Tensor Operations in Sage using Python libraries as backend

Mentors

Matthias Köppe

Aman Moon
Indian Institute of Technology Bombay

amanmoon

Aman Moon

SageMath is an open-source mathematics software system designed to provide a viable free alternative to commercial software like MATLAB, Mathematica, and Maple. It integrates many existing open-source mathematics libraries into a single, unified interface, making it a powerful tool for mathematical research, education, and exploration.

Project Description

In SageMath, tensors are currently stored in the Components class as a dictionary of indices and values. My Google Summer of Code project mainly focused on adding a new backend for this Component class and implementing complex decomposition methods.

A new backend utilizing the NumPy library has been added to improve memory efficiency and performance in tensor operations.

Now the user can select between backends and choose what library to use to perform tensor operations.

Contribution

Link to forked Repository: https://github.com/amanmoon/sage
Pull Request: #38325 (open) NumPy Backend

API for the new backend-numpy is as follows:

```
sage: N = FiniteRankFreeModule(ZZ, 3)
sage: t = N.tensor((1,1), implementation='numpy')
sage: e = N.basis('e')
sage: t[1,1] = 1
sage: t[:]
array([[0., 0., 0.],
       [0., 1., 0.],
       [0., 0., 0.]])
Sage: t._implementation
‘numpy’
```

Created tensor decomposition methods like Canonical Polyadic decomposition, Tensor-Train Decomposition, and Tucker decomposition.
API for decomposition algorithms:

```
sage: from sage.tensor.modules.comp_numpy import ComponentNumpy
sage: c = ComponentNumpy(ZZ, [1,2,3], 3)
sage: c[:] = [[[ 1,  2,  3], [ 4,  5,  6], [7,   8,  9]],
....:         [[10, 11, 12], [13, 14, 15], [16, 17, 18]],
....:         [[19, 20, 21], [22, 23, 24], [25, 26, 27]]]
```

```
sage: weights, factors = c.CPD(2) #Canonical Polyadic decomposition
sage: weights
array([1, 1])
sage: factors
[array([[42.60976245,  4.24438786],
        [58.75054048,  2.8414095 ],
        [74.89131852,  1.43843114]]),
 array([[0.53636712,   1.67807245],
        [0.59142972,   1.4468178 ],
        [0.64649232,   1.21556315]]),
 array([[ 0.57740747, -1.71787524],
        [ 0.59592194, -1.63132892],
        [ 0.61443641, -1.54478261]])]

sage: c.TT((1,2,3,1)) # tensor-train decomposition
[array([[-0.19223057,   0.89240167],
        [-0.51407797,   0.26278731],
        [-0.83592538,  -0.36682705]]),
 array([[[-0.47434688, -0.20442917,  0.50294746],
         [-0.57054452, -0.03010924, -0.70322613],
         [-0.66674217,  0.14421069,  0.28420507]],

        [[-0.05473072,     0.64307203,     -0.2033407 ],
         [-0.00555624,     0.55396286,      0.00647477],
         [ 0.04361825,     0.46485368,      0.36104705]]]),
 array([[ 4.53733725e+01,  4.79999709e+01,  5.06265692e+01],
        [-1.50235382e+00, -5.28776934e-02,  1.39659843e+00],
        [-1.95748426e-15,  3.91496852e-15, -1.95748426e-15]])]

sage: core, factors = c.Tucker((3,3,3))
sage: core
array([[[-8.30159812e+01, -3.70997401e-02, -1.10470575e-14],
        [-1.26803960e-02,  5.15711162e-01,  2.01400152e-15],
        [ 9.26001262e-16, -2.43094170e-17, -7.25194643e-16]],

       [[-5.00588180e-03,  1.91521381e+00,  7.25632942e-16],
        [ 5.84263038e+00, -5.24377612e-01,  2.01281470e-16],
        [-4.86116219e-15,  1.41265092e-15, -4.57825373e-16]],

       [[-7.49496258e-15, -2.04492748e-16,  8.29576015e-16],
        [ 1.57910822e-15,  3.61487339e-18, -1.14075975e-16],
        [-4.18274041e-16,  1.86882217e-17,  1.18423789e-15]]]),
```

```
Sage: factors
[array([[-0.19223057,  0.89240167,  0.40824829],
        [-0.51407797,  0.26278731, -0.81649658],
        [-0.83592538, -0.36682705,  0.40824829]]),
 array([[-0.47563923,  0.77916664,  0.40824829],
        [-0.57196783,  0.07865197, -0.81649658],
        [-0.66829643, -0.6218627 ,  0.40824829]]),
 array([[-0.54521326,  0.732172  ,  0.40824829],
        [-0.57677486,  0.02576994, -0.81649658],
        [-0.60833646, -0.68063212,  0.40824829]])])
```

What is left to do

Implementation of the tensor decomposition algorithms took longer than expected.
I am planning to work on this issues after my GSoC project.

- Implementation of the **PyTorch** backend.
- Implementation of the **to_sympy** method for porting components to SymPy.