AI and TN - a love affair

Bram Vanhecke

University of Vienna

28 Nov 2023

Collaborators





Overview

[1] https://arxiv.org/abs/2208.08713
[2] https://arxiv.org/abs/2311.11894

Overview

- Using MPS for active inference planning[1]
    - What is Active inference
    - How MPS can help

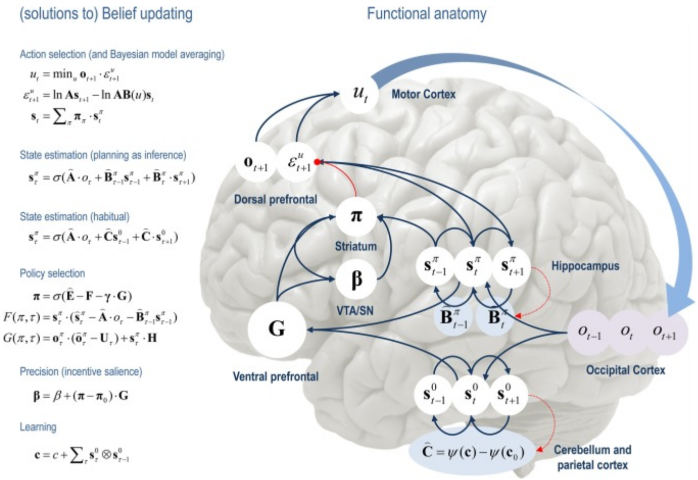
[1] https://arxiv.org/abs/2208.08713
[2] https://arxiv.org/abs/2311.11894

Overview

- Using MPS for active inference planning[1]
  - What is Active inference
  - How MPS can help
- AD for PEPS optimization[2]
  - What is PEPS
  - How AD can help

[1] https://arxiv.org/abs/2208.08713
[2] https://arxiv.org/abs/2311.11894

Active Inference



3

[3] https://doi.org/10.1016/j.neubiorev.2016.06.022

Active Inference

Active Inference

- Models behavior of an agent

Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world

Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world
- Agent plans to minimize 'surprise'.

Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world
- Agent plans to minimize 'surprise'.
- Agent has an internal model of the world, with 'hidden states' (s)

Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world
- Agent plans to minimize 'surprise'.
- Agent has an internal model of the world, with 'hidden states' (s)
  - may be imperfect

Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world
- Agent plans to minimize 'surprise'.
- Agent has an internal model of the world, with 'hidden states' (s)
  - may be imperfect
  - should in principle be updated as time goes by

Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world
- Agent plans to minimize 'surprise'.
- Agent has an internal model of the world, with 'hidden states' (s)
    - may be imperfect
    - should in principle be updated as time goes by
    - where ML comes in

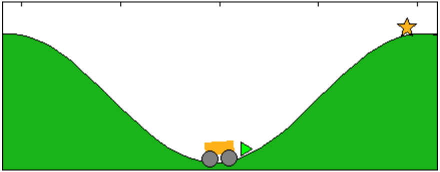
Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world
- Agent plans to minimize 'surprise'.
- Agent has an internal model of the world, with 'hidden states' (s)
  - may be imperfect
  - should in principle be updated as time goes by
  - where ML comes in
  - usually represented by a partially observable Markov decision process (POMDP)

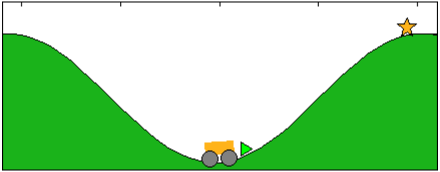
Active Inference

- Models behavior of an agent
- Agent makes observation (o) of- and actions (a) on the world
- Agent plans to minimize 'surprise'.
- Agent has an internal model of the world, with 'hidden states' (s)
  - may be imperfect
  - should in principle be updated as time goes by
  - where ML comes in
  - usually represented by a partially observable Markov decision process (POMDP)
  - the model may also be represented by a TN

Active Inference - Example - Mountain Car

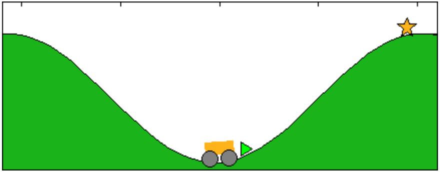


Active Inference - Example - Mountain Car



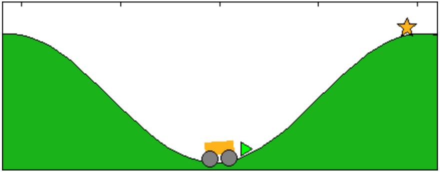
- o are observations of the location of the car

Active Inference - Example - Mountain Car



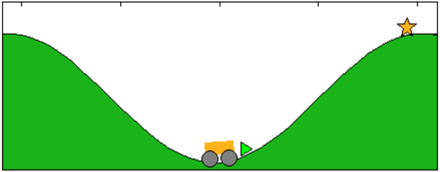
- o are observations of the location of the car
- a are push left or right actions

Active Inference - Example - Mountain Car



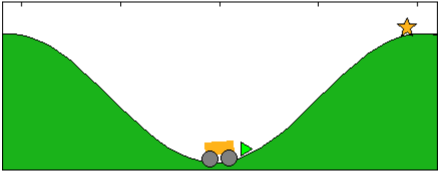
- o are observations of the location of the car
- a are push left or right actions
- The hidden state should be velocity v

Active Inference - Example - Mountain Car



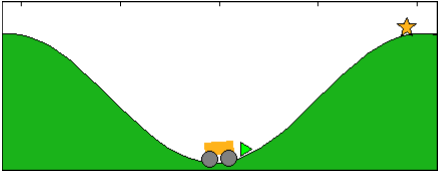
- o are observations of the location of the car
- a are push left or right actions
- The hidden state should be velocity v
- Agent is 'surprised' when the car is not at the star

Active Inference - Example - Mountain Car



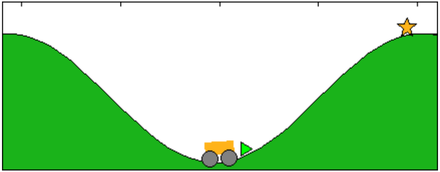
- o are observations of the location of the car
- a are push left or right actions
- The hidden state should be velocity v
- Agent is 'surprised' when the car is not at the star
  $\implies$ the preferred distribution $P(o)$ is centered around the star.

Active Inference - Example - Mountain Car



- o are observations of the location of the car
- a are push left or right actions
- The hidden state should be velocity v
- Agent is 'surprised' when the car is not at the star
  $\implies$ the preferred distribution $P(o)$ is centered around the star.
- Expected surprisal is $< -\log P >_{a_1, a_2, \ldots}$

Active Inference - Example - Mountain Car

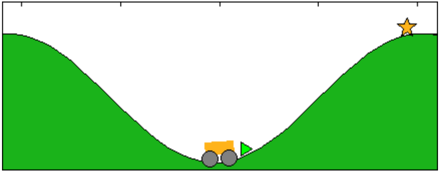


- o are observations of the location of the car
- a are push left or right actions
- The hidden state should be velocity v
- Agent is 'surprised' when the car is not at the star
  $\implies$ the preferred distribution $P(o)$ is centered around the star.
- Expected surprisal is $< -\log P >_{a_1, a_2, \ldots}$

.

- POMDP-based model: $f(o_{\text{old}}, a_{\text{old}}, v_{\text{old}}) = v_{\text{new}}$

Active Inference - Example - Mountain Car

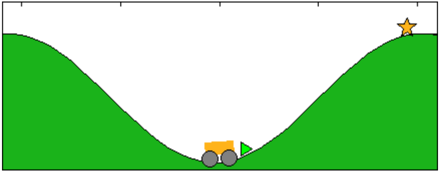


- o are observations of the location of the car
- a are push left or right actions
- The hidden state should be velocity v
- Agent is 'surprised' when the car is not at the star
  $\implies$ the preferred distribution $P(o)$ is centered around the star.
- Expected surprisal is $< -\log P >_{a_1, a_2, \ldots}$

.

- POMDP-based model: $f(o_{\text{old}}, a_{\text{old}}, v_{\text{old}}) = v_{\text{new}}$
- TN approach considers the world described by a tensor $C(o_1, a_1, o_2, a_2, \ldots)$

Active Inference - Example - Mountain Car



- o are observations of the location of the car
- a are push left or right actions
- The hidden state should be velocity v
- Agent is 'surprised' when the car is not at the star
  $\implies$ the preferred distribution $P(o)$ is centered around the star.
- Expected surprisal is $< -\log P >_{a_1, a_2, \ldots}$

.

- POMDP-based model: $f(o_{\text{old}}, a_{\text{old}}, v_{\text{old}}) = v_{\text{new}}$
- TN approach considers the world described by a tensor $C(o_1, a_1, o_2, a_2, \ldots)$

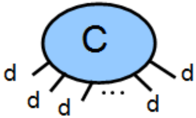
TN preeliminaries

TN preeliminaries

Big tensor $C_{i_1,..i_N}$

Arbitrary tensor
($|i_k\rangle$ – local $\mathcal{H}$ space of dim d):

$$|\Psi\rangle = \sum_{i_1,\dots,i_N} C_{i_1,i_2,\dots,i_N} |i_1 i_2 \dots i_N\rangle$$



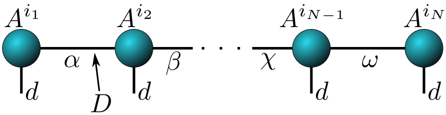
d^N parameters $C_{i_1,\dots,i_N}$

exp(N)

Network of tensors

Matrix product state:

$$\sum_{i_1,\dots,i_N} \sum_{\{\alpha\beta\dots\omega\}} A^{i_1}_{\alpha} A^{i_2}_{\alpha\beta} \dots A^{i_{N-1}}_{\chi\omega} A^{i_N}_{\omega} |i_1 i_2 \dots i_N\rangle$$

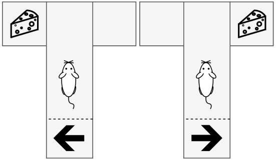


$\mathcal{O}(ND^2d)$ par. (D – dim of bond index)

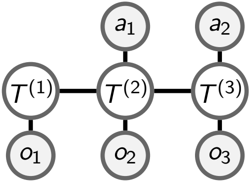
poly(N)

T-maze

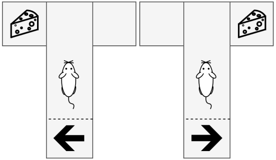
T-maze

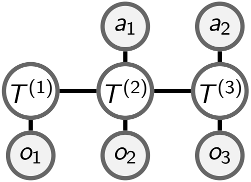


$$\Psi(\tilde{a}, \tilde{o}) =$$



, (1)

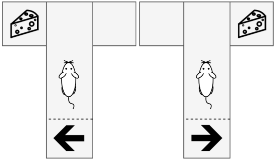


$$\Psi(\tilde{a}, \tilde{o}) =$$ 

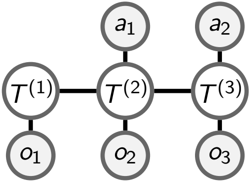
, (1)

- Generate data set of consistent actions and observations

T-maze



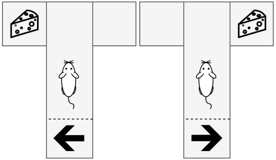
$$\Psi(\tilde{a}, \tilde{o}) =$$



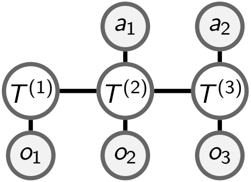
, (1)

- Generate data set of consistent actions and observations
- Maximize the overlap of the MPS with this data set.

T-maze



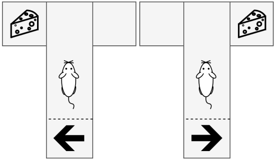
$$\Psi(\tilde{a}, \tilde{o}) = $$



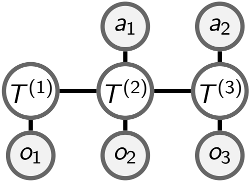
, (1)

- Generate data set of consistent actions and observations
- Maximize the overlap of the MPS with this data set.
- Through two-site updates we can 'learn' the bond dimension dynamically.

T-maze



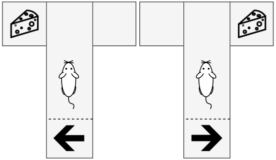
$$\Psi(\tilde{a}, \tilde{o}) =$$

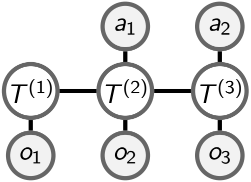


, (1)

- Generate data set of consistent actions and observations
- Maximize the overlap of the MPS with this data set.
- Through two-site updates we can 'learn' the bond dimension dynamically.
- Well understood techniques from TN world

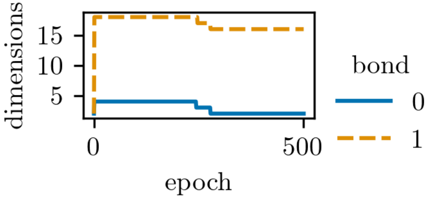
T-maze



$\Psi(\tilde{a}, \tilde{o}) =$ 

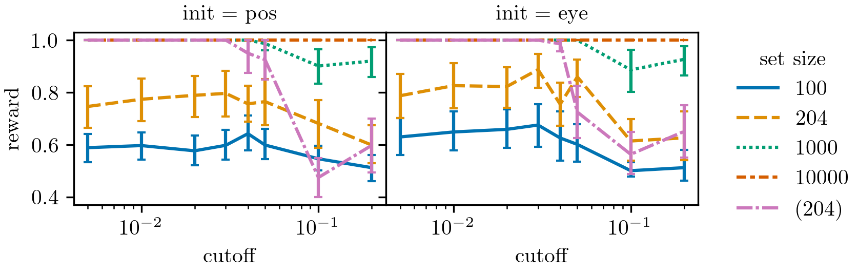
, (1)

- Generate data set of consistent actions and observations
- Maximize the overlap of the MPS with this data set.
- Through two-site updates we can 'learn' the bond dimension dynamically.
- Well understood techniques from TN world



T-maze - Results

T-maze - Results



Outlook

Outlook

- More complicated models

Outlook

- More complicated models
- Infinite horizon

Outlook

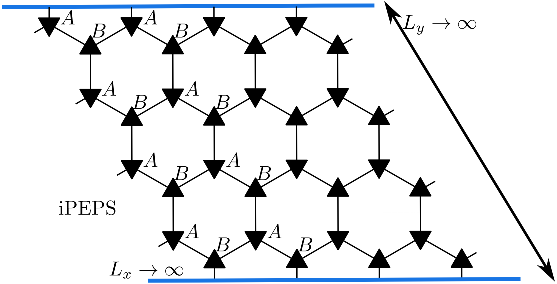
- More complicated models
- Infinite horizon
- Continuous variables

Outlook

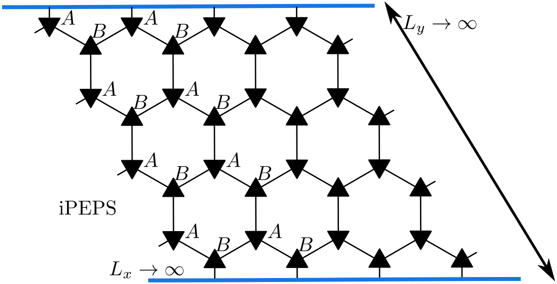
- More complicated models
- Infinite horizon
- Continuous variables
- Planning with TN

PEPS

PEPS

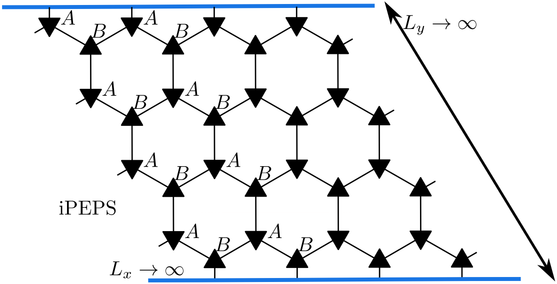


PEPS



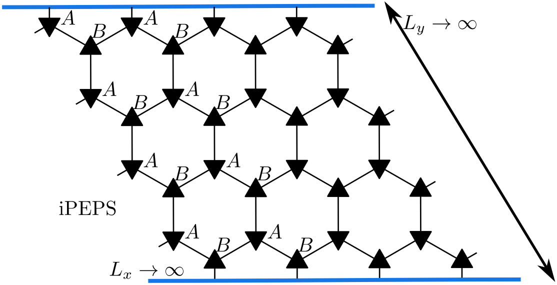
- Natural generalization of MPS to 2D

PEPS



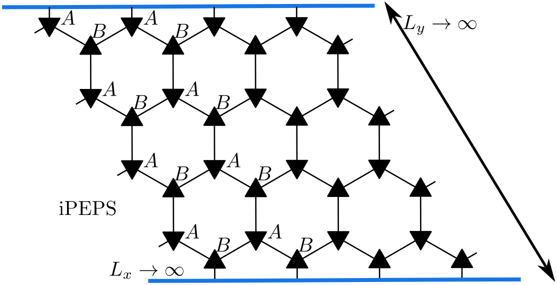
- Natural generalization of MPS to 2D
- Allows simulations in the thermodynamic limit

PEPS



- Natural generalization of MPS to 2D
- Allows simulations in the thermodynamic limit
- Captures strong correlations and exotic behavior

PEPS

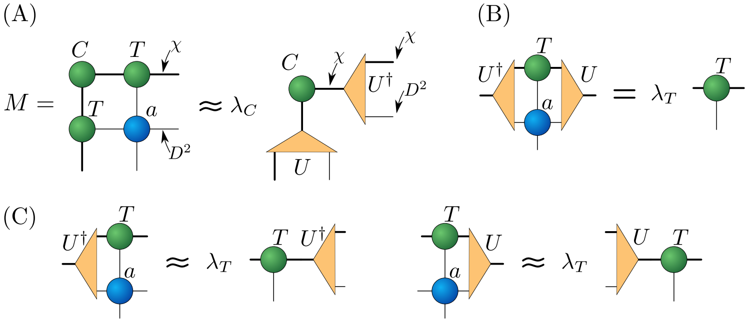


- Natural generalization of MPS to 2D
- Allows simulations in the thermodynamic limit
- Captures strong correlations and exotic behavior

Gapped $\mathbb{Z}_2$ vs gapless $U(1)$ SL in S=1/2 Kagome AF? H.J.Liao et al. PRL 118 (2017)

Contracting PEPS: Corner Transfer Matrix[4]

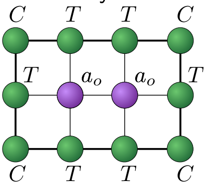
Contracting PEPS: Corner Transfer Matrix[4]



- Enables calculation with infinite TN
- Random initialization and iterated to convergence

Energy calculated approximately with CTM:

$$E \approx \tilde{E} = F(C, T, A, H) =$$



Energy gradient

Energy gradient

- Numerical gradients:

Energy gradient

- Numerical gradients:
  - Finite difference: $\frac{\partial F}{\partial A_i} = \frac{F(A+\delta * a_i) - F(A)}{\delta} + \mathcal{O}(\delta)$
    expensive, erroneous
  - Summation of terms with hole fixed and different Hamiltonian locations:
    approximate, expensive for 'larger' Hamiltonians
    P. Corboz, PRB 94, 035133 (2016),
    L. Vanderstraeten, J. Haegeman, P. Corboz, and F. Verstraete, PRB 94, 155123 (2016)
  - Summation of terms with Hamiltonian fixed and different hole locations:
    almost like AD, but ignoring isometry contribution, memory expensive
    S. P. G. Crone and P. Corboz, PRB 101, 115143 (2020)

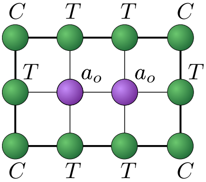
Energy gradient

- Numerical gradients:
  - Finite difference: $\frac{\partial F}{\partial A_i} = \frac{F(A+\delta * a_i) - F(A)}{\delta} + \mathcal{O}(\delta)$
    expensive, erroneous
  - Summation of terms with hole fixed and different Hamiltonian locations:
    approximate, expensive for 'larger' Hamiltonians
    P. Corboz, PRB 94, 035133 (2016),
    L. Vanderstraeten, J. Haegeman, P. Corboz, and F. Verstraete, PRB 94, 155123 (2016)
  - Summation of terms with Hamiltonian fixed and different hole locations:
    almost like AD, but ignoring isometry contribution, memory expensive
    S. P. G. Crone and P. Corboz, PRB 101, 115143 (2020)
- Analytical gradients: Automatic/Algorithmic differentiation
  memory expensive, can be problematic if treated as black box
  H-J. Liao, J-G. Liu, L. Wang, and T, Xiang, PRX 9, 031041 (2019)
  J. Hasik, D. Poilblanc, F. Becca, SciPost Phys. 10, 012 (2021)

Energy gradient

Energy calculated approximately with CTM:

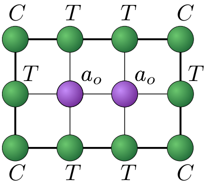
$$E \approx \tilde{E} = F(C_n, T_n, A, H) =$$



Energy gradient

Energy calculated approximately with CTM:

$$E \approx \tilde{E} = F(C_n, T_n, A, H) =$$

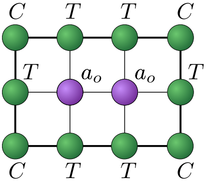


With $(C_k, T_k) \equiv x_k = f(x_{k-1}, A)$

Energy gradient

Energy calculated approximately with CTM:

$$E \approx \tilde{E} = F(C_n, T_n, A, H) =$$



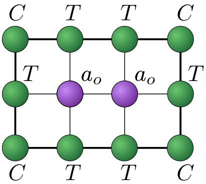
With $(C_k, T_k) \equiv x_k = f(x_{k-1}, A)$ its gradient is:

$$\begin{aligned} d\tilde{E} &= \frac{\partial F}{\partial A} dA + \frac{\partial F}{\partial x_n} dx_n, \\ dx_n &= \frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-1}} \left(\frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-2}} (...)) \right), \end{aligned}$$

Energy gradient

Energy calculated approximately with CTM:

$$E \approx \tilde{E} = F(C_n, T_n, A, H) =$$



With $(C_k, T_k) \equiv x_k = f(x_{k-1}, A)$ its gradient is:

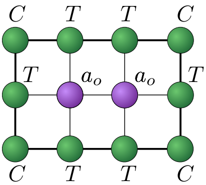
$$\begin{aligned} d\tilde{E} &= \frac{\partial F}{\partial A} dA + \frac{\partial F}{\partial x_n} dx_n, \\ dx_n &= \frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-1}} \left(\frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-2}} (...)) \right), \end{aligned}$$

Problems:

Energy gradient

Energy calculated approximately with CTM:

$$E \approx \tilde{E} = F(C_n, T_n, A, H) =$$



With $(C_k, T_k) \equiv x_k = f(x_{k-1}, A)$ its gradient is:

$$\begin{aligned} d\tilde{E} &= \frac{\partial F}{\partial A} dA + \frac{\partial F}{\partial x_n} dx_n, \\ dx_n &= \frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-1}} \left(\frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-2}} (...)) \right), \end{aligned}$$

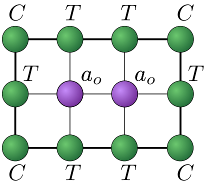
Problems:

1. costly in memory, need for many iterations

Energy gradient

Energy calculated approximately with CTM:

$$E \approx \tilde{E} = F(C_n, T_n, A, H) =$$



With $(C_k, T_k) \equiv x_k = f(x_{k-1}, A)$ its gradient is:

$$\begin{aligned} d\tilde{E} &= \frac{\partial F}{\partial A} dA + \frac{\partial F}{\partial x_n} dx_n, \\ dx_n &= \frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-1}} \left(\frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-2}} (\ldots)) \right), \end{aligned}$$

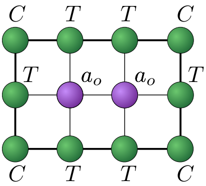
Problems:

1. costly in memory, need for many iterations
2. gradient of EIG (SVD) poorly conditioned in case of degenerate spectra

Energy gradient

Energy calculated approximately with CTM:

$$E \approx \tilde{E} = F(C_n, T_n, A, H) =$$



With $(C_k, T_k) \equiv x_k = f(x_{k-1}, A)$ its gradient is:

$$\begin{aligned} d\tilde{E} &= \frac{\partial F}{\partial A} dA + \frac{\partial F}{\partial x_n} dx_n, \\ dx_n &= \frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-1}} \left(\frac{\partial f}{\partial A} dA + \frac{\partial f}{\partial x_{n-2}} (\ldots)) \right), \end{aligned}$$

Problems:

1. costly in memory, need for many iterations
2. gradient of EIG (SVD) poorly conditioned in case of degenerate spectra
3. currently only approximate!

Problem: Current derivative of EIG(SVD) is only approximate

Problem: Current derivative of EIG(SVD) is only approximate

Solution: Calculate exact

$$M = PCP^{\dagger}$$

Problem: Current derivative of EIG(SVD) is only approximate

Solution: Calculate exact

$$M = PCP^\dagger + \underline{P_\perp C_\perp P_\perp^\dagger}$$

Problem: Current derivative of EIG(SVD) is only approximate

Solution: Calculate exact

$$M = PCP^\dagger + \underline{P_\perp C_\perp P_\perp^\dagger}$$

With $PP^\dagger + P_\perp P_\perp^\dagger = \mathbb{I}$, leading to Sylvester equation for dP:

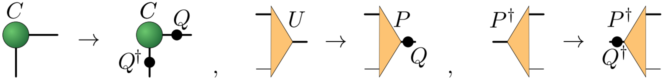
$$(\mathbb{I} - PP^\dagger)dMP = dPC - \underline{(\mathbb{I} - PP^\dagger)MdP}$$

Problem: divergencies in the gradient of EIG(SVD)

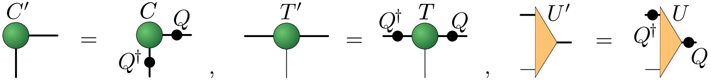
Problem: divergencies in the gradient of EIG(SVD)

Solution: Q-deformed CTM with $Q = \mathbb{I}$

(A) Q-deformation step



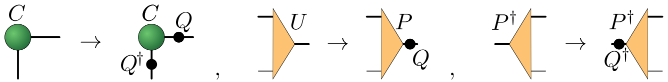
(B) Gauge transformation



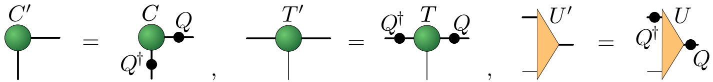
Problem: divergencies in the gradient of EIG(SVD)

Solution: Q-deformed CTM with $Q = \mathbb{I}$

(A) Q-deformation step



(B) Gauge transformation

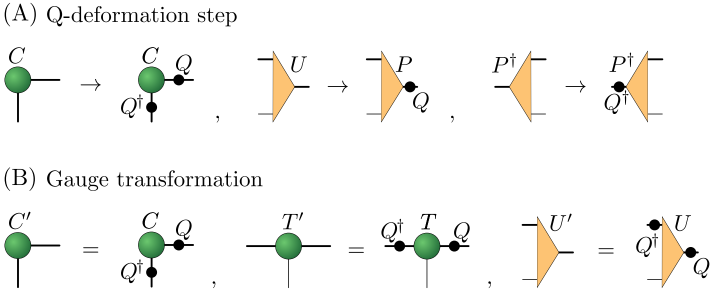


regular CTM:

$$\begin{aligned} dC &= \mathbb{I} \circ (P^\dagger dMP) \\ P^\dagger dP &= F \circ (P^\dagger dMP), \quad F_{ij} = 1/(c_j - c_i) \end{aligned}$$

Problem: divergencies in the gradient of EIG(SVD)

Solution: Q-deformed CTM with $Q = \mathbb{I}$



regular CTM:

$$
\begin{aligned}
dC &= \mathbb{I} \circ (P^\dagger dMP) \\
P^\dagger dP &= F \circ (P^\dagger dMP), \quad F_{ij} = 1/(c_j - c_i)
\end{aligned}
$$

$\rightarrow$

Q-deformed CTM:

$$
\begin{aligned}
dC &= P^\dagger dMP \\
P^\dagger dP &= 0
\end{aligned}
$$

Problem: memory intensive, many iterations

Problem: memory intensive, many iterations

Solution: fixed point differentiation

Problem: memory intensive, many iterations

Solution: fixed point differentiation

if $x = f(A, x)$:

Problem: memory intensive, many iterations

Solution: fixed point differentiation

if $x = f(A, x)$:

$$dx = \sum_{k=0}^{\infty} \left(\frac{\partial f}{\partial x} \right)^k \frac{\partial f}{\partial A} dA = \left(1 - \frac{\partial f}{\partial x} \right)^{-1} \frac{\partial f}{\partial A} dA.$$

Problem: memory intensive, many iterations

Solution: fixed point differentiation

if $x = f(A, x)$:

$$dx = \sum_{k=0}^{\infty} \left(\frac{\partial f}{\partial x} \right)^k \frac{\partial f}{\partial A} dA = \left(1 - \frac{\partial f}{\partial x} \right)^{-1} \frac{\partial f}{\partial A} dA.$$

But: requires element-wise convergence $x = f(x, A)$

Problem: memory intensive, many iterations

Solution: fixed point differentiation

if $x = f(A, x)$:

$$dx = \sum_{k=0}^{\infty} \left(\frac{\partial f}{\partial x}\right)^k \frac{\partial f}{\partial A} dA = \left(1 - \frac{\partial f}{\partial x}\right)^{-1} \frac{\partial f}{\partial A} dA.$$

But: requires element-wise convergence $x = f(x, A)$

EIG(SVD) is not unique, so gauge fixing is required:

Problem: memory intensive, many iterations

Solution: fixed point differentiation

if $x = f(A, x)$:

$$dx = \sum_{k=0}^{\infty} \left(\frac{\partial f}{\partial x}\right)^k \frac{\partial f}{\partial A} dA = \left(1 - \frac{\partial f}{\partial x}\right)^{-1} \frac{\partial f}{\partial A} dA.$$

But: requires element-wise convergence $x = f(x, A)$

EIG(SVD) is not unique, so gauge fixing is required:

$$UCU^\dagger = \big(U\sigma\big) C \big(\sigma^\dagger U^\dagger\big) \;\Rightarrow\; \hat{T} \xrightarrow{\text{f}} \sigma^\dagger \hat{T} \sigma = T.$$

Comparison of gradients

Comparison of gradients

State: $|\Psi\rangle = |\varphi_{NN}^{\text{RVB}}\rangle + |\varphi_{\text{long}}^{\text{RVB}}\rangle + \beta|\varphi_3\rangle$, with $D = 3, \chi = 160$

Hamiltonian: $H = J_1 \sum_{i,j \in NN,\alpha} f(\alpha) S_i^\alpha S_j^\alpha + J_2 \sum_{i,j \in NNN} \overrightarrow{S}_i \cdot \overrightarrow{S}_j$

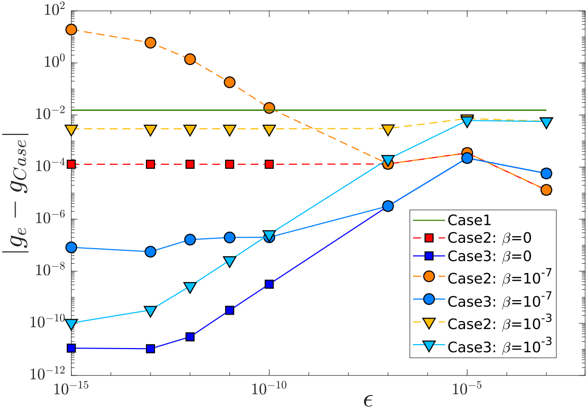
with $SU(2)$ symmetry breaking anisotropy $f([x, y, z]) = [-1, 1 + \beta, -1 + \beta]$

Comparison of gradients

State: $|\Psi\rangle = |\varphi_{NN}^{\text{RVB}}\rangle + |\varphi_{\text{long}}^{\text{RVB}}\rangle + \beta|\varphi_3\rangle$, with $D = 3, \chi = 160$

Hamiltonian: $H = J_1 \sum_{i,j \in NN, \alpha} f(\alpha) S_i^\alpha S_j^\alpha + J_2 \sum_{i,j \in NNN} \vec{S}_i \cdot \vec{S}_j$

with $SU(2)$ symmetry breaking anisotropy $f([x,y,z]) = [-1, 1+\beta, -1+\beta]$



$F_{ij} \to \frac{c_j - c_i}{(c_j - c_i)^2 + \epsilon}$

- our gradient g_e
- Case 1: $dP = 0$
- Case 2: current AD
- Case 3: using Sylvester equation for dP

Conclusions

Conclusions

We significantly improved the AD for PEPS optimization problems:

- stability
- accuracy

Conclusions

We significantly improved the AD for PEPS optimization problems:

- stability
- accuracy

PEPS libraries cannot yet be treated as blackbox like with DMRG

Conclusions

We significantly improved the AD for PEPS optimization problems:

- stability
- accuracy

PEPS libraries cannot yet be treated as blackbox like with DMRG

AD is a great tool if used carefully:

- reduces workcost
- eliminates bugs
- allows for more efficient algorithms

Conclusions

We significantly improved the AD for PEPS optimization problems:

- stability
- accuracy

PEPS libraries cannot yet be treated as blackbox like with DMRG

AD is a great tool if used carefully:

- reduces workcost
- eliminates bugs
- allows for more efficient algorithms
- may allow to reach higher bond dimensions D, χ and hence higher correlation lengths ξ, tackle challenging problems with bigger accuracy

Thank You