CS 598 EVS: Tensor Computations

Tensor Networks

Edgar Solomonik

University of Illinois at Urbana-Champaign

Tensor Networks

- Tensor network methods seek to approximately compute quantities involving high-order tensors, by making use of their decompositions
  - *Given one or more tensors decomposed via CP, Tucker, tensor train, or another decomposition, may want to*
    - *compute eigenvalues of a matricization of a tensor*
    - *solve a linear system or least squares problem where the right-hand side is an unfolding of a decomposted tensor and the matrix is a matricization of another tensor*
    - *apply transformations to 'evolve' decomposed tensor to another tensor of interest*
- Tensor network methods are often employed in problems where the tensor order varies with problem size
  - *Given n variables that take on binary values, all combinations of variable states can be enumerated in an order n tensor, but values of all 2^n entries cannot be stored at once*
  - *Different tensor decompositions (tensor networks) aim to represent such tensors with $O(\text{poly}(n))$ degrees of freedom*
  - *Such problems are fundamental in the study of quantum systems*

Tensor Representation of Quantum States

- An n-qubit state can be represented by an order n *amplitude tensor* $\mathcal{T} \in \mathbb{C}^{2\times\cdots\times 2}$
  - *The state is typically written as*

$$|\psi\rangle = \sum_{(i_1,\ldots,i_n)\in\{0,1\}^n} t_{i_1\ldots i_n}|i_1\ldots i_n\rangle$$

  - *The value $t^*_{i_1\ldots i_n} t_{i_1\ldots i_n}$ is the probability that a classical observation of the qubit states $|\psi\rangle$ yields classical bit values $(i_1,\ldots,i_n)$*
  - *If $\mathcal{T}$ can be written as a product of rank one factors $\mathcal{T} = \bigotimes_{i=1}^{n} \boldsymbol{u}^{(i)}$, it is referred to as a product state and does not contain nonclassical information (quantum entanglement)*
  - *One of the most common decompositions used to approximate states with little entanglement is the tensor train decomposition, which in this context is known as the matrix product state*

Hamiltonians

- Quantum states and their evolution are described by *Hamiltonians*
  - *A Hamiltonian is a Hermitian linear operator H that acts on a state to product a new one $|\phi\rangle = H|\psi\rangle$*
  - *For an n-qubit system, Hamiltonians are typically described as a sum of local operators $H = \sum_i H_i + \sum_{i,j} G_i F_j$ where H_i, G_i, F_i are operators acting on the ith qubit, described by matrices $\boldsymbol{H}_i, \boldsymbol{G}_i, \boldsymbol{F}_i \in \mathbb{C}^{2\times 2}$*
  - *The amplitude tensor is transformed by application $|\phi\rangle = H|\psi\rangle$ of this 'local' Hamiltonian as*

$$\boldsymbol{\mathcal{T}}^{\phi} = \sum_i \boldsymbol{\mathcal{T}}^{\psi} \times_i \boldsymbol{H}_i + \sum_{i,j} \boldsymbol{\mathcal{T}}^{\phi} \times_i \boldsymbol{G}_i \times_j \boldsymbol{F}_j$$

  - *For example, a 3-qubit Hamiltonian may correspond to a matrix of the form,*

$$\boldsymbol{H} = \boldsymbol{H}_1 \otimes \boldsymbol{I} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{H}_2 \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{H}_3 + \boldsymbol{G}_1 \otimes \boldsymbol{F}_2 \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{G}_2 \otimes \boldsymbol{F}_3$$

  - *The ground state of a Hamiltonian H is the minimizer $|\psi\rangle$ to $\langle\psi|H|\psi\rangle / \langle\psi|\psi\rangle$, which is the lowest eigenvalue (ground state energy) of H*

Time-Evolution of Quantum States

- The Schrödinger equation perscribes a time-evolution for a quantum state given a Hamiltonian
  - *The Schrödinger equation may be written as* $i\frac{d}{dt}|\psi\rangle = H|\psi\rangle$
  - *Given a state* $|\psi\rangle$*, its evolution for time* t *is given by solving the Schroedinger equation as*

$$\phi = e^{-iHt}|\psi\rangle$$

- If H is a local Hamiltonian, *Trotterization* provides a method for time-evolution
  - *Given a timestep* $\tau \ll 1$*, we will simulate the time evolution in timesteps*

$$e^{-iHt}|\psi\rangle = \prod_{j=1}^{t/\tau} e^{-iH\tau}|\psi\rangle$$

  - *We make use of the Baker-Campbell-Hausdorff formula*

$$e^{\boldsymbol{X}}e^{\boldsymbol{Y}} = e^{\boldsymbol{X}+\boldsymbol{Y}+[\boldsymbol{X},\boldsymbol{Y}]/2+[\boldsymbol{X},[\boldsymbol{X},\boldsymbol{Y}]]/12+\ldots}$$

  - *Let* $H = \sum_i H_i$*, then Trotterization gives* $e^{-iH\tau} = \prod_j e^{-iH_j\tau} + O(\tau^2)$

Time-Evolution of a Matrix Product State (MPS)

- To simulate time-evolution $|\psi(t)\rangle$ from a product state $|\psi(0)\rangle$, we can approximate each $|\psi(j\tau+\tau)\rangle = e^{-iH\tau}|\psi(j\tau)\rangle$ as a matrix product state
  - *Initial bond dimension (rank) of the MPS is 1*
  - *Consider a simple Hamiltonian $H_i = Z_i Z_{i+1}$ where Z_i acts on the ith qubit, then our goal is to be able to apply $e^{-iH_j\tau}$ to an MPS ψ to produce a new MPS ϕ*

$$|\phi\rangle = e^{-iH_j\tau}|\psi\rangle \quad \Leftrightarrow \quad t^{\phi}_{i_1\ldots i_n} = \sum_{k_j,k_{j+1}} t^{\psi}_{i_1\ldots i_{j-1}k_jk_{j+1}i_{j+2}\ldots i_n} g^{(j)}_{k_jk_{j+1}i_ji_{j+1}}$$

  where $\mathcal{G}^{(i)}$ is described by its 4×4 matrix unfolding as $e^{\boldsymbol{Z}_i\otimes\boldsymbol{Z}_{i+1}}$
  - *If $|\phi\rangle$ is an MPS, so that $\mathcal{T}^{\phi}$ is in a tensor train decomposition, application of $\mathcal{G}^{(i)}$ increases the bond dimension between factors i and $i+1$ by at most a factor of 4*
  - *Time evolution methods typically maintain an MPS with a maximal bond dimension or desired level of accuracy and truncate accordingly*
  - *Imaginary time-evolution computes $|\psi(-it)\rangle$, which gives $e^{-Ht}|\psi(0)\rangle$ and with increasing t converges to the eigenvector of H with the lowest eigenvalue*

Hamiltonians as Matrix Product Operators (MPOs)

- A *matrix product operator (MPO)* is a tensor-train decomposition where every factor tensor is assigned a pair of modes of the original tensor
  - *For example, if we wish to represent an 8×8 Hamiltonian $\boldsymbol{H}$ as an MPO, we could encode it in a decomposition of a $2 \times 2 \times 2 \times 2 \times 2 \times 2$ tensor*

$$h_{i_1 i_2 i_3 j_1 j_2 j_3} = \sum_{k_1, k_2} u^{(H)}_{i_1 j_1 k_1} v^{(H)}_{i_2 j_2 k_1 k_2} w^{(H)}_{i_3 j_3 k_2}$$

  the dimensions of the auxiliary indices k_1 and k_2 (internal legs) are the bond dimensions of an MPO
  - *More generally for an order $2d$ tensor, an MPO would be composed of d factors, 2 of which are order 3 and the rest of which are order 4*
  - *A matrix-vector product with a Hamiltonian represented as an MPO with bond dimension R and a state in MPS format with bond dimension R', $|\phi\rangle = H|\psi\rangle$ yields an MPS with bond dimension at most RR'*

$$\underbrace{\sum_{k_1=1}^{R} \sum_{k_1'=1}^{R'}}_{\sum_{K_1=1}^{RR'}} \underbrace{\sum_{k_2=1}^{R} \sum_{k_2'=1}^{R'}}_{\sum_{K_2=1}^{RR'}} \underbrace{\left(\sum_{j_1} u^{(H)}_{i_1 j_1 k_1} u^{(\psi)}_{j_1 k_1'} \right)}_{u^{(\phi)}_{i_1 K_1}} \underbrace{\left(\sum_{j_2} v^{(H)}_{i_2 j_2 k_1 k_2} v^{(\psi)}_{j_2 k_1' k_2'} \right)}_{v^{(\phi)}_{i_2 K_1 K_2}} \underbrace{\left(\sum_{j_3} w^{(H)}_{i_3 j_3 k_2} w^{(\psi)}_{j_3 k_2'} \right)}_{w^{(\phi)}_{i_3 K_3}}$$

Hamiltonians as Matrix Product Operators (MPOs)

- Hamiltonians describing quantum systems can typically be represented in an MPO format with low bond dimension
  - *Consider a Hamiltonian of the form*

$$\boldsymbol{H} = \boldsymbol{A} \otimes \boldsymbol{I} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{B} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{C}$$

  - *We can describe it by a matrix product operator with factors that have the following matrix unfoldings*

$$\boldsymbol{U}^{(H)} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{I} & \boldsymbol{I} \end{bmatrix}, \boldsymbol{V}^{(H)} = \begin{bmatrix} \boldsymbol{I} & & \\ & \boldsymbol{B} & \\ & & \boldsymbol{I} \end{bmatrix}, \boldsymbol{W}^{(H)} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{C} \end{bmatrix}$$

    with bond dimension equal to 3

  - *Further, its Trotterized time-evolution operator,*

$$e^{-i\boldsymbol{H}\tau} = \left(e^{-i\boldsymbol{A}\tau} \otimes \boldsymbol{I} \otimes \boldsymbol{I}\right)\left(\boldsymbol{I} \otimes e^{-i\boldsymbol{B}\tau} \otimes \boldsymbol{I}\right)\left(\boldsymbol{I} \otimes \boldsymbol{I} \otimes e^{-i\boldsymbol{C}\tau}\right)$$

    is an MPO with bond dimension equal to 1, since its a product of three MPOs with unit bond dimension

Generation of Efficient MPO Representations of Hamiltonians

- More sophisticated ways of embedding a local Hamiltonian into an MPO yield lower bond dimension
  - *For example, our 3-term Hamiltonian from the previous slide actually has a rank 2 tensor train decomposition (the more general d-term case remains rank 2 with similar structure)*

$$\boldsymbol{U}^{(H)} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{I} \end{bmatrix}, \boldsymbol{V}^{(H)} = \begin{bmatrix} \boldsymbol{I} & \\ \boldsymbol{B} & \boldsymbol{I} \end{bmatrix}, \boldsymbol{W}^{(H)} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{C} \end{bmatrix}$$

  - *One generic scheme is to represent each local term as an MPO, sum to obtain an MPO with larger bond dimension, then compress bond dimensions by SVDs of each pair of neighboring tensor factors*
    - *This scheme is not guaranteed to find the global minima, but is generally effective in practice [Hubig, McCulloch, and Schollwöck, 2017]*
  - *The generation of an MPO from a local Hamiltonian can also be done by expressing the latter as a complex-weighted finite-state automata [Crosswhite and Bacon, 2008]*
  - *A further complication is the consideration of presevation of group symmetries (quantum number symmetries) in the MPO factors, see e.g., [McCulloch, 2008]*

Density Matrix Renormalization Group (DMRG)

- *DMRG* [S. White, 1992] is an alternating optimization scheme to approximate by an MPS the ground state (lowest eigenvector) of a Hamiltonian described by an MPO

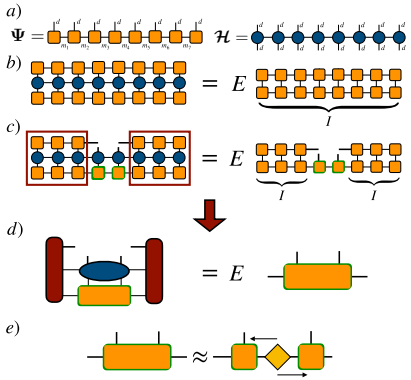


Diagram from [R. Levy, E.S., and B. Clark, 2020]

DMRG Algorithm Description

- A sweep of the DMRG algorithm updates all factor tensors in the MPS
  - *Our eigenproblem is equivalent to a minimization problem,*

$$\min_{\Psi} \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle} \Leftrightarrow \min_{\psi,\langle\psi|\psi\rangle=1} \lambda(\psi),\ \textit{s.t.}\ H|\psi\rangle = \lambda(\psi)|\psi\rangle$$

  - *Two variants possible*
    - *single-site optimizes one factor tensor at a time*
    - *two-site optimizes a pair of neighboring factor tensors at a time and adjusts the bond dimension between them*
  - *Each optimization subproblem must consider the left and right environment tensors, which are given by the contraction of everything except the tensor factors being optimized in the expression* $\langle\psi|H|\psi\rangle$
  - *In the two-site case, each subproblem is described by a left environment tensor* $\mathcal{L}$ *and right environment tensor* $\mathcal{R}$ *(see part d of Figure on last page except with center blue tensor split and absorbed into environments to the left and right) as*

$$\min_{\mathcal{U},\|\mathcal{U}\|_F=1} \sum_{i_L,j_L,i_R,j_R,a_L,b_L,a_R,b_R,k} u_{i_L j_L i_R j_R} l_{i_L j_L a_L b_L k} r_{k i_R j_R a_R b_R} u_{a_L b_L a_R b_R},$$

  which can be solved efficiently with the use of iterative solvers

Cost Analysis of DMRG

- Often the maximum bond dimension of the MPS (R) exceeds the MPO bond dimension significantly, and while each subproblem finds an eigenvector of an $O(R^2)$-by-$O(R^2)$ matrix, the optimization of each site in DMRG generally has cost $O(R^3)$
  - *Contracting an iterate $\mathcal{U}$ (of size $O(R^2)$) with the environment tensor $\mathcal{L}$ (of size $O(R^2)$) has cost $O(R^3)$ and the subsequent contraction with $\mathcal{R}$ is similarly $O(R^3)$*
  - *Sweeps alternate direction so as to preserve intermediate environment tensors*
  - *When sweeping rightwards, environment tensors for the nth site can be formed by updating the left environment tensor for $n-1$th factor tensor and using the same right environment tensor as when last updating the nth factor*
  - *Typically DMRG begins with a low MPS bond dimension and increases it by a factor of 2 after 1-4 sweeps, performing 2-4 iterations of an iterative method such as Lanczos on each subproblem*
  - *The cost of this full procedure given d factors is $O(dR^3)$*

Sources of Error in Tensor Network Calculations

- Tensor network optimization algorithms perform approximations that are local to a few tensor factors
  - *Time-evolution methods need to approximately apply two-site gates onto two neighboring tensor factors in the MPS*
    - *this can be performed by contraction followed by SVD or alternative approaches such as implicit randomized SVD, which avoids forming a large contracted intermediate tensor*
    - *in both cases, a local truncation error is incurred*
  - *DMRG has two sources of truncation error*
    - *approximate SVD of neighboring factors to reduce bond dimension*
    - *each subproblem is typically only solved approximately (Lanczos is not executed to full accuracy)*
  - *To gauge stability of these algorithms, need to bound the extent to which these local errors may be amplified to errors in the overall state/energy*

Perturbation Analysis of Tensor Networks

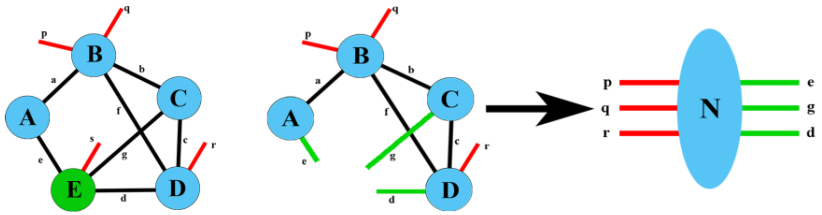
- Suppose we perturb a factor tensor of an tensor network by $\delta\mathcal{U}$, with $\|\delta\mathcal{U}\|_F/\|\mathcal{U}\|_F < \epsilon$, what is the magnitude of the error in the tensor represented by the tensor network $\delta\mathcal{T} = \mathcal{T}(\mathcal{U}) - \mathcal{T}(\mathcal{U} + \delta\mathcal{U})$?
  - *As ϵ goes to zero, the amplification of the error can be bounded by differentiating the tensor network $\mathcal{T}(\mathcal{U})$ with respect to $\mathcal{U}$*
  - *If $\mathcal{T}(\mathcal{U})$ is an order d tensor and $\mathcal{U}$ is an order p tensor, we can associate an environment tensor $\mathcal{J}^{(\mathcal{T})}(\mathcal{U})$ of order $d + p$ with the Jacobian of $\mathcal{T}(\mathcal{U})$ as well as the matricized form $\boldsymbol{J}^{(\mathcal{T})}(\mathcal{U}) \in \mathbb{R}^{size(\mathcal{T}) \times size(\mathcal{U})}$*
  - *The error can then be described to first order in ϵ (or exactly, if the factor tensor $\mathcal{U}$ appears only once in the tensor network, in which case $\mathcal{T}(\mathcal{U})$ is linear) by*

$$\delta\mathcal{T} = \boldsymbol{J}^{(\mathcal{T})}(\mathcal{U})\,\mathrm{vec}(\delta\mathcal{U})$$

  - *We can then bound the magnitude of the absolute and relative errors as*

$$\|\delta\mathcal{T}\|_2 \leqslant \|\boldsymbol{J}^{(\mathcal{T})}(\mathcal{U})\|_2\epsilon, \quad \frac{\|\delta\mathcal{T}\|_2}{\|\mathcal{T}(\mathcal{U})\|_2} \leqslant \kappa(\boldsymbol{J}^{(\mathcal{T})}(\mathcal{U}))\epsilon$$

Environment Tensor and Jacobian Matrix



(a) Tensor Network (b) Environment Matrix

Diagram from [Y. Zhang and E.S., 2020]

Canonical Forms

- A tensor network is in a *canonical form* w.r.t. factor tensor $\mathcal{U}$ if the columns of $\boldsymbol{J}^{(\mathcal{T})}(\mathcal{U})$ are orthonormal, i.e., $\kappa(\boldsymbol{J}^{(\mathcal{T})}(\mathcal{U})) = 1$
  - *A canonical form insures that any relative perturbation to $\mathcal{U}$ is not amplified when considering $\mathcal{T}(\mathcal{U})$*
  - *Within algorithms such as time evolution and DMRG, stability is ensured if local approximations happen when the MPS is in an appropriate canonical form*
  - *An MPS is easy to put into a canonical form as in this case $\mathcal{T}(\mathcal{U})$ is linear in $\mathcal{U}$, so the Jacobian matrix is of the form*

$$\boldsymbol{J}^{(\mathcal{T})}(\mathcal{U}) = \boldsymbol{J}_L^{(\mathcal{T})}(\mathcal{U}) \otimes \boldsymbol{J}_R^{(\mathcal{T})}(\mathcal{U})$$

    where $\boldsymbol{J}_L^{(\mathcal{T})}(\mathcal{U})$ is the contraction of all factors to the left of $\mathcal{U}$ and $\boldsymbol{J}_R^{(\mathcal{T})}(\mathcal{U})$ is the contraction of all factors to the left of $\mathcal{U}$
  - *These left and right parts can be orthogonalized by performing QR factorizations starting from the ends of the MPS, absorbing R factors into neighboring MPS factors, until an R factor is absrobed into $\mathcal{U}$ from both sides*

Canonical Forms in DMRG

- The DMRG algorithm maintains an appropraite canonical form for all local computations
  - *In the two-site version, the center of the canonical form should include two factors (so everything to the left and right of these two should be orthogonalized)*
  - *An initial sweep is performed to setup a canonical form with the center being the last factor matrix*
  - *Moving the canonical form to the left is done by performing SVD of the two factors in the center so as to maintain left-orthogonality of the right factor*
  - *Right orthogonality of factors to the left is automatically maintained since its satisfied with request to any subsequent of sites starting from the left most factor*
  - *Consequently, DMRG achieves stability with no additional cost overhead*
  - *However, canonical forms of more complicated tensor networks than the 1D MPS may not generally exist or be easy to compute*

Tensor Networks beyond 1D

- An MPS is efficiently contractable and easy to put into canonical form, but other tensor networks often provide more desirable representations
  - *For example, a 2D tensor network, known in physics literature as a projected entangled pair state (PEPS) [J. Jordan, R. Orus, G. Vidal, F. Verstraete, and J. I. Cirac, 2008]*
  - *PEPS tensor networks satisfy area laws expected in theory for 2D and 3D physical systems*
  - *However, contraction of PEPS is #P-complete, meaning there is no known algorithm to contract a 2D tensor network with a constant rank in polynomial time with the number of factors*
  - *Canonical forms for PEPS networks exist, but an arbitrary PEPS could be not reducible to canonical form, and practical algorithms to put PEPS into canonical form rely on iterative optimization (see e.g., [R. Haghshenas, M.J. O'Rourke, G.K.L Chan, 2019])*
  - *More generally, a particular tensor network provides low-rank representations for a particular class of functions [K. Ye, and L.H. Lim, 2008], hence the desired low-rank tensor network is application-dependent*