Quantum simulation on a random tensor network

February 3, 2022

1 Differentiating single step time evolution

The m site Rydberg Hamiltonian is

$$H_{\text{Rydberg}} = \sum_{i,j=1,i>j}^{m} \frac{C}{|r_i - r_j|^6} n_i n_j + \Omega(t) \sum_{i=1}^{m} \frac{1}{2}\sigma_i^x + \Delta(t) \sum_{i=1}^{m} n_i \tag{1}$$

For simplicity, we consider the following general representation of a time-space dependent Hamiltonian with k terms

$$H = \sum_{k=1}^{K} c_k O_k \tag{2}$$

where c_k can be dependent on a set of parameters like locations $r_1, r_2, \ldots, r_m$, and pulses $\Omega(t)$ and $\Delta(t)$.

1.1 The ODE version

In each step of the ODE solver, it performs the following update

$$|\psi'\rangle = (1 - iH\Delta t)|\psi\rangle \tag{3}$$

We derive the backward rules for the gradients by inspecting the following equations

$$\begin{aligned} \overline{\mathcal{L}}\delta\mathcal{L} &= \overline{|\psi'\rangle} \circ \delta|\psi'\rangle \\ &= \sum_k \overline{c_k}\delta c_k + \overline{|\psi\rangle} \circ \delta|\psi\rangle + \overline{\Delta t}\delta\Delta t \end{aligned} \tag{4}$$

where $\circ$ is the Hadamard product applied on real numbers, note a complex number in computer is composed of two real numbers. The above euqations has a more elegant linear algebra version as the following.

$$\begin{aligned} \overline{\mathcal{L}}\delta\mathcal{L} &= \overline{\langle\psi'|}\delta|\psi'\rangle \\ &= \sum_k \overline{c_k}\delta c_k + \overline{\langle\psi|}\delta|\psi\rangle + \overline{\Delta t}\delta\Delta t \end{aligned} \tag{5}$$

where we have used $\langle\psi|$ to represent the hermitian conjugate of $|\psi\rangle$.

$$\delta|\psi'\rangle = -i\sum_k \delta c_k O_k \Delta t|\psi\rangle - iH\delta\Delta t|\psi\rangle + (1 - iH\Delta t)\delta|\psi\rangle \tag{6}$$

By observing Eq. (5) and Eq. (6), one can see

$$\overline{\langle\psi|} = \overline{\langle\psi'|}(1 - iH\Delta t) \tag{7}$$

$$\overline{c_k} = \Re\left[-i\Delta t\overline{\langle\psi'|}O_k|\psi\rangle\right] \tag{8}$$

$$\overline{\Delta t} = \Re\left[-i\overline{\langle\psi'|}H|\psi\rangle\right] \tag{9}$$

After a step, a normalization procedure might be called on the wave functions, this is trivial so that we do not discuss it at this stage.

1.2 The `expmv` version

To differentiate the time evolution directly, one can use the Taylor expansion

$$\begin{aligned}|\psi'\rangle &= e^{-iHt}|\psi\rangle \\ &= \sum_{n=0}^{\infty}\frac{(-it)^n H^n}{n!}|\psi\rangle\end{aligned} \tag{10}$$

Similarly, we have

$$\delta|\psi'\rangle = e^{-iHt}\delta|\psi\rangle + \sum_{n=0}^{\infty}\frac{(-it)^n\delta(H^n)}{n!}|\psi\rangle + \left(e^{-iH(t+\delta t)} - e^{-iHt}\right)|\psi\rangle \tag{11}$$

$$\overline{\langle\psi|} = \overline{\langle\psi|}e^{-iHt}\delta \tag{12}$$

$$\bar{t} = \overline{\langle\psi'|} - iHe^{-iHt}|\psi\rangle \tag{13}$$

$$\overline{c_k} = \sum_n \frac{(-it)^n}{n!}\sum_{p=0}^{n-1}\overline{\langle\psi'|}H^p O_k H^{n-p-1}|\psi\rangle \tag{14}$$

2 How to differentiate an ODE solver

2.1 The adjoint state method (or neural ODE)

Since the time evolution is reversible, one can reverse it by doing inverse time evolution (or the adjoint state method [4, 2]).

Algorithm 1: The continuous adjoint state method

input : parameters θ, initial time t_0, ending time t_n, final state s_n and its adjoint $\overline{s_n}$

output: $\overline{s_0}$, $\overline{\theta}$

function aug_dynamics($(s, a, \text{-}), t, \theta$)

  $q = f(s, t, \theta)$ `# the augmented dynamics`

  return $(q, -a^T \frac{\partial q}{\partial s}, -a^T \frac{\partial q}{\partial \theta})$

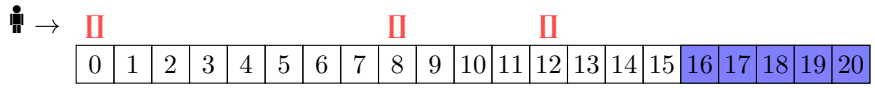
end

$S_0 = (s_n, \overline{s_n}, 0)$ `# initial state of the augmented dynamics`

$(s_0, \overline{s_0}, \overline{\theta}) = \text{ODESolve}(\text{aug\_dynamics}, S_0, \theta, t_n, t_0)$ `# integrate the augmented dynamics in the reversed time`

2.2 The treeverse algorithm (or optimal checkpointing)

For the cases reversibility is not guarented, one can use treeverse algorithm [3, 1]. Peter is a road maintainer. He is assigned a job to paint a one-way road to blue in the reversed order (do not ask me why!). In the following illustration, we represent the road as N grids arranged in one dimension.



He can paint one cell a time as limited the painting material he can carry at one time. Since the road is one-way, Peter soon find once he reaches the working spot, he can not drive back. Lucky enough, Peter picked up some teleportation magics in Hogwarts, he can setup a teleportation gate at where he locates and teleport his car to any existing teleportation gate (marked with red symbols). One limitation of this magic is he can at most create $\delta \leq N$ gates at the same time. When this upper limit is reached, he must distroy an existing teleportation gate to create a new one. Given $N = 10000$, $\delta = 10$, can you please help Peter design a scheme so that he can drive the least?

References

[1] TreeverseAlgorithm.jl. https://github.com/GiggleLiu/TreeverseAlgorithm.jl.

[2] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[3] Andreas Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and software*, 1(1):35–54, 1992.

Algorithm 2: The Treeverse algorithm

input : State cache $S = \{0 : s_0\}$, the initial adjoint $\overline{s_n} \equiv \frac{\partial \mathcal{L}}{\partial s_n}$, the maximum number of checkpoints δ, the number of scan τ, the starting location of current block $\beta = 0$, the end point of current block $\phi = n$, and the bisection point of current block $\sigma = 0$

output: Back-propagated adjoint $\overline{s_0} \equiv \frac{\partial \mathcal{L}}{\partial s_0}$

function treeverse(S, $\overline{s_\phi}$, δ, τ, β, σ, ϕ)
  if $\sigma > \beta$ **then**
    $\delta = \delta - 1$
    $s = S[\beta]$ # load initial state s_β
    for $j = \beta, \beta + 1, ..., \sigma - 1$ **do**
      $s_{j+1} = f_j(s_j)$ # compute s_σ
    end
    $S[\sigma] = s_\sigma$
  end
  # let κ be the division point, call the treeverse algorithm recursively
  while $\tau > 0$ **and** $\kappa = \text{mid}(\delta, \tau, \sigma, \phi) < \phi$ **do**
    $\overline{s_\kappa}$ = treeverse(S, $\overline{s_\phi}$, δ, τ, σ, κ, ϕ)
    $\tau = \tau - 1$
    $\phi = \kappa$
  end
  $\overline{s_\sigma} = \overline{f_\sigma}(\overline{s_{\sigma+1}}, s_\sigma)$ # back propagate the gradient
  if $\sigma > \beta$ **then**
    remove($S[\sigma]$) # remove state s_σ from cache
  end
  return $\overline{s_\sigma}$
end
function mid(δ, τ, σ, ϕ)
  # choose the bisection point
  $\kappa = \lceil (\delta\sigma + \tau\phi)/(\tau + \delta) \rceil$
  if $\kappa \geq \phi$ **and** $\delta > 0$ **then**
    $\kappa = \max(\sigma + 1, \phi - 1)$
  end
end

[4] R.-E. Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 11 2006.