Slide 1:

Consider trying to find the ground state of this Hamiltonian for a 1D chain of spin-1/2’s particles. As you may recognize, this is the Ising model with a transverse field (not too important for now).

The Hilbert space of the ground state has dimension 2^N. The Hamiltonian is therefore a 2^N x 2^N matrix. To solve this (by which I mean to find the ground state and ground state wavefunction), I could employ exact diagonalization. While (as Eric knows) I like to use this method whenever I can, it is only good up to N=20 or so before my computer runs out of memory.

When N becomes large, especially when we are interested in the thermodynamic limits, finding the ground state and ground state wavefunction will be like finding a needle in the haystack.

---------------

So are we doomed? No fortunately! It turns out that for relevant Hamiltonians (more specifically Hamiltonians that contain local interactions that are gapped), the haystack is actually quite small. In fact it is infinitesimally small compared to the full Hilbert space. To be more precise, it is rigorously proven that for the local gapped Hamiltonians that I just talked about, the “haystack” is a small subspace of the Hilbert space that contains states with low entanglement entropy – i.e. states whose entanglement entropy follows the area law: entanglement entropy scales like the size of the boundary of the “partition”.

This means that if we could devise some clever parameterization of our guesses for the ground state wavefunction and come up with some efficient algorithm, then we will be able to solve this seemingly intractable optimization problem above.

---------------

Slide 2:

Okay, so we do start to try to come up with these tricks? One thing we could try is to answer the following. Since we want a efficient way to represent a state with some low entanglement entropy, given some full wavefunction |\psi>:

* How do approximate |\psi> well without keeping track of all d^N coefficients? (Here, d is the number of internal states. For the case of spin-1/2, d=2)
* Is it possible to reduce the entanglement entropy by this approximation?

Idea: if we could come up with some way to accomplish both goals, then we could construct some ansatz/guess states with both properties: (a) “compact” and (b) "having low entanglement entropy”

---------------

Slide 3:

The answer, as you might have sensed coming, has a lot to do with the single value decomposition – a well known technique in image processing, especially for image compression (also a very powerful result in linear algebra with many applications). Here I will provide a brief summary of the theory of SVD, specifically two results:

* The first is just the statement of the theorem: singular value decomposition, which states that any matrix M can be decomposed into the form U S V where U,V are unitary matrices and S is a diagonal matrix (not necessarily a square) whose diagonal entries are the *singular values* of M. And just to be complete, the singular values of M are the eigenvalues of the matrix Sqrt[adj(M) M], which means that the singular values are nonnegative.
  + Just to connect to quantum mechanics, in the case where adj(M)M is a density matrix \rho, then |s\_i|^2 are the eigenvalues of \rho, and we can think of them as probabilities (of what you might ask? Coming soon)
* The second result is the well-known *low-rank approximation.* This states that if I am given some matrix A with rank n, then the HS-distance from A its best approximation with rank k, is the sum of the squares of the n-k smallest singular values of A. Here, the HS-distance is simply the sum of the square root of modulus squares of the entries.

---------------

Slide 4:

Just to quickly show you what second result means, I have prepared this slide. Suppose I have some matrix A which I have decomposed into a product of U, Sigma, and V, where Sigma is the rectangular diagonal matrix with singular values highlighted in orange. The theorem says that the closest matrix to A of rank k is this one on the right where you keep k largest singular values of A. It is obvious then that the distance from A to this new matrix is the sum of the squares of the singular values that we just dropped.

As a quick application: If we treat an image as a matrix then we immediately have an application: image compression? On the left is a full-res image, and on the right is a compressed image.

---------------

Slide 5:

Now, it turns out that we can also do this for wavefunctions. The idea is to represent the wavefunction as a matrix, then SVD it.

To do this “matrixification,” we can basically split the system into two parts LEFT and RIGHT, each with their own orthonormal basis. Then, the full wavefunction will be given in *this* form (on the slide), where the coefficients here have two indices – which means we can treat it as a matrix. One thing to note is that this matrix is NOT an operator.

Now we apply SVD on this matrix \psi\_{lr} = [U D V]\_{lr}. Here, U, V are unitary and D is diagonal. s\_i are the singular values of \psi\_{lr} like I mentioned before. They are the eigenvalues of \sqrt{density matrix}, as we have discussed before.

---------------

Slide 6:

When I said that s\_i^2 could be interpreted as probabilities, we could have asked “okay, probabilities for what?” and this is where we go a little further. If we look at the SVD of \psi more carefully, we will notice that the unitary matrices U and V have (in the background) transformed the orthonormal bases into new ones which makes the wavefunction take on a clean and “diagonal” form. We call this the **Schmidt decomposition**, and the states |i>\_{L,R} are called the Schmidt states. We now interpret the s\_i^2 as the probabilities for the Schmidt states.

One really nice thing about the Schmidt decomposition is that from it we could immediately read off the reduced density matrix for each subsystem.

And the fact that we can interpret the s\_i^2 as probabilities is cemented by the normalization condition on \psi!

---------------

Slide 7:

Let’s do a quick example. Consider the following wavefunction.

Let’s matrixify by splitting the system down the middle. So we have two small Hilbert spaces, each with one spin. The system can then be written in this form, with the matrix \psi\_{ij} being…

Okay, doing a SVD on this matrix, we find the following decomposition. We could also calculate the Schmidt states, but I won’t show the results here.

---------------

Slide 8:

So, coming back to our timeline. So far, we have answer our first question of whether there is some procedure that can let us approximate some complicated wavefunction. But about the second one about entanglement entropy? Does \psi approximation using SVD and Schmidt decomposition reduce the entanglement entropy? Is for this very reason that SVD is the absolute best candidate.

To see this, let us look at how von Neumann entanglement entropy is calculated. By definition, the von Neumann entanglement entropy can be calculated using the reduced density matrices (turns out that either one can be used… since they have the same spectrum).

Now do we calculate this? Well, instead of using the usual definition of these reduced density matrices where you would trace out the density matrix of the entire system, we also know what these are in terms of the Schmidt states, from which we know that the spectrum of \rho\_L, \rho\_R are nothing more than the s\_i^2 themselves. And so the von Neumann entanglement entropy is simply this.

Under compression, what people typically do is keep the first 100s significant singular values or so and drop ~2^N of the remaining values (which are on the order of 10^-6 or something like that). This immediately reduces the entanglement entropy and give us exponential compression!

Truly amazing!

---------------

Slide 9:

Here’s the old example. Let us calculate the entanglement entropy in this case. It is obvious that because ln(1) = 0, S = 0. This is as expected, because \psi is a product state.

---------------

Slide 10:

Okay, so far we have seen how the SVD gives us a way to generate a low-entanglement entropy approximation of some given some wavefunction \psi.

But wait, you might ask… why are we doing this exactly? Isn’t our goal to find the ground state wavefunction of some complicated Hamiltonian? If we don’t have some wavefunction to start, what is the point of this compression procedure. There’s nothing to compress? So we have kind of a chicken-and-egg problem.

---------------

Slide 11:

This brings me to the second part of my ZGS which will most be on the matrix product state and finally on the DMRG (roughly).

What we have done so far with SVD and Schmidt decomposition is to set up the scene for introducing MPS, which stands for matrix product state, which is the basis for this DMRG method for very efficiently finding ground states of 1d Hamiltonians.

As a rough idea, MPS is a way of expressing wavefunctions, but instead of in terms of coefficients, it is in terms of product of matrices. Any wavefunction can be expressed in terms of a matrix product state. However, the advantage of matrix product states is that it is a very natural way to express, or more importantly, to generate states with low entanglement entropy, which is suitable for ground state searches (just to continue with the haystack analogy… basically by starting in the MPS, we are already making our haystack very small)

---------------

And just a few works on DMRG:

* It is a numerical, variational method for ground state searches
* Most efficient method for 1d systems
* Works with MPS, not \psi. So these guys go hand in hand

---------------

Slide 12:

To show you what a matrix product state, I feel like it is best to do this constructively… in the sense that I will show you have to convert a \psi in wavefunction form into an MPS. Let us begin with some wavefunction written in the Schmidt decomposition form. Here, I’ve added the subscript m to indicate where the L-R split happens on the chain of N particles. So m here is a physical index.

What I want to do next is to write the Schmidt states in terms of the standard basis with spins. And I will do this recursively. If the cut is at m=1, then the Schmidt state is just some superposition of the internal levels of a single particle. So, we just have this expression. Here, A \sigma\_1 i\_1 is a row vector of size (1 x d), where the d the number of possible values for \sigma\_1. The i\_1 subscript is just a label for the site location. So far so good.

Next, we basically move the cut over, or can also think of this as adding a spin. Then, we have a similar construction as before, where we are combining two systems: i\_1 and \sigma\_2. And we now need a matrix. By inserting the first expression and summing over i\_1, we’re left with the expression on the far right. Here, the first matrix has dimension (1xd), the second matrix has dimension (dxd^2).

Now we continue the pattern:… Here, A^\sigma\_3 has dimension (d^2xd^3), and so on

---------------

Slide 13:

Ultimately, we have that the Schmidt states for the L/R subsystems have the form: (slide)

The full wavefunction can now be written as (slide) where the sum is over all spin configurations. And S is the diagonal matrix with all the singular values s\_i = S\_{i,i}.

Here, the matrix dimensions are:

(1xd), (dxd^2), … (d^2,d), (dx1),

So that the entire expression AAAAASBBBBBB is a scalar!

This is \psi in the MPS form.

---------------

Slide 14:

Just to give an example, consider the GHZ state, which we say should take this form in the MPS language. What would the matrices be? It turns out that they are:

(slide)

And as you can see, the product of these matrices is 1 exactly when all of the \sigma’s are 0 or 1.

---------------

Slide 14:

Okay, so in what sense are matrix product states a natural language for generating the so-called compressed states with low entanglement entropy? Ans: it is “natural” in the sense that one could easily generate such a state by limiting the dimensions of the matrices in the product. What I mean is that instead of letting the matrices grow up to d^(N/2) like the case before (where we want to exactly represent the original wavefunction), we could just set some cutoff D for the matrix dimensions.

What you could imagine is that the matrices grow in dimension up to D only. Again, like before, D is typically on the order of 100-1000, where as 2^N ~ 2^1000. This is essentially the same as dropping all but the D most dominant eigenvalues of diagonal matrix in SVD.

Also, to make handling matrix dimensions less annoying, sometimes we can simply square all matrices to DxD, and then take the trace of the matrix product. In the end, the number of entries is D^2 \* N, which is typically much less than 2^(N/2). And what we can do in the end is simply take the trace of this matrix product and get a scalar and have the following form for matrix product states. This form is what you will typically find in arxiv papers of on Wikipedia.

---------------

Slide 15:

To really show that this definition using the trace really works the same way, I could come back to the example with the GHZ state. Instead of obtaining a representation via the brute force route using big matrices, we could observe that the density matrix associated with this state is something of rank 2. This means the diagonal matrix after SVD is also going to be rank-2. This means we can simply use 2x2 matrices to handle the state representation, and these are the necessary matrices that do the job. And once again, we could set all matrices to be 2x2 and take the trace of the product like so. Again, the trace is 1 exactly when the spins are all 0 or all 1, and is 0 otherwise.

---------------

Slide 16:

That’s the first half of my talk, and just to give a summary of what we’ve seen so far:

* First we looked at how we could use SVD to compress some known \psi
* We also introduced the Schmidt decomposition, and used it to build the MPS representation
* Then, we now know how to make Hilbert spaces small! From the wavefunction side, we make Hilbert spaces small by the low-rank approximation: dropping small entries of the diagonal matrix in the SVD… From the MPS side, that is equivalent to limiting the dimension of the constituent matrices in the MPS.
* All this gives us exponential compression aka exponential decimation of the full Hilbert space. The new Hilbert space is much smaller (ND^2 vs d^N) and the MPS corresponds to \psi with low entanglement entropy
* For these reasons, MPS is a natural language for describing ground states of relevant Hamiltonians

---------------

Slide 17:

The last half/third of my zgs will be on how ground state search works in the MPS language. The first thing that you will see if you try to look at the literature regarding this part is “tensor networks.” Basically, it’s a graphical representation of MPS states, and it’s a nice way to represent what you need to do without writing out all of the sums and the indices (physical indices or just dummy).

To start, the circles represent matrices. The ones of the left and right represent rank-1 tensors: these are vectors. The one is the middle is a rank-2 tensor, so it’s a matrix. Of course, you will tell me that a vector is also a matrix.

Matrix multiplication is done by connecting the legs. Here, the legs represent the indices being summed over. The top leg is a physical index, while horizontal legs are “bonds.” Since a MPS as a product of matrices, an MPS looks like this: connected circles with N physical indices. This makes sense, because the coefficient for a basis state is \psi with N indices.

Using this graphical notation, we could represent an overlap between \psi with itself or \psi with some other state. In this case, the conjugate transpose is similar to the MPS except with the vertical legs pointing down instead of up. The overlap is represented like so. The lines represent sums.

So not only are we summing over the indices, we’re also summing over the spins (physical). You end up with something with no legs, and it’s a scalar (as expected).

I won’t go into the details of how to explicitly write down the summations. I’m sure you can work it out for yourself.

---------------

Slide 18:

Operators can be defined in a similar way. If the “coefficient” of a wavefunction is given as a product of matrices where there is a matrix for every site, then the matrix elements of an operator can also be define in a similar fashion, except that each matrix factor now has two indices, sigma 1 and sigma 2, for example.

Essentially, any operator can be written in this form, where each matrix here now has two indices (think of this as a matrix-valued matrix)… because each constituent matrix will be “sandwiched” between two A-matrices from the other MPS.

In any case, with MPS and MPO, we can now evaluate expectation values and so on. And of course, the expectation value of highest interest here is the energy.

---------------

Slide 19:

Graphically, an MPO acting on an MPS looks something like this. As you can see, one of the physical indices on W is summed out, and we end up with a new matrix product state.

---------------

Slide 20:

With this brief aside, we can now look at how ground state searches typically work. Suppose we want to find the ground state and ground state energy for some Hamiltonian H. What we would do is extremize the following expression, where \lambda is the Lagrangian multiplier and after some variational procedure, \psi 🡪 \psi\_0 (which is the ground state) and \lambda 🡪 E\_0 the ground state energy

Graphically, this looks like this: (show)

And how do we do this? Well, how do we do this? The answer is gradient descent. The idea is to look at one site l, keeping everywhere else constant, and optimize the matrix A^{\sigma\_l}.

---------------

Slide 21:

What does this look like? How does gradient descent look like in this “tensor network” language?

Well, minimizing the full expression up here, is the same as solving for this equation below, which as we may recognize is the derivative of the top one with respect to the matrix of interest. (d/dA^\sigma\_l).

And this is a generalized eigenvalue problem, which could be solved efficiently.

---------------

Slide 22:

As another aside, if the matrices on the left and the right of A^{\sigma\_l} are left- and right-normalized, then (back to old slide), we can simply contract them to identity, the network simplifies to a more standard eigenvalue problem, which is better.

So in practice, we make sure that the matrices in the MPS are properly normalized

---------------

Slide 23:

The algorithm for ground state search is then as follows:

* Start with an initial guess for \psi, written in some form \psi = BBBBBB… where all the B matrices are right-normalized.
* Calculate the current state of the network for sites N-1 to 1
* Now we do a *right-sweep:* for each l in l=1 to l=N-1, we optimize/solve for the matrix M^\sigma\_l. Then, we SVD it, so that M = AM’U, and A is now a left-normalized matrix and we let U be absorbed into the next matrix down the line, which is M^{\sigma\_{l+1}}
* After the right-sweep is done, we do a *left-sweep:* for each l in l=L to l=2, do the same thing: solve for M^\sigma\_l, then SVD into UMB where B is right-normalized, and we let U be aborbed into the next M down the line.
* Repeat until convergence

---------------

Slide 23:

Formally, the algorithm looks like this (repeat)

---------------

Slide 24:

In practice, however, this algorithm has a number of problems related to convergence issues… the fact that it is prone to getting stuck in local minima, etc. (refer to the reference for better details).

So in practice, the algorithm that is used is not this, but something else called DMRG. DMRG stands for density-matrix renormalization group. There’s some history behind the naming here, which I’m not too clear on, but the point is there’s really no group theory involved in the renormalization process.

The only thing I will say about the “density-matrix renormalization” part is that it will be a little more clear after I’ve gone through what the algorithm is (roughly speaking).

* We don’t start with the full system. And instead build it up from smaller pieces
* In particular, we first introduce left and right blocks called A, B with only one spin in each
* This is easily solved by exact diagonalization
* Then, insert a pair of spins between the blocks, so now the chain looks like A oo B
* Once again diagonalize to get ground state energy and wavefunction.
* We can’t keep doing because the Hilbert space will get too big. So, we have to start decimating the Hilbert space at some point. How do we do this?
  + The solution is to find reduced Hilbert space for the Ao and oB (which are now the two L and R) blocks.
  + We already know how to do this! Since we already have a |\psi\_0> and the L+R blocks, we could do a Schmidt decomposition on \psi\_0, and in the spirit of SVD, keep only the D Schmidt pairs with highest eigenvalues! 🡪 this is the same as finding the best rank-D (out of Dd) density matrix for the subsystems.
  + With this, we have stopped the exponential growth of the Hilbert space
* Then, repeat.

---------------

Slide 25:

More formally, this look like the left picture (repeat what was said in previous slide, but more overview-vibe).

---------------

The end