[Solution] Tensor decomposition and entanglement entropy

Author: Seung-Sup Lee

Solution to Exercise (a): Check the integrity of the tensor decomposition

For this solution, I will use the function `contract` (under `Tensor` directory) which is introduced as a solution of another tutorial "Tensor contractions".

Let's repeat the construction and the decomposition of the same rank-5 tensor T.

```
clear
sz = [2 3 2 3 4]; % local space dimensions
T = reshape((1:prod(sz)),sz); % rank-5 tensor
T = T/norm(T(:)); % normalize

Q = cell(1,numel(sz));
R = T; % temporary tensor to be QR-decomposed
szl = 1; % the bond dimension of the left leg of Q{n} to be obtained after
% the QR decomposition at iteration n; for n = 1, szl = 1 for the dummy leg
for it = (1:(numel(sz)-1))
    R = reshape(R,[szl*sz(it), prod(sz(it+1:end))]);
    [Q{it},R] = qr(R,0);
    Q{it} = reshape(Q{it},[szl, sz(it), numel(Q{it})/szl/sz(it)]);
    Q{it} = permute(Q{it},[1 3 2]); % permute to the left-right-bottom order
    szl = size(Q{it},2); % update the bond dimension
    R = reshape(R,[szl,sz(it+1:end)]);
end
Q{end} = permute(R,[1 3 2]);
```

Now let's contract the tensors `Q{n}` to make a rank-5 tensor again.

We first remove the first (i.e., left) leg of `Q{1}` which is dummy, either by reshaping `Q{1}` or by permuting the first leg to the end. The latter approach works since MATLAB automatically suppresses trailing singleton dimensions; such permuted leg will not appear explicitly in the array. Also, we want to sort the remaining two legs of `Q{1}` in the bottom (physical)-right order. All these treatment can be done by a single line of `permute`:

```
T2 = permute(Q{1},[3 2 1]);
```

And we contract tensors.

```
for it = (2:numel(sz))
    T2 = contract(T2,it,it,permute(Q{it},[1 3 2]),3,1);
end
```

Let's check whether T2 and T are the same.

```
size(T)
```

```
ans = 1×5
```

```
     2     3     2     3     4
```

```
size(T2)
```

```
ans = 1×5
     2     3     2     3     4
```

```
max(abs(T(:)-T2(:)))
```

```
ans = 3.8858e-16
```

The maximum difference between their corresponding elements is of order of double precision noise $\sim 10^{-16}$. So T and T2 are numerically equivalent.

Solution to Exercise (b): Entanglement entropies for different bipartitions

Let's compute the entanglement entropies for different bipartitions.

```
% (i) A = {1, 2}, B = {3, 4, 5}
svec = svd(reshape(T,[sz(1)*sz(2) prod(sz(3:5))])); % singular values
Spart = -(svec.^2).*log(svec.^2)/log(2); % contributions to entanglement entropy
disp(sum(Spart(~isnan(Spart)))) % entanglement entropy
```

```
    0.0015
```

Here I have chosen only the non-NaN elements of `svec`, which can happen for vanishing singular values.

For the other ways of bipartitioning, we need to permute the legs of T.

```
% (ii) A = {1, 3}, B = {2, 4, 5}
svec = svd(reshape(permute(T,[1 3 2 4 5]), ...
    [sz(1)*sz(3) prod(sz([2 4 5]))])); % singular values
Spart = -(svec.^2).*log(svec.^2)/log(2); % contributions to entanglement entropy
disp(sum(Spart(~isnan(Spart)))) % entanglement entropy
```

```
    0.0042
```

```
% (iii) A = {1, 5}, B = {2, 3, 4}
svec = svd(reshape(permute(T,[1 5 2 3 4]), ...
    [sz(1)*sz(5) prod(sz(2:4))])); % singular values
Spart = -(svec.^2).*log(svec.^2)/log(2); % contributions to entanglement entropy
disp(sum(Spart(~isnan(Spart)))) % entanglement entropy
```

```
    0.0343
```

Solution to Exercise (c): Use the SVD for the tensor decomposition and compute the entanglement entropy

This process can be implemented in a similar way of using the QR decomposition. The differences are that (i) the left-unitary matrices "U" whose columns are left singular vectors are reshaped to be `M{n}`, and that (ii) the entanglement entropy is computed.

```
M = cell(1,numel(sz)); % MPS tensors
```

```
Sent = zeros(1,numel(sz)-1); % entanglement entropy
R = T; % temporary tensor to be SVD-ed
szl = 1; % the bond dimension of the left leg of R{n} to be obtained
% after the SVD at iteration n; trivially 1 for n = 1
for it = (1:(numel(sz)-1))
    R = reshape(R,[szl*sz(it), prod(sz(it+1:end))]);
    [U,S,V] = svd(R,'econ');
    svec = diag(S); % singular values

    % truncate the column vectors of U and V associated with
    % singular values smaller than eps
    ok = (svec < eps);
    U(:,ok) = []; V(:,ok) = [];

    M{it} = reshape(U,[szl, sz(it), numel(U)/szl/sz(it)]);
    M{it} = permute(M{it},[1 3 2]); % permute to the left-right-bottom order
    szl = size(M{it},2); % update the bond dimension
    R = reshape(diag(svec(~ok))*V',[szl,sz(it+1:end)]);

    % compute entanglement entropy
    Spart = -(svec.^2).*log(svec.^2)/log(2);
    Sent(it) = sum(Spart(~isnan(Spart)));
end
M{end} = permute(R,[1 3 2]);
```

The resulting `M{n}`'s have smaller dimensions as the corresponding `Q{n}`'s, since truncations are made.

```
M
```

M = 1×5 cell

	1	2	3	4	5
1	1×2×2 double	2×2×3 double	2×3×2 double	3×3×3 double	3×1×4 double

```
Q
```

Q = 1×5 cell

	1	2	3	4	5
1	1×2×2 double	2×6×3 double	6×12×2 double	12×4×3 double	4×1×4 double

The values of entanglement entropy are:

```
Sent
```

```
Sent = 1×4
    0.0002    0.0015    0.0053    0.0344
```

Note that the value of `Sent(2)` is the same as the result obtained in the above Exercise (b) for the bipartition of A = {1, 2}, B = {3, 4, 5}, since the second iteration treats the same bipartition.

Let's check whether the contraction of `M{n}`'s give the original tensor T. Here we take the code lines from above, only replacing `Q` with `M`.

```
T2 = permute(M{1},[3 2 1]);
for it = (2:numel(sz))
    T2 = contract(T2,it,it,permute(M{it},[1 3 2]),3,1);
end
size(T)
```

```
ans = 1×5
     2     3     2     3     4
```

```
size(T2)
```

```
ans = 1×5
     2     3     2     3     4
```

```
max(abs(T(:)-T2(:)))
```

```
ans = 2.9057e-16
```

Though `M{n}`'s have smaller sizes by truncation, we can still get the original tensor T. It shows that the SVD can be used to rule out useless parts of tensors!