IE531: Algorithms for Data Analytics
Spring, 2020
Programming Assignment 5: Gibbs-Sampling Implementation for Discrete-Time Markov Chains
Due Date: April 6, 2020

1 Introduction

When we use *Gibbs-Sampling* to compute the (*Stochastic*) *Probability Matrix* $\mathbf{P}$ of a *Discrete-time Markov Chain* that will yield a desired *Stationary Probability Distribution* $\boldsymbol{\pi}$. We assume the structure of the Markov Chain can be represented as an undirected-graph that is formed by the Cartesian-Product of d-many copies of the set $\{0, 1, \ldots, n-1\}$. Figure 1 shows the structure of the undirected graph, that represents the Markov Chain, when $d = 2$ and $n = 3$.

In this context, we have a desired *Stationary Probability Distribution* $\boldsymbol{\pi} \in \mathcal{R}^{n^d}$, and we need to find a Stochastic Matrix $\mathbf{P} \in \mathcal{R}^{n^d \times n^d}$, such that $\lim_{k \rightarrow \infty} \mathbf{P}^k$ results in a matrix where all rows of the product-matrix are identical to the row-vector $\boldsymbol{\pi}$. For purposes of checking if this is indeed the case, it would make sense to attach a number to each state (the *lexicographic-index*) – this number is shown in red, alongside each state in figure 1 . This index will help with the identification of the relevant row/columns of the stochastic matrix $\mathbf{P}$ and the probability vector π

The algorithm for the assignment of values to the entries in $\mathbf{P}$ can be found in the text (or, in my lectures). Keep in mind that the illustrative example in figure 4.3 of the text has numerical errors[1] – but the method/algorithm is fine.

You are going to write a Generic Gibbs-Sampling procedure that will work for any n, any d, and any valid Stationary Probability Distribution $\boldsymbol{\pi} \in \mathcal{R}^{n^d}$. I have provided a `hint.py` file, but you do not have to use it, I am expecting to see something along the lines of what is shown in figures 2, 3, 4 and 5.

These files can be submitted on Compass directly. Please do not e-mail them to the TA or me.

[1] All this stems from the fact that $\sum_{i=1}^{3} \sum_{j=1}^{3} P(i,j) = \frac{37}{24} > 1$. If this is a Stationary Distribution, this sum should be 1.

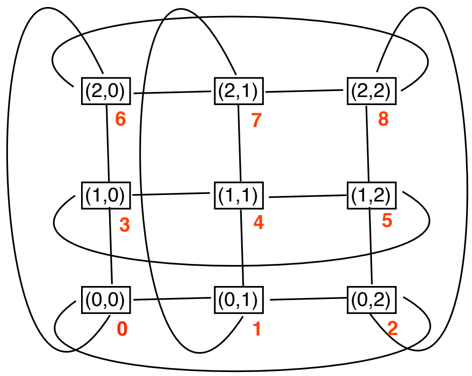

Figure 1: The undirected graph of the Markov Chain for Gibbs-Sampling, where $n = 3$ and $d = 2$. The lexicographic index of each state is shown in red along side each state.

```
In [7]: # Trial 1... States: {(0,0), (0,1), (1,0), (1,1)} (i.e. 4 states)
n = 2
dim = 2
a = generate_a_random_probability_vector(n**dim)
print("(Random) Target Stationary Distribution\n", a)
p = create_gibbs_MC(n, dim, a, True)
print ("Probability Matrix:")
print (np.matrix(p))
print ("Does the Probability Matrix have the desired Stationary Distribution?", np.allclose(np.matrix(a), np.matrix(a)

(Random) Target Stationary Distribution
 [0.3450820304758646, 0.1306515322224142, 0.35647172769378466, 0.16779470960793652]
Generating the Probability Matrix using Gibbs-Sampling
Target Stationary Distribution:
π (0, 0)  = π( 0 ) =  0.3450820304758646
π (0, 1)  = π( 1 ) =  0.1306515322224142
π (1, 0)  = π( 2 ) =  0.35647172769378466
π (1, 1)  = π( 3 ) =  0.16779470960793652
Probability Matrix:
[[0.6086 0.1373 0.2541 0.0000]
 [0.3627 0.3562 0.0000 0.2811]
 [0.2459 0.0000 0.5940 0.1600]
 [0.0000 0.2189 0.3400 0.4411]]
Does the Probability Matrix have the desired Stationary Distribution? True
```

Figure 2: Sample Output 1.

```
In [8]: # Trial 2... States{(0,0), (0,1),.. (0,9), (1,0), (1,1), ... (9.9)} (i.e. 100 states)
n = 10
dim = 2
a = generate_a_random_probability_vector(n**dim)
p = create_gibbs_MC(n, dim, a, False)
print ("Does the Probability Matrix have the desired Stationary Distribution?", np.allclose(np.matrix(a), np.matrix(a)*

Does the Probability Matrix have the desired Stationary Distribution? True
```

Figure 3: Sample Output 2.

```
In [12]: # Trial 3... 1000 states
n = 10
dim = 3
t1 = time.time()
a = generate_a_random_probability_vector(n**dim)
p = create_gibbs_MC(n, dim, a, False)
t2 = time.time()
hours, rem = divmod(t2-t1, 3600)
minutes, seconds = divmod(rem, 60)
print ("It took ", hours, "hours, ", minutes, "minutes, ", seconds, "seconds to finish this task")
print ("Does the Probability Matrix have the desired Stationary Distribution?", np.allclose(np.matrix(a), np.matrix(a)*

It took  0.0 hours,  0.0 minutes,  32.47895121574402 seconds to finish this task
Does the Probability Matrix have the desired Stationary Distribution? True
```

Figure 4: Sample Output 3.

```
In [13]: # Trial 4... 10000 states
n = 10
dim = 4
t1 = time.time()
a = generate_a_random_probability_vector(n**dim)
p = create_gibbs_MC(n, dim, a, False)
t2 = time.time()
hours, rem = divmod(t2-t1, 3600)
minutes, seconds = divmod(rem, 60)
print ("It took ", hours, "hours, ", minutes, "minutes, ", seconds, "seconds to finish this task")
print ("Does the Probability Matrix have the desired Stationary Distribution?", np.allclose(np.matrix(a), np.matrix(a)*

It took  1.0 hours,  7.0 minutes,  53.21802496910095 seconds to finish this task
Does the Probability Matrix have the desired Stationary Distribution? True
```

Figure 5: Sample Output 4.