Probabilistic Graphical Models Documentation

Release 0.0.1

Avinash Kori

Jun 30, 2020

Contents:

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

1.1 Installation

This package is available on the PyPi repository. Therefore you can install, by running the following.

```
pip3 install ppgm
```

1.2 Usage

Check examples to understand all routines

1.3 Representation

1.3.1 1. LinkedList BN representation

```
from pgm.helpers.common import Node
from pgm.representation.LinkedListBN import Graph

rootNode = Node('rootNode')

graph = Graph(rootNode)
graph.add_node('node1', ['rootNode'])
```

(continues on next page)

(continued from previous page)

```
graph.add_node('node2', ['rootNode'])
graph.add_node('node3', ['node1', 'node2'])
graph.add_edge('rootNode', 'node3')
graph.print(rootNode)

# =============== OUTPUT ===============
node: rootNode, children: ['node1', 'node2', 'node3'], parents: []
node: node1, children: ['node3'], parents: ['rootNode']
node: node2, children: ['node3'], parents: ['rootNode']
node: node3, children: [], parents: ['node1', 'node2', 'rootNode']
```

1.3.2 2. LinkedList MN representation

```
from pgm.helpers.common import Node
from pgm.representation.LinkedListMN import Graph

rootNode = Node('rootNode')

graph = Graph(rootNode)
graph.add_node('node1', ['rootNode'])
graph.add_node('node2', ['rootNode', 'node1'])
graph.add_node('node3', ['node1'])
graph.add_edge('node2', 'node3')
graph.print(rootNode)

# =============== OUTPUT ===============
node: rootNode, nbrs: ['node1', 'node2']
node: node1, nbrs: ['rootNode', 'node2', 'node3']
node: node2, nbrs: ['rootNode', 'node1', 'node3']
node: node3, nbrs: ['node1', 'node2']
```

1.3.3 3. Set Local (conditional) Distribution

Set conditional distribution per node:

In case of BN

$$\mathbb{P}(node \mid Par_{node})$$

```
from pgm.helpers.common import Node
from pgm.representation.LinkedListBN import Graph

rootNode = Node('rootNode')
rootNode.values = [0,1,2]
rootNode.set_distribution(probability=None)
# you can set probability to some flag to directly feed values
# else you need to manually enter conditional probabilities

graph = Graph(rootNode)
graph.add_node('node1', ['rootNode'])
graph.print(rootNode)
```

(continues on next page)

(continued from previous page)

```
node1 = graph.get_node('node1')
node1.values=[1, 2, 3]
node1.set_distribution(probability=None)
```

In case of MN

$$\mathbb{P}(node \mid Nbr_{node})$$

It's very similar to the example of BN showen above

```
from pgm.helpers.common import Node
from pgm.representation.LinkedListMN import Graph

rootNode = Node('rootNode')
rootNode.values = [0,1,2]
rootNode.set_distribution()


graph = Graph(rootNode)
graph.add_node('node1', ['rootNode'])
graph.print(rootNode)

node1 = graph.get_node('node1')
node1.values=[1, 2, 3]
node1.set_distribution()
```

1.3.4 4. Caculate Conditional

$$\mathbb{P}(nodeB \mid nodeA) = \frac{\mathbb{P}(nodeB, nodeA)}{nodeA}$$

Not Implemented yet

1.3.5 5. Caculate Marginal

$$\mathbb{P}(nodeB \mid somenodes) = \sum_{nodeA} \mathbb{P}(nodeA, nodeB \mid somenodes)$$

Not Implemented yet

1.4 Inference

1.4.1 1. MH: Metropolis Hastings

```
from pgm.inference.MetropolisHastings import MH
from matplotlib import pyplot as plt

def Gamma(theta, k = 1):
    def G(k):
        if k <= 0: return 1
        elif k == 0.5: return np.pi **0.5
```

(continues on next page)

(continued from previous page)

```
        return k*G(k-1)
    def distribution(x):
        x = np.abs(x)
        return (x**(k-1))*np.exp(-x/theta)/((theta**k) * G(k))
    return distribution


def proposalDistribution(sigma=0.1):
    """
        Describes example proposal distribution
        considers gaussion distribution with fixed sigma
        as the mean keeps changing it's made an inner function argument
    """
    def QDistribution(param = 0):
        return lambda x: (1/(((2*np.pi)**0.5) * sigma))*np.exp(-((x-param)**2)/␣
↪(sigma**2))

    return QDistribution, lambda x: np.random.normal(x, sigma)

x = np.linspace(-20, 20, 500)
fx = function(x)

proposalDist, proposalSamp = proposalDistribution(sigma = 2.0)
mh = MH(function, 100, proposalDist, proposalSamp)
for _ in range(1000):
    next(mh.sampler())

sampledvalues = np.array(mh.x_seq)
plt.plot(x, fx, 'b--', linewidth=2.0)
plt.hist(sampledvalues, 50, density=True, stacked=True, facecolor='g', alpha=0.7,␣
↪linewidth=0)
plt.legend(['target pdf', 'sampled histogram'])
plt.show()
```

1.4.2 2. Gibbs Sampling

1.4.3 3. Message Parsing

1.4.4 4. Loopy Belief

1.5 Learning

1.6 Helpers

1.6.1 1. Search dfs

Use type='MN' for Markov Networks

```
from pgm.helpers.search import dfs

root = rootNode
```

(continues on next page)

(continued from previous page)

```
searchNode = 'node2'
node = dfs(root, searchNode, type='BN').searchNode
```

1.6.2 2. Search bfs

Use type='MN' for Markov Networks

```
from pgm.helpers.search import bfs

root = rootNode
searchNode = 'node2'
node = bfs(root, searchNode, type='BN').searchNode
```

1.6.3 2. GetTrails for BN

```
from pgm.helpers.common import Node
from pgm.representation.LinkedListBN import Graph
from pgm.helpers.trails import findTrails

rootNode = Node('rootNode')

graph = Graph(rootNode)
graph.add_node('node1', 'rootNode')
graph.add_node('node2', 'rootNode')
graph.add_node('node3', 'node1')
graph.add_edge('node2', 'node3')


ftrails = findTrails(rootNode, 'rootNode', 'node3', type='BN')
ftrails.print()

# =============== OUTPUT ===============
[['rootNode', 'node1', 'node3'], ['rootNode', 'node2', 'node3']]
```

1.6.4 3. Random Graphs

```
from pgm.helpers.misc import GenerateRandomGraph

graph = GenerateRandomGraph(10, type='BN', skeleton_only=True).Graph
graph.print(graph.rootNode)

# =============== OUTPUT ===============
node: rootNode, children: ['node1', 'node5'], parents: []
node: node1, children: ['node2', 'node4'], parents: ['rootNode']
node: node5, children: ['node7', 'node8'], parents: ['node3', 'node4', 'node3',
↪'rootNode']
node: node2, children: ['node3', 'node6', 'node9', 'node9'], parents: ['node1']
node: node4, children: ['node5', 'node7'], parents: ['node3', 'node1']
node: node7, children: ['node8'], parents: ['node4', 'node5']
node: node8, children: [], parents: ['node7', 'node5']
node: node3, children: ['node4', 'node5', 'node5'], parents: ['node2']
```

(continues on next page)

(continued from previous page)

```
node: node6, children: [], parents: ['node2']
node: node9, children: [], parents: ['node2', 'node2']
```