Title: Differentiable Programming Tensor Networks and Quantum Circuits

Speakers: Lei Wang

Collection: Machine Learning for Quantum Design

Date: July 12, 2019 - 11:30 AM

URL: http://pirsa.org/19070017

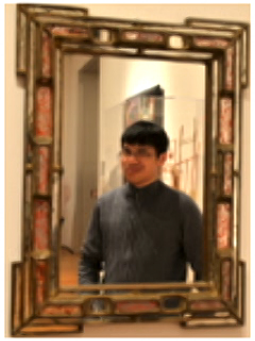
Abstract: Differentiable programming makes the optimization of a tensor network much cheaper (in unit of brain energy consumption) than before [e.g. arXiv: 1903.09650]. This talk mainly focuses on the technical aspects of differentiable programming tensor networks and quantum circuits with Yao.jl (https://github.com/QuantumBFS/Yao.jl). I will also show how quantum circuits can help with contracting and differentiating tensor networks.

∂ifferentiable programming tensor networks and quantum circuits

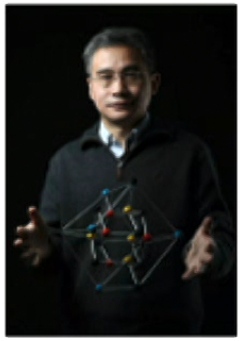
Lei Wang (王磊)
https://wangleiphy.github.io
Institute of Physics, CAS

Collaborators

Hai-Jun Liao



Jin-Guo Liu
(on the market)

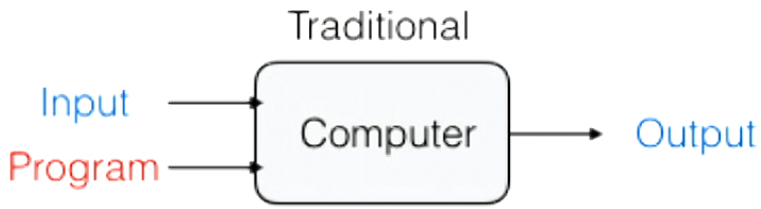


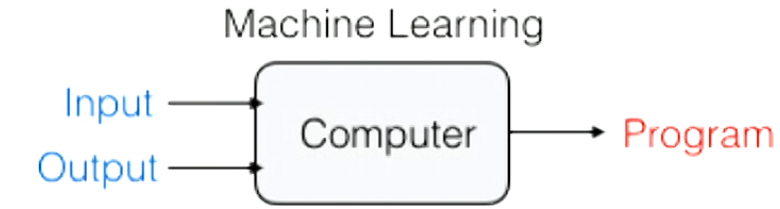
Tao Xiang

1903.09650

https://github.com/wangleiphy/tensorgrad

Differentiable Programming

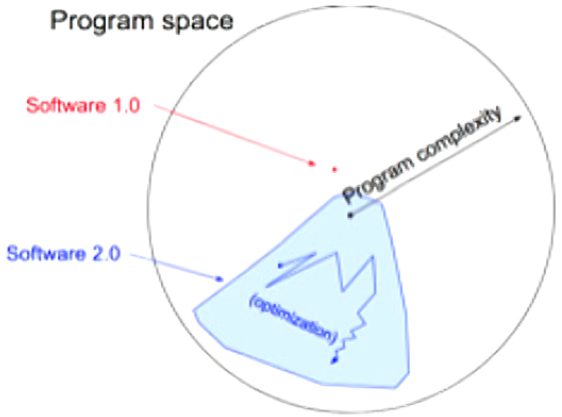




Andrej Karpathy

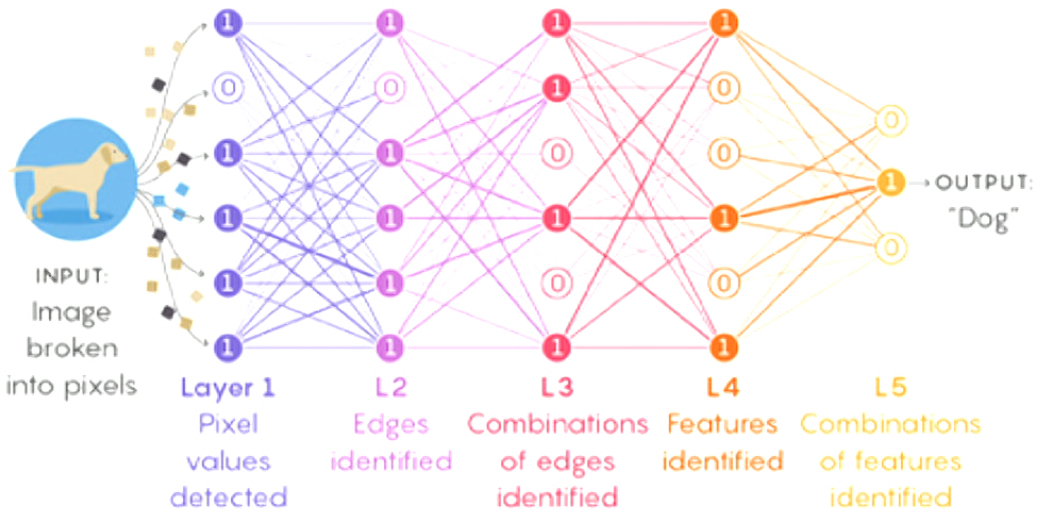
Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

https://medium.com/@karpathy/software-2-0-a64152b37c35



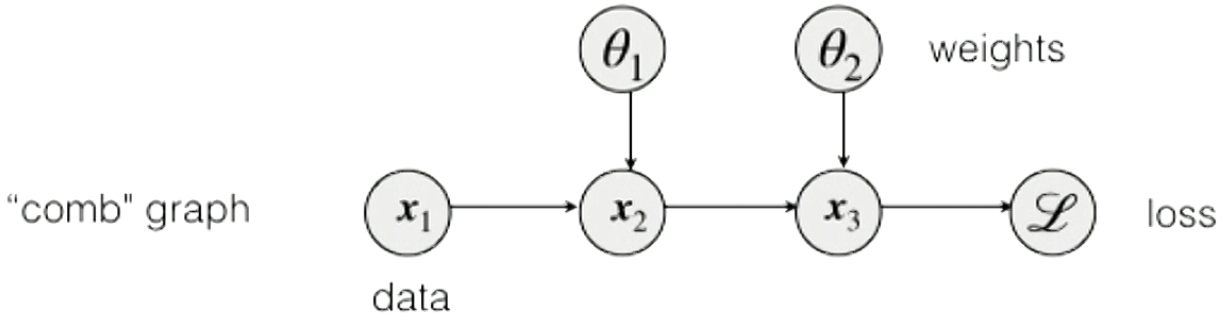
Writing software 2.0 by gradient search in the program space

The engine of deep learning



Compose differentiable components to a program
e.g. a neural network, then optimize with gradient

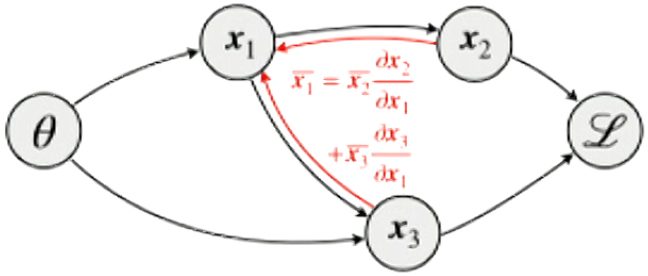
Computation Graph



Pullback the adjoint through the graph

Computation Graph

directed
acyclic graph

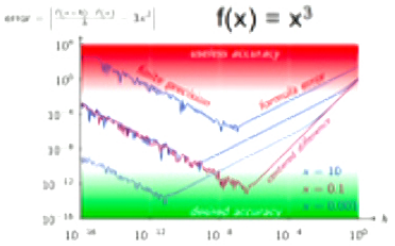


$$\overline{x_i} = \sum_{j:\text{ child of } i} \overline{x_j} \frac{\partial x_j}{\partial x_i} \qquad \text{with} \quad \overline{\mathscr{L}} = 1$$

Message passing for the adjoint at each node

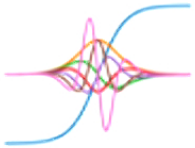
Advantages of automatic differentiation

- Accurate to the machine precision



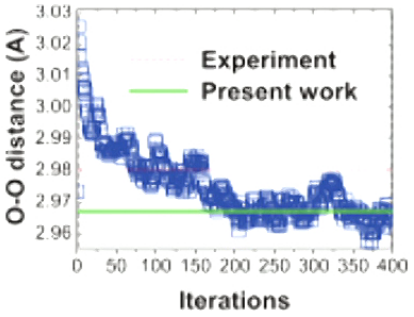
- Same computational complexity as the function evaluation: Baur-Strassen theorem '83

- Supports higher order gradients



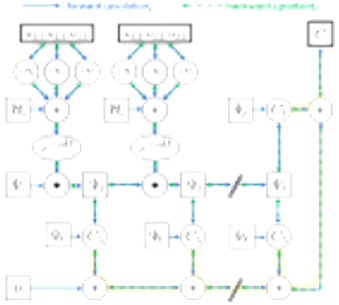
Applications of AD

Computing force



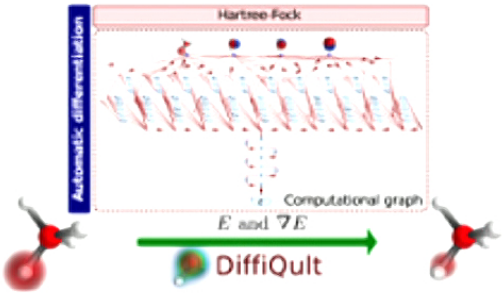
Sorella and Capriotti
J. Chem. Phys. '10

Quantum optimal control



Leung et al
PRA '17

Variational Hartree-Fock



Tamayo-Mendoza et al
ACS Cent. Sci. '18

Reverse versus forward mode

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{x}_n}\frac{\partial \boldsymbol{x}_n}{\partial \boldsymbol{x}_{n-1}}\cdots\frac{\partial \boldsymbol{x}_2}{\partial \boldsymbol{x}_1}\frac{\partial \boldsymbol{x}_1}{\partial \boldsymbol{\theta}}$$

Reverse mode AD: Vector-Jacobian Product of primitives

- Backtrace the computation graph $v_o\,(J)_{o\times i}$
- Needs to store intermediate results
- Efficient for graphs with large fan-in

Backpropagation = Reverse mode AD applied to neural networks

How to think about AD ?

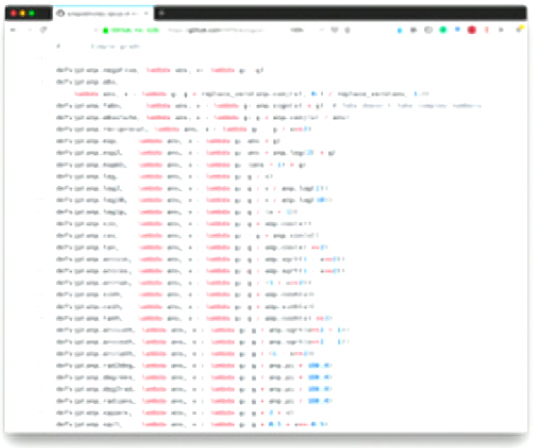
- AD is modular, and one can control its granularity
- Benefits of writing customized primitives
  - Reducing memory usage
  - Increasing numerical stability
- Call to external libraries written agnostically to AD
  (or, even a quantum processor)

PENNYLANE

Example of the primitives

~200 functions to cover most of numpy in HIPS/autograd

https://github.com/HIPS/autograd/blob/master/autograd/numpy/numpy_vjps.py



Loop/Condition/Sort/Permutations are also differentiable

Differentiable programming tools

Differentiable Scientific Programming

- Most linear algebra operations (Eigen, SVD!) are differentiable
- ODE integrators are differentiable with O(1) memory
- Differentiable ray tracer and Differentiable fluid simulations
- Differentiable Monte Carlo/Tensor Network/Functional RG/ Dynamical Mean Field Theory/Density Functional Theory/ Hartree-Fock/Coupled Cluster/Gutzwiller/ Molecular Dynamics…

Differentiable programming is more than training neural networks

Differentiable Eigensolver

$$H\Psi = \Psi\Lambda$$

Forward mode: What happen if $H \to H + dH$? Perturbation theory

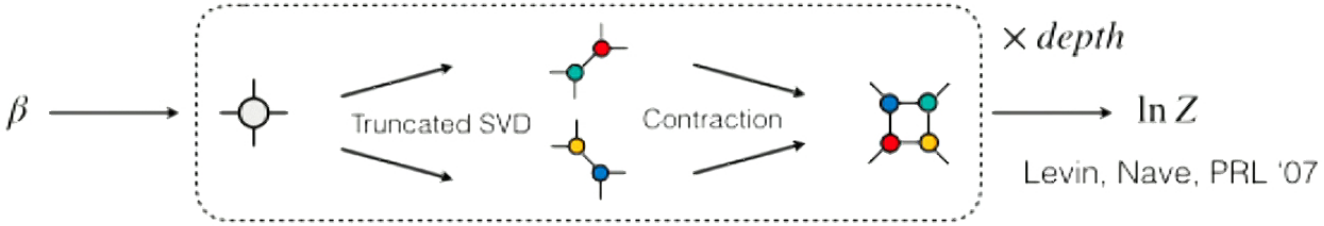
Reverse mode: How should I change H given $\partial\mathscr{L}/\partial\Psi$ and $\partial\mathscr{L}/\partial\Lambda$? **Inverse perturbation theory!**

Hamiltonian engineering via differentiable programming

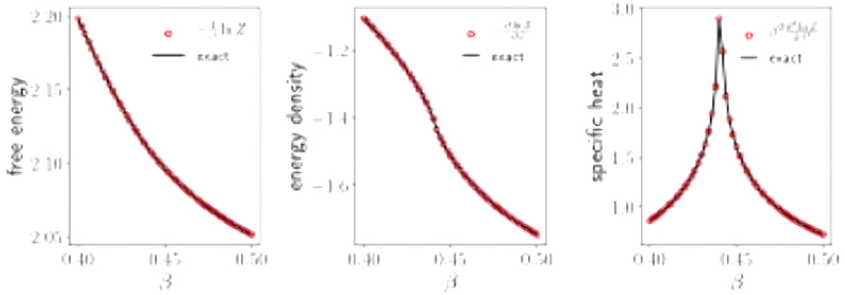
https://github.com/wangleiphy/DL4CSRC/tree/master/2-ising See also Fujita et al, PRB '18

Differentiate through TRG for Ising

Computation graph



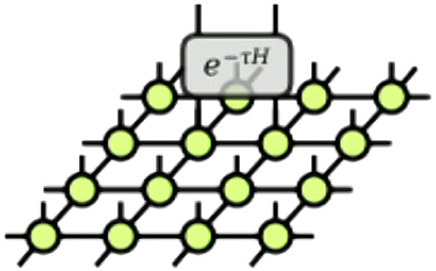
Levin, Nave, PRL '07

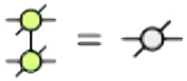


AD computes physical observables as high-order gradients

Tensor network quantum states

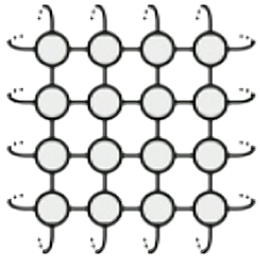
Optimization





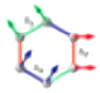
- Trotterized imaginary-time projection
- Update schemes: "simple", "full" "cluster", "faster full"…

Contraction

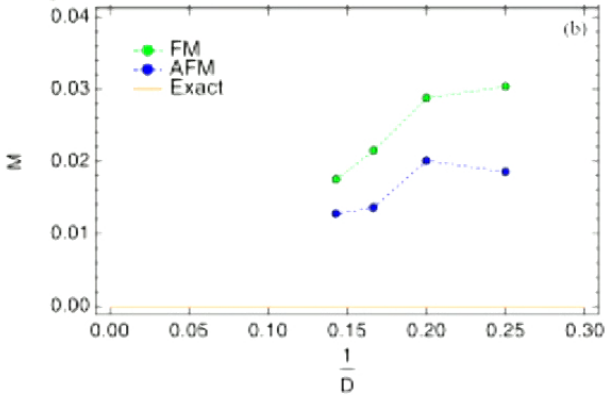


- #P hard in general
- Approximated schemes: TRG, Boundary MPS, Corner transfer matrix RG

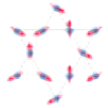
Expressibility v.s. Optimization: an eternal problem



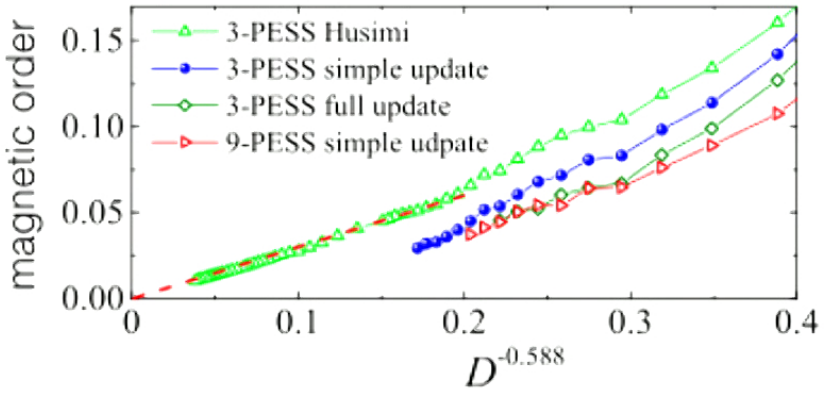
Kitaev Honeycomb model



Osorio, Corboz, Troyer, PRB '14



Kagome Heisenberg model

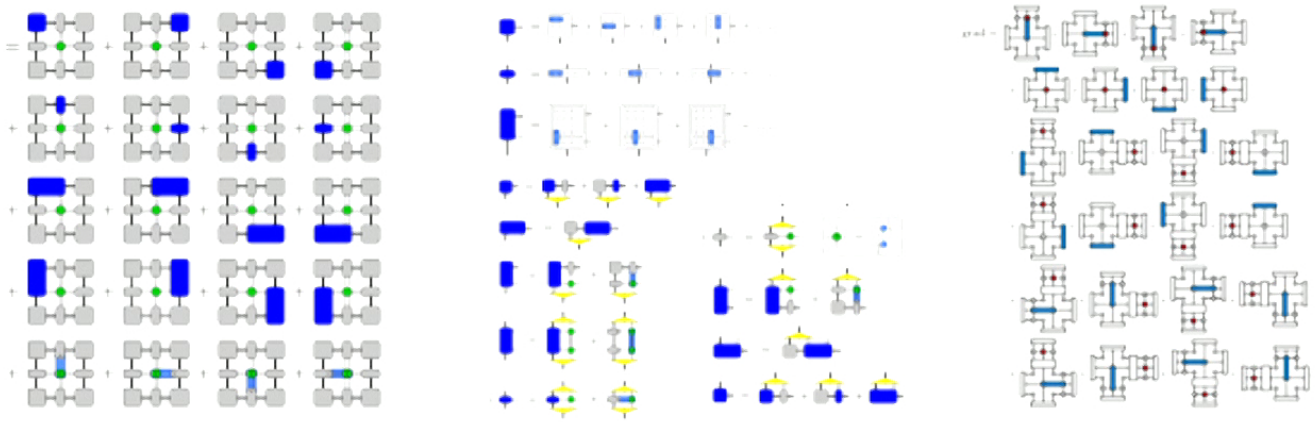


Liao et al, PRL '17

Typically, one finds ordered states at small D and tries hard to push up the bond dimensions

Variational optimization infinite tensor networks

$$\mathscr{L}_\theta = \langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle / \langle \Psi_\theta | \Psi_\theta \rangle$$



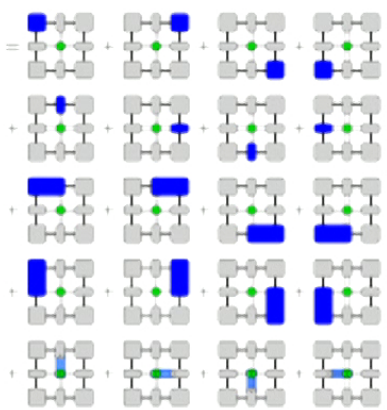
Corboz, PRB '16 Vanderstraeten et al, PRB '16

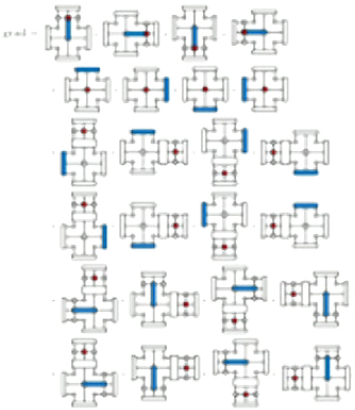
Variational optimization with gradient indeed help!
However, manually deriving gradients is cumbersome

Corboz et al, PRX '18
Rader et al, PRX '18

Variational optimization infinite tensor networks

$$\mathscr{L}_\theta = \langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle / \langle \Psi_\theta | \Psi_\theta \rangle$$





Corboz, PRB '16 Vanderstraeten et al, PRB '16

Variational optimization with gradient indeed help!
However, manually deriving gradients is cumbersome

Corboz et al, PRX '18
Rader et al, PRX '18

Nuts and Bolts

- Numerical stable backward through SVD

$$A = UDV^T \qquad \overline{A} \xleftarrow{?} \overline{U}, \overline{D}, \overline{V}$$

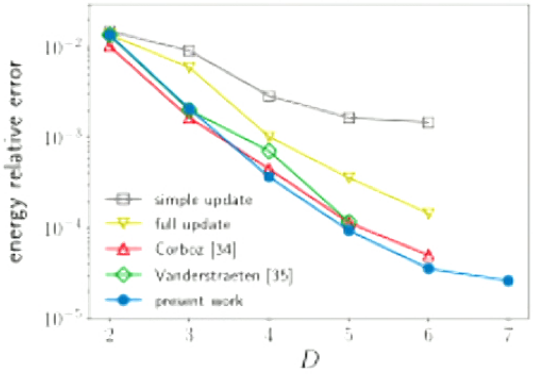
- Reduce memory via checkpointing or exploiting RG fixed point property

$$T_{i+1} = f(T_i, \theta) \xrightarrow{\text{Iterate}} T^* = f(T^*, \theta) \qquad \bar{\theta} = \overline{T^*} \left[1 - \frac{\partial f}{\partial T^*} \right]^{-1} \frac{\partial f}{\partial \theta}$$

Liao, Liu, LW, Xiang, 1903.09650

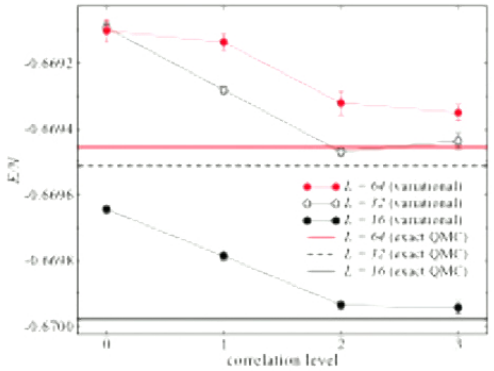
Square lattice Heisenberg model

Infinite size
AD optimized iPEPS



Liao, Liu, LW, Xiang, 1903.09650

Finite size
VMC of an RVB-type state

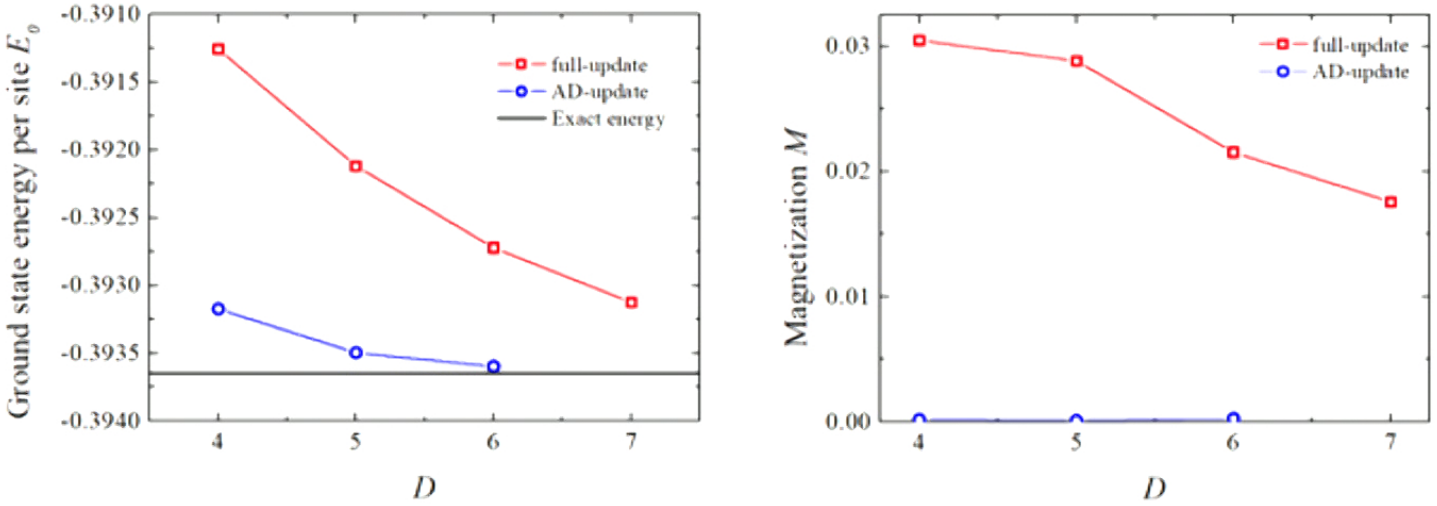


1×10^{-4} @L=64

4×10^{-5} @L=16

Lin, Tang, Sandvik, PRB '12

Kitaev honeycomb model



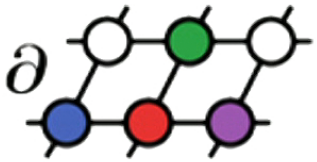
Reaches lower energy even at smaller bond dimensions with substantially reduced magnetic order

c.f. analytically constructed iPEPS, Lee et al, 1901.05786

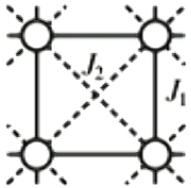
The morals

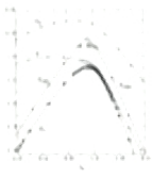
- iPEPS with small bond dimensions are more expressive than we thought. We just did not optimize them hard enough
- Differentiable programming tensor networks has a bright future: variational contraction, gauge fixing, fermions…

Google summer of code
Andreas Peter mentored by Jin-Guo Liu
https://github.com/under-Peter/TensorNetworkAD.jl

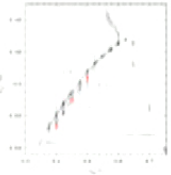


Ground state phase diagram of the J1-J2 model

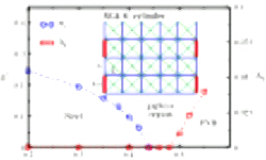




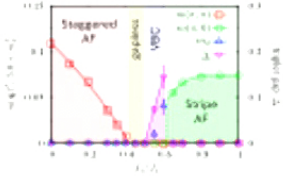
Exact Diagonalization
Diagotto et al
PRL '89



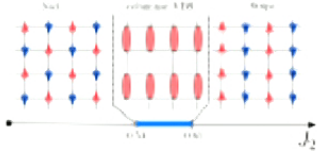
Series Expansion
Sirker et al
PRB '06



DMRG
Gong et al
PRL '14



VMC
Morita et al
JPSJ '15



iPEPS
Haghshenas et al
PRB '19

+ many many others

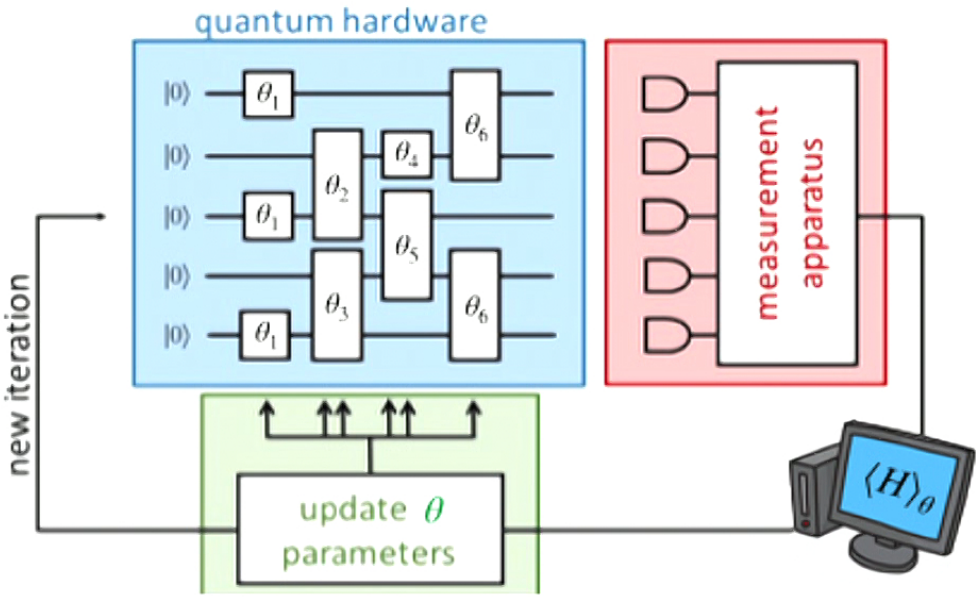
Ground state phase diagram of the J1-J2 model

Please stay tuned !

IOP, CAS
Hai-Jun Liao

J_2/J_1

Variational quantum eigensolver

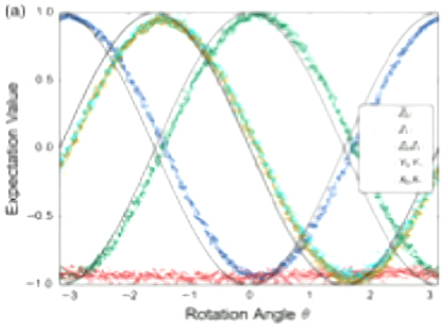


Quantum circuit as a variational ansatz

Peruzzo et al, Nat. Comm. '13

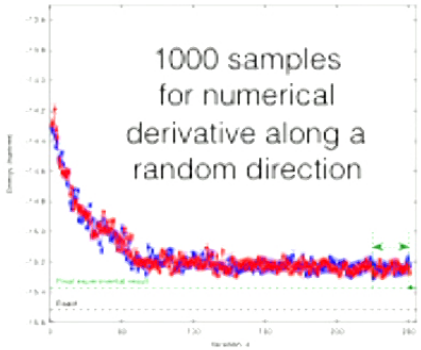
Optimize the quantum circuit

Scan 1000 values of the single variational parameter



Google PRX '16

Stochastic gradient descend with numerical derivative

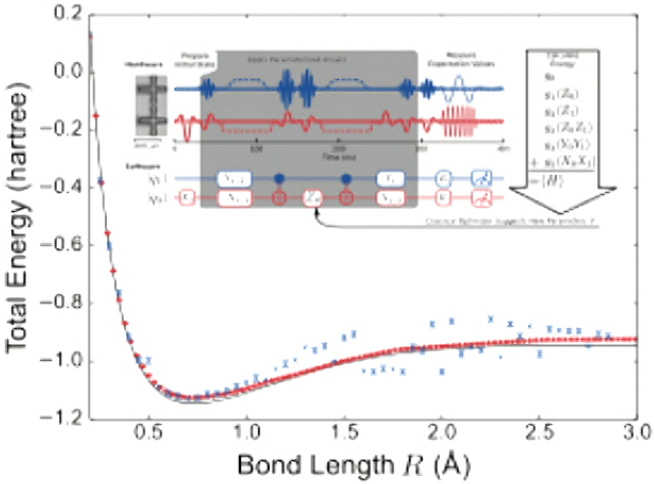


IBM Nature '17

These optimization schemes do not scale to higher dimensions

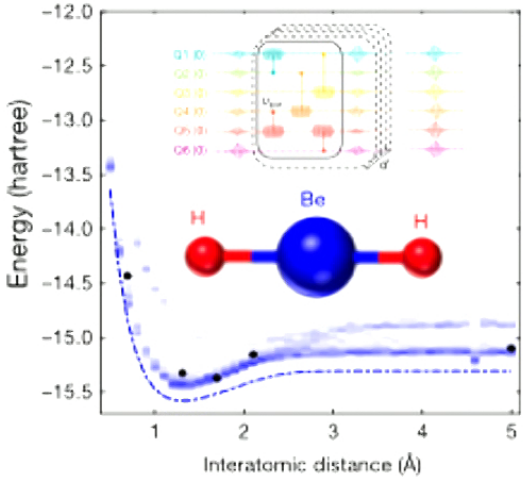
VQE on actual quantum devices

H_2 molecule with 2 qubits



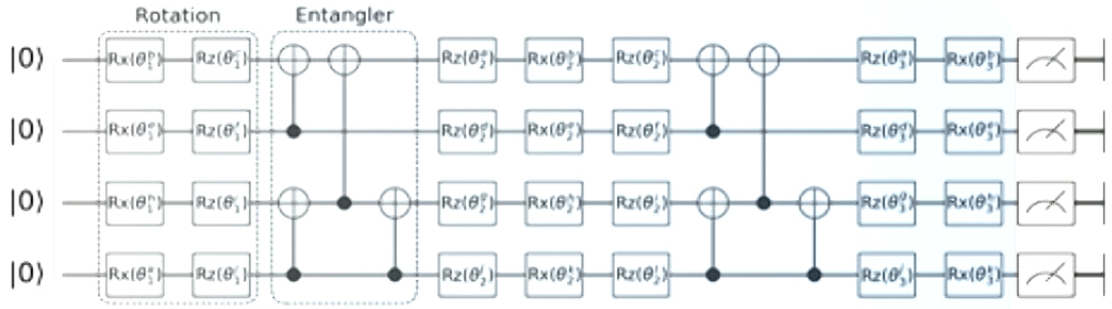
Google PRX '16

BeH_2 molecule with 6 qubits



IBM Nature '17

Differentiable quantum circuits



Parametrized gate of the form

$$e^{-\frac{i\theta}{2}\Sigma} \quad \text{with} \quad \Sigma^2 = 1$$

eg, X, Y, Z, CNOT, SWAP…

Li et al, PRL '17, Mitarai et al, PRA '18
Schuld et al, PRA '19, Nakanishi et al '19

$$\nabla \langle H \rangle_\theta = \left(\langle H \rangle_{\theta+\pi/2} - \langle H \rangle_{\theta-\pi/2} \right) / 2$$

Unbiased gradient estimator measured on quantum circuits

Monte Carlo Gradient Estimation in Machine Learning

Shakir Mohamed[*] SHAKIR@GOOGLE.COM
Mihaela Rosca[*] MIHAELACR@GOOGLE.COM
Michael Figurnov[*] MFIGURNOV@GOOGLE.COM
Andriy Mnih[*] AMNIH@GOOGLE.COM
[*]*Equal contributions, DeepMind, London*

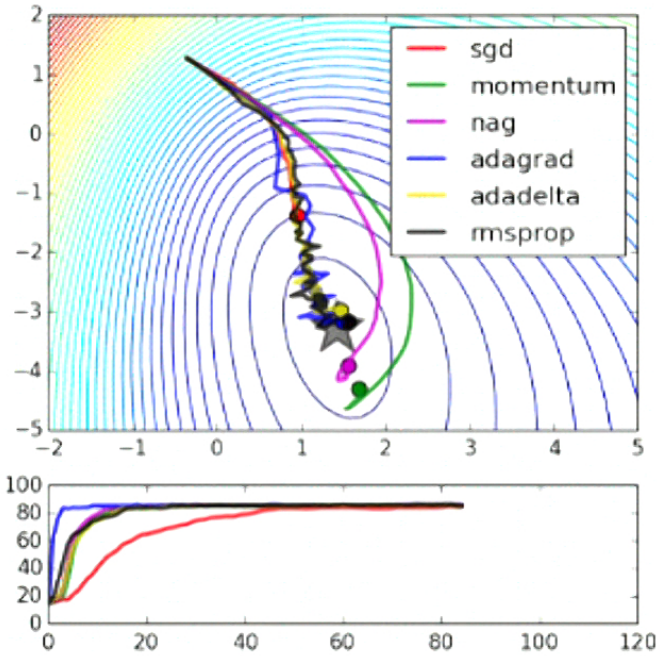
59 pages survey, three types of gradient estimators 1906.10652

10.1 Guidance in Choosing Gradient Estimators $\nabla_{\theta}\mathbb{E}_{x\sim p_{\theta}}\left[f(x)\right]$

With so many competing approaches, we offer our rules of thumb in choosing an estimator, which follow the intuition we developed throughout the paper:

- If our estimation problem involves continuous functions and measures that are continuous in the domain, then using the pathwise estimator is a good default. It is relatively easy to implement and a default implementation, one without other variance reduction, will typically have variance that is low enough so as not to interfere with the optimisation.
- If the cost function is not differentiable or a black-box function then the score-function or the measure-valued gradients are available. If the number of parameters is low, then the measure-valued gradient will typically have lower variance and would be preferred. But if we have a high-dimensional parameter set, then the score function estimator should be used.
- If we have no control over the number of times we can evaluate a black-box cost function, effectively only allowing a single evaluation of it, then the score function is the only estimator

Optimization with noisy gradient

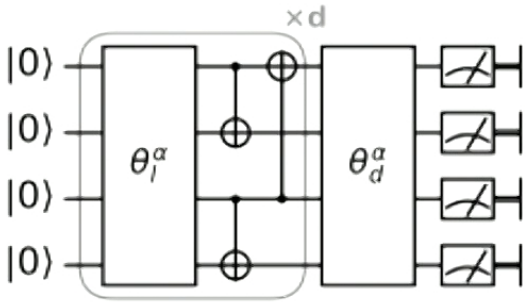


Ruder, 1609.04747

VQE encounters the "same type" of stochastic optimization in deep learning

Quantum Circuit Born Machine

With Liu, Zeng, Wu, Hu
PRA '18, PRA '19



measure

outcomes

two-sample test

classical data

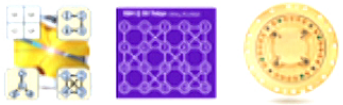
loss & gradient

$\Delta\theta$

classical optimizer

Experiments:

1801.07686
1812.08862
1811.09905
1901.08047
1904.02214



Train quantum circuits as probabilistic generative models with implicit density

Strong expressibility due to quantum sampling complexity

Differentiable Quantum Programming

It is a paradigm beyond quantum-classical hybrid

- Variational quantum eigensovler (VQE)
- Quantum circuit Born machine (QCBM)
- Quantum approximate optimization algorithm (QAOA)
- Quantum pattern recognition

...

Quantum circuit classifier	Farhi, Neven, 1802.06002 Havlicek et al, 1804.11326
TNS inspired circuit architecture	Huggins, Patel, Whaley, Stoudenmire, 1803.11537
VQE with fewer qubits	Liu, Zhang, Wan, LW, 1902.02663
Quantum generative model	Gao, Zhang, Duan, 1711.02038
Quantum adversarial training	Dallaire-Demers, Lloyd, Benedetti 1804.08641,1804.09139, 1806.00463

Differentiable Quantum Programming

It is a paradigm beyond quantum-classical hybrid

- Variational quantum eigensovler (VQE)
- Quantum circuit Born machine (QCBM)
- Quantum approximate optimization algorithm (QAOA)
- Quantum pattern recognition

Quantum code

...

Quantum circuit classifier	Farhi, Neven, 1802.06002 Havlicek et al, 1804.11326
TNS inspired circuit architecture	Huggins, Patel, Whaley, Stoudenmire, 1803.11537
VQE with fewer qubits	Liu, Zhang, Wan, LW, 1902.02663
Quantum generative model	Gao, Zhang, Duan, 1711.02038
Quantum adversarial training	Dallaire-Demers, Lloyd, Benedetti 1804.08641,1804.09139, 1806.00463

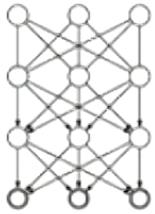
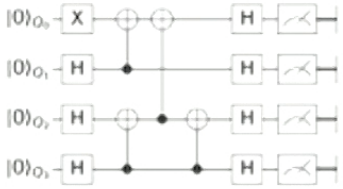
Thank You!

Differentiable Programming Tensor Networks

Hai-Jun Liao, Jin-Guo Liu, LW, Tao Xiang, 1903.09650, PRX in press

Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design

Xiu-Zhe Luo, Jin-Guo Liu, Pan Zhang, LW, up coming

∂ [ Neural Networks  Tensor Networks  Quantum Circuits]