Python Assignment 3

Alex Milut

Mike Vanger

**EINSUM**

This function was built on Python 2.x. The end of the script contains three commented-out examples.

my\_einsum(*subscripts*, \**operands*)

*subscripts*

A string consisting of three parts: the input tensors, ‘->’, and the output tensors. The input tensors are comma-separated for each operand matrix, and each tensor of each matrix must be specified. The function mimics the einsum() function of the numpy package.

\**operands*

The numpy arrays over which to take the Einstein summation.

Examples:

a = np.arange(60.).reshape(3,4,5)

b = np.arange(24.).reshape(4,3,2)

print my\_einsum('ijk,jil->kl', a, b)

# result

# [[ 4400. 4730.]

# [ 4532. 4874.]

# [ 4664. 5018.]

# [ 4796. 5162.]

# [ 4928. 5306.]]

a = np.arange(25).reshape(5,5)

b = np.arange(5)

print my\_einsum('ij,j->i', a, b)

# result

# [ 30. 80. 130. 180. 230.]

print my\_einsum('ii->i', a)

# result

# [ 0. 6. 12. 18. 24.]

print my\_einsum('ii->', a)

# result

# 60.0

**MST**

This function was built on Python 2.x. The MST program implements Kruskal’s algorithm to find the minimum spanning tree over a user-specified network. The network is input as an n x 2 numpy array, where each row of the array represents an x,y coordinate location. The program returns a tuple containing the tree length and the edges included in the tree. Should more information on the network or the selection algorithm be desired, the respective lines of code can be uncommented and the program will output the relevant information.

Usage:

1. Import the MST class.

e.g. *import MSTclass as m*

2. Instantiate the class using the MST command and specifying the numpy array name.

e.g. *network = m.MST(numpyPoints)*

3. Use getMST() function in order to calculate and return the minimum spanning tree

e.g. *tree = network.getMST()*

4. Program returns a two-element tuple consisting of the minimum spanning tree length and a list of the edges included in the tree. The result can be viewed using the print command.

e.g. *print tree*

*(220.38781999921534, ['3,4', '1,7', '0,6', '2,3', '4,9', '1,5', '6,9', '0,5', '1,8'])*

*Example network:*

***Raw randomly-generated network.***

*[[25.106436556848255, 33.11583380351523], [62.809627435200156, 10.794141443593396], [5.950316244837273, 91.90195242156311], [17.373888408020342, 84.33391273846263], [23.152762314339824, 81.87994509602639], [35.49682924740668, 3.931852120332835], [11.956289559499055, 32.01778863553745], [74.2324704285982, 6.24233588940486], [91.64620429001262, 67.95363178421961], [14.261614190772764, 60.79800971430378]]*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

***Converted to numpy array***

*[[ 25.10643656 33.1158338 ]*

*[ 62.80962744 10.79414144]*

*[ 5.95031624 91.90195242]*

*[ 17.37388841 84.33391274]*

*[ 23.15276231 81.8799451 ]*

*[ 35.49682925 3.93185212]*

*[ 11.95628956 32.01778864]*

*[ 74.23247043 6.24233589]*

*[ 91.64620429 67.95363178]*

*[ 14.26161419 60.79800971]]*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

***Program outputs***

*(220.38781999921534, ['3,4', '1,7', '0,6', '2,3', '4,9', '1,5', '6,9', '0,5', '1,8'])*