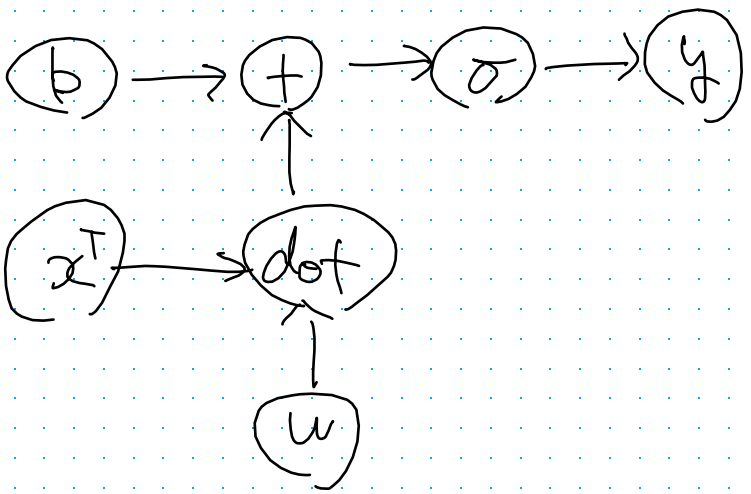
Subject 2|A-Q4 Automatic Differentiation Date 1

Attendants April 12th

(a) $\hat{y} = \sigma(x^T w + b)$ computation graph



(b) What kind of graph is computational graph in general?

→ DAG (Directed Acyclic Graph)

except for RNN.

How many edges can it have at most?

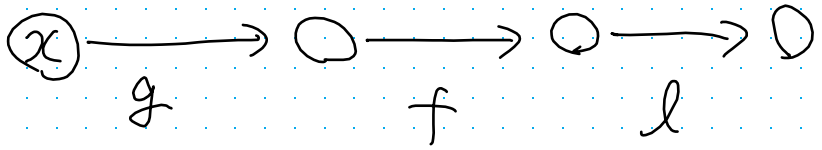
Assume N node

$(N-1 + N-2 + \cdots + 0) = (N-1) \times \frac{N}{2}$

Subject 2|A_Q4 Automatic Differentiation 2 Date April 12

Attendants

(c) Explain how the back prop algo find the grad

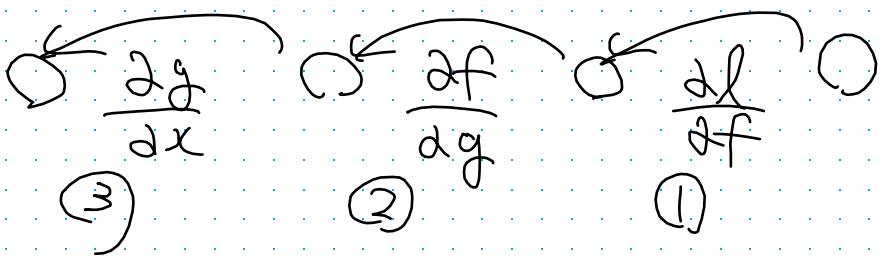


$\nabla(l \circ f \circ g)(x)$

$x \in \mathbb{R}^n$

$\frac{\partial l}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$ can be computed as follows

Back propagation



④ using chain rule

$$\nabla(l \circ f \circ g)(x) = \frac{\partial l}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial x}$$

⑤ Loop num of output dimension

in this case output $\in \mathbb{R}$ then we don't have to accumulate.

Subject 2(H_Q4 Automatic Differentiation Date April 12

Attendants

(d) What is the order of computation required to compute the grad in this case

let T as number of transition, in other words edge.

let D as dimension of output

Then Order of computation of backward is

$$O(TD)$$

Subject 21H_Q4 Automatic Diff Date 4 April 12th

Attendants

(e) $l : \mathbb{R}^n \to \mathbb{R}^k$ $k >> 1$

which mode is prefer to compute $D(\cdot)$ efficiently

Answer

forward mode automatic diff is preferable if $k > n$

otherwise backward mode automatic differentiation is preferable

because forward mode AD computational cost of depends on dim of input (n)

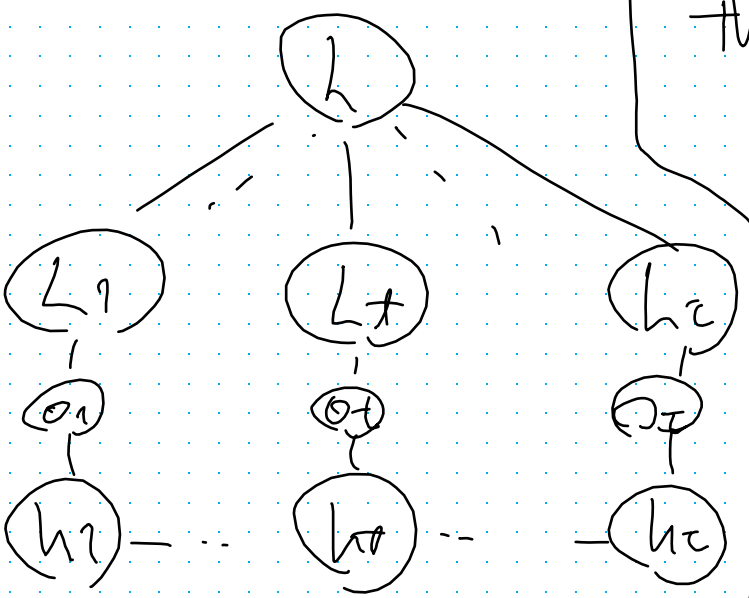
Similarly, reverse mode AD's cost depends on dim of output (k).

Subject 2(f)_Q4 Automatic Diff Date 5

Attendants

(f) (Bonus) Which mode of AD can you use to train an RNN without storing the activation?

How does it work?



the drawback of this approach is computational cost depends on dim of input in general RNN has large dim of output.

Without storing activation we have to use forward mode AD.

Which can be computed at the same time of forward process and doesn't require to store activation

Subject 21H_Q4 AD

Date 6 April (27th

Attendants

(g) Bonus

$$x_{t+1} = x_t - H^{-1}\nabla l(x_t)$$

How can you combine the two mode of AD to compute Hv efficiently for any vector v?

Ans Hessian vector product doesn't calculate H explicitly

step 1. compute grad and store activation as reverse mode at t step

step2 as a forward process using input as x_{t+1} and compute Hv by using t step grad.

Subject Hessian vector Product Date 7 April 12

Attendants

(from JAX Tutrial)

$f: \mathbb{R}^n \to \mathbb{R}$

The Hessian at a point $x \in \mathbb{R}^n$ is written as $\partial^2 f(x)$.

A Hessian-vector product function is then able to evaluate

$$v \mapsto \partial^2 f(x) \cdot v \quad \text{for any } v \in \mathbb{R}^n$$

$$\partial^2 f(x)\, v = \partial \left[x \to \partial f(x) \cdot v \right] = \partial g(x)$$

where $g(x) = \partial f(x) \cdot v$

grad (jnp.vdot (grad(f(x)), v))

jax numpy

Subject Hessian-Vector Product using f Date

Attendants both forward- and reverse- mode April 8th

$$(x, v) \longmapsto \partial^2 f(x)\, v = \partial g(x) \cdot v$$

$$g : \mathbb{R}^n \to \mathbb{R}^n, \quad g(x) = \partial f(x)$$

$$\frac{\partial}{\partial x}\left(\underbrace{\partial f(x)}_{②} \cdot v\right)$$ ①, ③

- fwdrev_hvp(f, x, v)
  - jvp(df, x, v)
- revfwd_hvp(f, x, v)
  - g = jvp(f, x, v)
  - grad(g, x)

fwdと revfwdのパターンでよく使われる

$$\frac{\partial}{\partial x}\left(\underbrace{\partial f(x)}_{①}\right) \cdot v$$ ②

JVP

JVP (Jacobian-Vector Products)

$$f : \mathbb{R}^n \to \mathbb{R}^m \qquad \partial f(x) \in \mathbb{R}^{m \times n}$$

$$(x, v) \longmapsto \partial f(x)\, v$$

Subject Computing Hessian-Vector Products Efficiently Date 9

Attendants

$\partial f(x) \cdot v$ ← compute by forward mode

setting $\dot{x} = v$

$\partial^2 f(x) \cdot v$ ← take the result and apply backprop

[Pros]

it allows to evaluate a function and its derivative at the same time.

⇓

This enable to overcome the two phase doctrine of reverse mode.

(教義)

This comes at cost of storing dual representation of all intermediate node.

<reverse-on-forward> Hvp

はこのメモリ消費とおさえるレ forward での関数評価と速い！

Subject Computing with the Hessian Date 10

Attendants

HVP

⟨key insight⟩

defining $g(w) = \nabla J(w) = \frac{\partial f}{\partial w}$

then Hl is just the Jacobian of g

⇓

This leads to an HVP implementation called forward-over-reverse AD

Subject Second Derivatives Date 11

Attendants from Slide of U Date

How to take HVP $\nabla^2 f(x) \cdot u$

Notation jvp (f, x, u)

$\left(= Dg \cdot u = \frac{\partial}{\partial t} Dg(x + tu) \right)$?

$\boxed{(x, u) \mapsto \nabla f(x) \cdot u}$ ← JAX 2'nd Definition

---

Reverse - over - Reverse

$\nabla (x \longmapsto \nabla f(x) \cdot u)$

Forward - over - Reverse

$\frac{\partial}{\partial t} D(\nabla f)(x + tu) = \text{jvp}(\nabla f, x, u)$

Reverse - over - Forward

$\nabla \left[\frac{\partial}{\partial t} (Df)(x + tu) \right] = \text{Rjvp}(f, x, u)$

Subject Automatic Differentiation Date 12

Attendants

$jvp(f, x, u): (x, u) \mapsto \partial f(x) \cdot u$

$grad(f, x) \quad x \mapsto \nabla f(x)$

1. reverse-over-reverse

[computation] → $grad(dot(grad(f, x), u), x)$

$\nabla_x(\nabla_x f(x) \cdot u) = \nabla_x^2 f(x) \cdot u$

2. forward-over-reverse

[computation] → $jvp(grad(f, x), x, u)$

$\partial \nabla_x f(x) \cdot u = \nabla_x^2 f(x) \cdot u$

3. reverse-over-forward

[computation] → $grad(jvp(f, x, u), x)$

$\nabla_x(\partial f(x) \cdot u) = \nabla_x^2 f(x) \cdot u$