birkhoff Documentation

Release 0.0.6.dev

Jeffrey Finkelstein

Dec 19, 2017

Contents

1 Installation **3**

2 Basic usage **5**

3 Mathematical background **7**

3.1 Birkhoff's Algorithm 7

4 API **9**

5 Changes **11**

Python Module Index **13**

This package provides Birkhoff's algorithm for computing the Birkhoff–von Neumann decomposition of a doubly stochastic matrix.

Source code · Packaging · Issues

CHAPTER 1

Installation

```
pip install birkhoff
```

CHAPTER 2

Basic usage

```python
import numpy
from birkhoff import birkhoff_von_neumann_decomposition

# Create a doubly stochastic matrix.
#
# D = numpy.array(...)

# The decomposition is given as a list of pairs in which the right element
# is a permutation matrix and the left element is the scalar coefficient
# applied to that permutation matrix in the convex combination
# representation of the doubly stochastic matrix.
result = birkhoff_von_neumann_decomposition(D)
for coefficient, permutation_matrix in result:
    print('coefficient:', coefficient)
    print('permutation matrix:', permutation_matrix)
```

CHAPTER 3

Mathematical background

A *doubly stochastic matrix* is a matrix in which each row and each column sum to one. In other words, a matrix D is doubly stochastic if

$$D1 = 1$$
$$1^T D = 1^T$$

A *permutation matrix* is a matrix in which each entry is either zero or one. By the Birkhoff–von Neumann Theorem, each doubly stochastic matrix is a convex combination of permutation matrices. In other words, for each $n \times n$ doubly stochastic matrix D, there is a sequence of real numbers $\alpha_1, \ldots, \alpha_N$ and permutation matrices $P_1, \ldots, P_N$ such that

$$D = \sum_{i=1}^{N} \alpha_i P_i$$

where $\sum_{i=1}^{N} \alpha_i = 1$ and the number N is guaranteed to be at most n^2. Furthermore, the proof of the Birkhoff–von Neumann Theorem provides an explicit algorithm for computing each α_i and P_i. This is the algorithm employed by the `birkhoff_von_neumann_decomposition()` function.

The theorem and corresponding algorithm also apply to scalar multiples of doubly stochastic matrices, that is, matrices of the form cD, for some positive real number c.

3.1 Birkhoff's Algorithm

Describe the algorithm here.

CHAPTER 4

API

birkhoff.**birkhoff_von_neumann_decomposition**(*D*)

Returns the Birkhoff–von Neumann decomposition of the doubly stochastic matrix *D*.

The input *D* must be a square NumPy array representing a doubly stochastic matrix (that is, a matrix whose entries are nonnegative reals and whose row sums and column sums are all 1). Each doubly stochastic matrix is a convex combination of at most `n ** 2` permutation matrices, where `n` is the dimension of the input array.

The returned value is a list of pairs whose length is at most `n ** 2`. In each pair, the first element is a real number in the interval **(0, 1]** and the second element is a NumPy array representing a permutation matrix. This represents the doubly stochastic matrix as a convex combination of the permutation matrices.

The input matrix may also be a scalar multiple of a doubly stochastic matrix, in which case the row sums and column sums must each be *c*, for some positive real number *c*. This may be useful in avoiding precision issues: given a doubly stochastic matrix that will have many entries close to one, multiply it by a large positive integer. The returned permutation matrices will be the same regardless of whether the given matrix is a doubly stochastic matrix or a scalar multiple of a doubly stochastic matrix, but in the latter case, the coefficients will all be scaled by the appropriate scalar multiple, and their sum will be that scalar instead of one.

For example:

```
>>> import numpy as np
>>> from birkhoff import birkhoff_von_neumann_decomposition as decomp
>>> D = np.ones((2, 2))
>>> zipped_pairs = decomp(D)
>>> coefficients, permutations = zip(*zipped_pairs)
>>> coefficients
(1.0, 1.0)
>>> permutations[0]
array([[ 1.,  0.],
       [ 0.,  1.]])
>>> permutations[1]
array([[ 0.,  1.],
       [ 1.,  0.]])
>>> zipped_pairs = decomp(D / 2)  # halve each value in the matrix
>>> coefficients, permutations = zip(*zipped_pairs)
>>> coefficients  # will be half as large as before
(0.5, 0.5)
>>> permutations[0]  # will be the same as before
array([[ 1.,  0.],
       [ 0.,  1.]])
>>> permutations[1]
array([[ 0.,  1.],
       [ 1.,  0.]])
```

The returned list of pairs is given in the order computed by the algorithm (so in particular they are not sorted in any way).

CHAPTER 5

Changes

Not yet released.

No changes yet.

Released on December 19, 2017.

- Updated code to work with NetworkX version 2.0 (issue #3).

Python Module Index

b

birkhoff, 9

Index

B

birkhoff (module), 9

birkhoff_von_neumann_decomposition() (in module birkhoff), 9