_j^l - \eta\, \delta_j^l$$

η = learning rate

- Adagrad
- RMSprop
- ADAM

Paths for Project1 (Part 1)

- Deep learning

FFNN CNN RNN AE GANs

Path 1

solve ODE (Dynamical problem)

$$x_{t+1} = x_t + \underbrace{\text{ODE solver}}_{RK4}$$

Set up a series of initial conditions

input is x_t

output (target) x_{t+1}

FFNN, RNN, AE, GANs

Can extend to partial DEs (PDEs)

- Navier-Stokes
- Schrödinger eq.

Path 2

write own code for CNN and RNN and apply these to data of your choice

Path 3

use tensorflow/pytorch to explore path 1 or apply to own data. As an example use CNNs to imaging.

Path 1

(i) $$m\frac{d^2x}{dt^2} = -kx - \nu\frac{dx}{dt} + F_{ext}(x,t)$$

Set up ~ 100 initial
conditions
x_0, v_0

Call ODE solver and generate
x_t which is an array

$\Delta t = 10^{-2}$ $\quad t =$ np.arange(
0, t_{final}, Δt)

for i in range(100)
nn_input x_t
nn_output x_{t+1}

(ii) Lorentz attractor

$$\frac{dx}{dt} = \sigma(y-x)$$

$$\frac{dy}{dt} = x(\rho-z)-y$$

$$\frac{dz}{dt} = xy-\beta \cdot z$$

$\Delta t = 0{,}01$

$t_{final} = 8$

$\beta = 8/3 \quad \sigma = 10 \quad \rho = 28$

Define 100 random values
for x_0, y_0, z_0

$X_{t+1} = X_t +$ ODEsolver

Generate nn_input $= X_t$
nn_output $= X_{t+1}$

(iii) own data sets