Reinforcement Learning: Homework #4

Due on April 30, 2020 at 11:59am

Professor Ziyu Shao

Tianyuan Wu
63305667

Problem 1

Solution
To get the average number of 1s in good sequences, we can apply MCMC method.

1. Init the sequence to $(000\ldots0)_n$.
2. Generate a random number $k = [0, n-1]$, and filp the bit at index k.
3. If the new sequence is a good sequence, goto the new sequence; Otherwise stay at the same position.
4. Jump to step (2).

The length of the sequence is set to 100, then we can obtain the expected number of 1s in good sequence is 27.727864, which is very close to the real value.

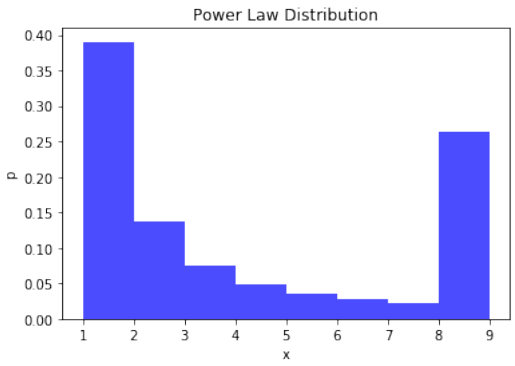
Problem 2

Solution
We can generate the power-law distribution (using MH method) by following steps:

1. Construct a Markov chain with
$$P = \{p_{ij}\} = \begin{cases} 1, & i == 1 \ and \ j == 1 \\ \frac{1}{2}, & otherwise \end{cases}$$
2. Start at a random state (or a given state), called s_0.
3. Determine a proposal state s' according to P.
4. Filp a coin with probability of landing head $p_H = \min\{1, \frac{p_{ij}\pi_i}{p_{ji}\pi_j}\}$
5. If the coin landing head, accept the proposal, otherwise reject the proposal.
6. Jump to step (2)

The simulation results are shown below, **the right-most bar of the histogram (i.e. The bar with subscript 9) means** $i \geq 9$

X	P
1	0.390231
2	0.137773
3	0.075105
4	0.048989
5	0.035108
6	0.026976
7	0.021640
8	0.017884
≥ 9	0.246295



Problem 3

Solution

We can solve Knapsack's problem (using MH method) by following steps:

1. Init the sequence to $s_i = (000\ldots0)_n$.
2. Generate a random number $k = [0, n-1]$, and filp the bit at index k, to get the new state s_{new}.
3. If the new sequence satisfies $\sum_i s_i w_i > w$, stay at current position.
4. Let $V_{s_i} = \sum_i s_i g_i$, filp a coin with probability of landing head $p = e^{\beta(V_{new} - V_{curr})}$
5. If the coin landing head, accept the proposal, otherwise reject the proposal.
6. Jump to step (2)

The simulation result for $m = 5$, $w = 10$, vector of worth $\{g_j\} = (6, 3, 5, 4, 6)$, vector of weight $\{w_j\} = (2, 2, 6, 5, 4)$ is: the best solution is 11001 (i.e. choose the item with index 1,2,5), and the maximum worth is 15.0.

Problem 4

Solution

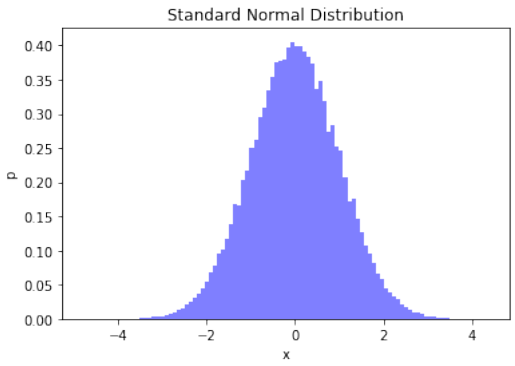
We can generate the standard normal distribution (using MH method) by following steps:

1. Generate a random number by $w = Unif(0, 1)$ as state w_0
2. Generate a new random number by $u = Unif(0, 1)$.
3. Filp a coin with probability of landing head $p_H = \min\{1, \frac{e^{-\frac{1}{2}u^2}}{e^{-\frac{1}{2}w^2}}\}$
4. If the coin landing head, accept the proposal (set $w = u$), otherwise reject the proposal (stay at w).
5. Jump to step (2)

Comparison:

- The Box-Muller method can generate 2 independent normal distributions at the same time, but if we only want 1 normal distribution, we also need 2 uniform distribution samples for one normal distribution sample $(\sqrt{-log(u_1)}cos(2\pi u_2))$.
- But by using MH method, we can only using 1 uniform distribution sample to generate a normal distribution sample.

The simulation results are shown below:



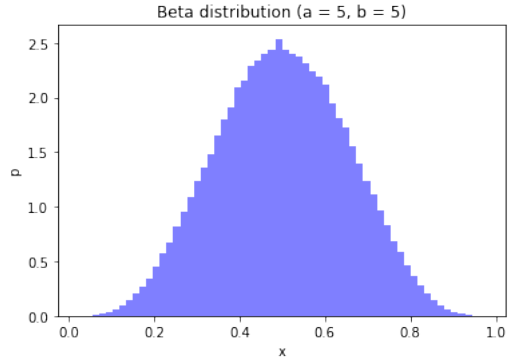
Problem 5

Solution

We can generate the beta distribution (using MH method) by following steps:

1. Generate a random number by $w = Unif(0, 1)$ as state w_0
2. Generate a new random number by $u = Unif(0, 1)$.
3. Filp a coin with probability of landing head $p_H = \min\{1, \frac{u^{a-1}(1-u)^{b-1}}{w^{a-1}(1-w)^{b-1}}\}$
4. If the coin landing head, accept the proposal (set $w = u$), otherwise reject the proposal (stay at w).
5. Jump to step (2)

The simulation result (with parameters $a = 5, b = 5$) are shown below:



Problem 6

Solution

We can solve the Normal-Normal conjugacy problem (using MH method) by following steps:

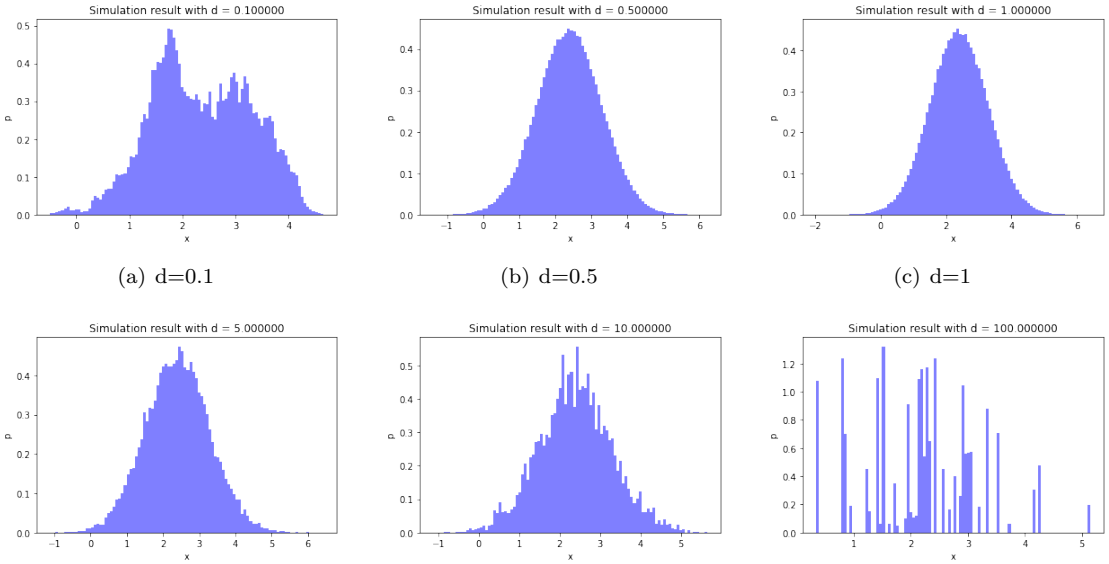
1. Generate a random number by $w = Norm(\mu, \sigma^2)$ as state w_0
2. Generate a new random number by $u = w + \mathcal{N}(0, d^2)$.

3. Filp a coin with probability of landing head

$$p_H = \min\{1, \frac{e^{-\frac{1}{2\sigma^2}(y-u)^2} e^{-\frac{1}{2\tau^2}(u-\mu)^2}}{e^{-\frac{1}{2\sigma^2}(y-w)^2} e^{-\frac{1}{2\tau^2}(w-\mu)^2}}\}$$

4. If the coin landing head, accept the proposal (set $w = u$), otherwise reject the proposal (stay at w).
5. Jump to step (2)

The simulation results are shown below:



(a) d=0.1 (b) d=0.5 (c) d=1

(d) d=5 (e) d=10 (f) d=100

Figure 1: Simulation results

Problem 7

Solution

We can generate the bivariate normal distribution (using Gibbs method) by following steps:

1. Init $x = 0$, $y = 0$
2. Update $x = \mathcal{N}(\rho y, 1 - \rho^2)$
3. Update $y = \mathcal{N}(\rho x, 1 - \rho^2)$
4. Jump to step (2)

The simulation result (with parameters $\rho = 0.6$ and $\rho = -0.6$) are shown below:

Figure 2: $\rho = 0.6$

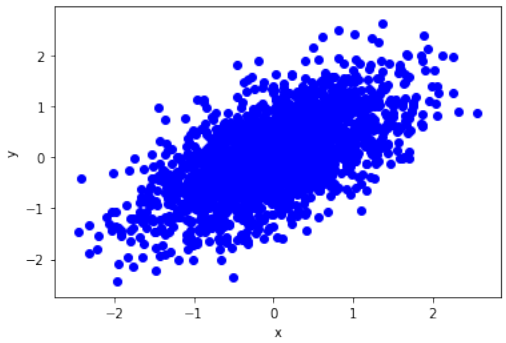
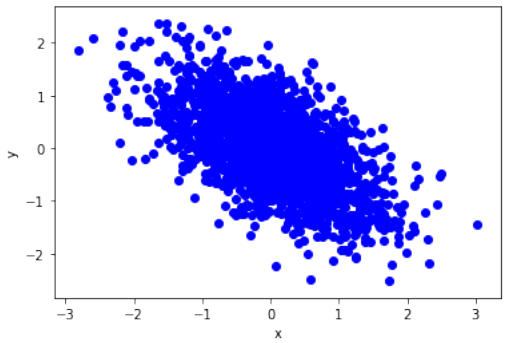


Figure 3: $\rho = -0.6$



Problem 8

Solution
We can solve the Normal-Normal conjugacy problem (using MH method or Gibbs sampling) by following steps:
MH:

1. Generate a random number by $w \sim Beta(a, b)$ as state w_0
2. Generate a new random number by $u \sim Beta(a, b)$
3. Filp a coin with probability of landing head

$$p_H = \min\{1, \frac{e^{\lambda u}(\lambda u)^x u^{a-1}(1-u)^{b-1}}{e^{\lambda w}(\lambda w)^x w^{a-1}(1-w)^{b-1}}\}$$

4. If the coin landing head, accept the proposal (set $w = u$), otherwise reject the proposal (stay at w).
5. Jump to step (2)

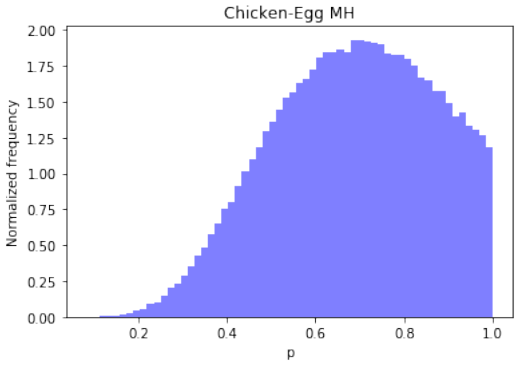
Gibbs:

1. Init n to some random value or given value

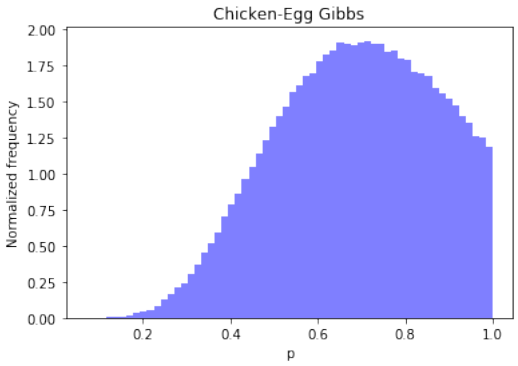
2. Generate a $p \sim Beta(x + a, n - x + b)$
3. Sample $y \sim Pois(\lambda(1 - p))$
4. Update $n = x + y$
5. Jump to step (2)

Comparsion:
- The MH method is actually a accept-reject method, if the simulation steps are not enough, it may reject many times and stay at the same state, also, it may need more calculations.
- The Gibbs sampling need less calculations, but the choice of systemic or random scan may effects the simulation results.

The simulation results are shown below:
MH method:
The parameters of p: Mean: 0.6848, Variance: 0.0320



Gibbs method:
The parameters of p: Mean: 0.6850, Variance: 0.0319

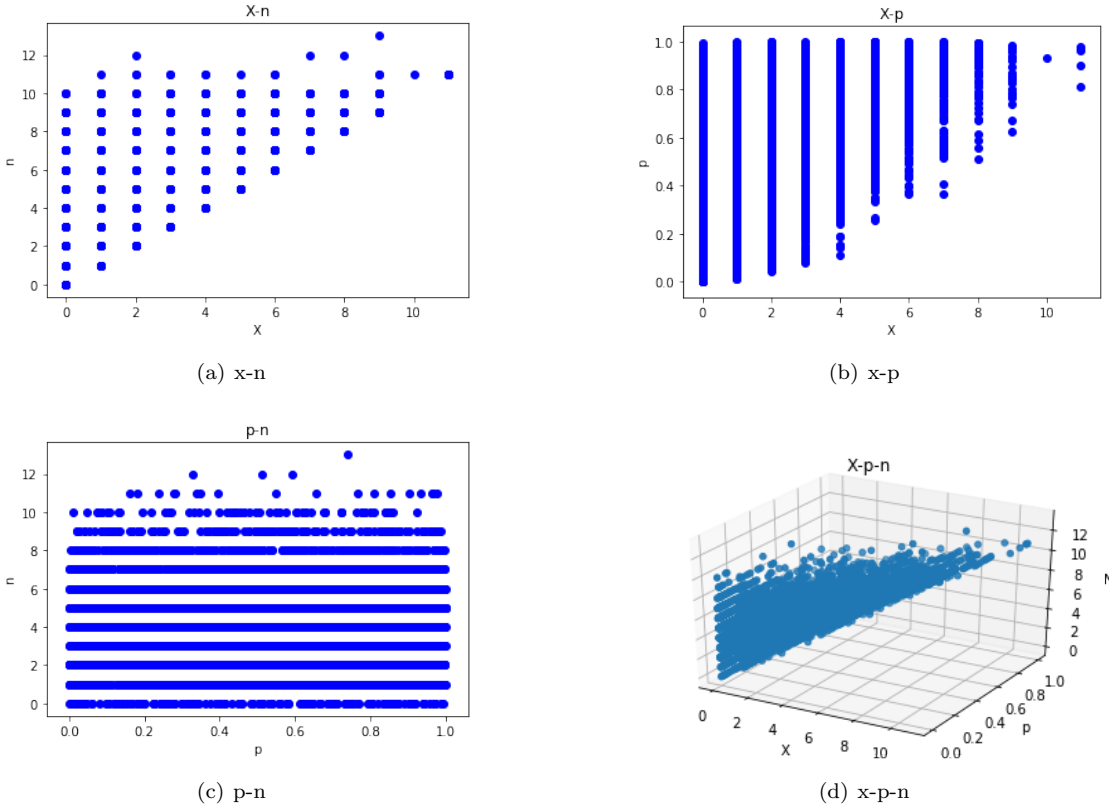


Problem 9

Solution
We can generate the three dimensional joint distribution (using Gibbs sampling) by following steps:

MH:

1. Init $(x_0, p_0, n_0) \leftarrow (1, 0.5, 2)$
2. Generate $x_m \sim Bino(n_{m-1}, p_{m-1})$
3. Generate $p_m \sim Beta(x_m + 1, n_{m-1} - x_m + 1)$
4. Generate $n_m = x_m + z$, where $z \sim Pois(4(1 - p_m))$
5. Jump to step (2)



(a) x-n

(b) x-p

(c) p-n

(d) x-p-n

Figure 4: Simulation results

Problem 10

Solution

(a) Discrete Time

First, we analysis the problem theoretically. The independent set problem is quite similar like the Knapsack problem, and we can solve it by similar ways. The basic idea is, use a binary sequence to represent the choice of independent set. $s[i] = 1$ means choose vertex i, and $s[i] = 0$ means not choose. Then, construct a irreducible Markov Chain, each sate X_i represent a binary sequence, and in each iteration, we filp a bit randomly, if the new sequence does not satisfy the independent set constraint, stay at the current state, otherwise go to the new state. Then, the stationary distribution π of this chain is an uniform distribution. Then, we can modify the chain to make it concentrate on states with larger size of independent set (i.e. larger

number of 1s), by MH method. That is, filp a coin with $p_H = e^{\beta(V_{new}-V_{curr})}$, where $V_{s_i} = \#\ of\ 1s\ in\ s_i$. Then if the coin landing head, accept the proposal, otherwise reject the proposal.

That idea can be described in details as:

1. Label all vertices by a number $0, 1, \ldots, n$
2. Init the sequence to $s_i = (000\ldots0)_n$, 0 means the vertex is not in the independent set, and 1 means it's in the set.
3. Generate a random number $k = [0, n-1]$, and filp the bit at index k, to get the new state s_{new}.
4. If the new sequence not satisfies the constraint of independent set (i.e. Exists 2 adjacent vertices are both "1" in new sequence), stay at current position.
5. Let $V_{s_i} = \#\ of\ 1s\ in\ s_i$, filp a coin with probability of landing head $p = e^{\beta(V_{new}-V_{curr})}$
6. If the coin landing head, accept the proposal, otherwise reject the proposal.
7. Jump to step (3)

We do simulation 5×10^5 steps, and we can observe the maximum size of the independent set is 8, we run it several times, and some of the simulation results are shown below:

The choice of maximum independent set is: 000101001010010010010101, with size = 9
The choice of maximum independent set is: 010000011010101010100100, with size = 9

The result is the same as the optimal solution (size=9).

(b) Continuous time
The basic idea is similar to the discrete time solution, the only difference is, the time is continuous. If the chain at state s_i, and a proposal state s_j. Then the transfer time $t_{trans} \sim f(V_{s_i})$, where f is a distribution with parameter V_{s_i}. That is, we want the chain stay at high value states longer, and spend little time on states with lower value. Here we use $t_{trans} \sim Pois(V_{s_i})$
That idea can be described in details as:

1. Consider $G(V, E)$ is the graph that we want to find the independent set.
2. Let X_0 is an arbitrary independent set of G (we can choose a vertex as X_0 in practice).
3. Then we compute X_{i+1}, first we choose a vertex v uniformly at random from V
4. If $v \in X_i$, then set $X_{i+1} = X_i \setminus v$ with probability $min(1, \frac{1}{\lambda})$, otherwise set $X_{i+1} = X_i$
5. If $v \notin X_i$, and adding v to X_i still gives an independent set, then set $X_{i+1} = X_i \cup v$, with probability $min(1, \lambda)$, otherwise set $X_{i+1} = X_i$
6. Jump to step (3)

We do simulation 5×10^5 steps, and we can observe the maximum size of the independent set is 8, we run it several times, and some of the simulation results are shown below:

The choice of maximum independent set is: 000101101000001010010101, with size = 9

Which is the same as the DTMC result.

(c) Comparison

- DTMC:
  - Pros: This method make the chain stay at states with high values with higher probability. It's easy to implement and can converge to the optimal value fast (if β is set properly).
  - Cons: Due to the extremely large state space (2^24 in this problem), if the simulation steps are not enough, it may be "locked" on some sub-optimal states .

- Continuous Time:
  - Pros: The time is continuous, and the time it stay on some state is determined by the value of the state. It may have better performance than DTMC for it will not "locked" on some sub-optimal state.
  - Cons: Due to the time is continuous, it's hard to implement, and it may converge slower than DTMC.