Bayesian parameter synthesis of Markov population models

Master Project on Modeling of Complex, Self-organizing systems

Huy Phung
University of Konstanz

July 15, 2020

Contents

Abstract

We study the collective behavior of a bee colony. Each bee in a colony possibly stings after observing a threat in the surrounding environment, and warn other bees by releasing a special substance, pheromone. By sensing the pheromone released in the environment, other bees in the colony may also sting. However, since stinging leads to the termination of an individual bee, it reduces the total defense capability as well. With parametric Discrete-time Markov chain as the model, we study how the actions of a single bee change with regarding to the colony size of and pheromone amount. This project propose and discuss Bayesian methods to estimate parameters of bees population models. It shows that the proposed methods is scalable and can deliver estimations of model parameters.

1 Preliminaries

1.1 Discrete time Markov Chain

Assume that each bee in a colony decides its next action (to sting or not to sting) based only on the current state of the environment, and the number of bees who sting or not sting can be modeled as a Markov process. To reduce the complexity of the model, we make another assumption that the states of the bees colony are observed after uniform time duration, hence the model is of discrete-time.

1.1.1 Definition

Definition 1.1 (Discrete Time Markov Chain)**.** A Discrete Time Markov Chain (DTMC) is a tuple $(S, \mathbf{P}, S_{init}, AP, L)$ [1]

- S is a countable non-emty set of *states*
- $\mathbf{P} : S \times S \to [0, 1]$ is the *transition probability* function, s.t

$$\sum_{s' \in S} \mathbf{P}(s, s') = 1$$

- $S_{init} : S \to [0, 1]$ is the *initial distribution*, s.t

$$\sum_{s' \in S} S_{init}(s') = 1$$

- AP is a set of *atomic propositions*
- $L : S \to 2^{AP}$ is the labelling function on states.

The transition probability function in DTMC is given as a stochastic matrix $\mathbf{P}$, which satisfies the following properties

- $\mathbf{P}$ is a square matrix.
- $\mathbf{P}_{i,j} \in [0, 1]$.
- $\sum_j \mathbf{P}_{i,j} = 1$

Definition 1.2 (Strongly Connected Component)**.** Let $\mathcal{M} = (S, \mathbf{P}, S_{init}, AP, L)$ a DTMC. A subset $S' \subset S$ is strongly connected if and only if for every pair $s_1, s_2 \in S'$ there is a path between s_1 and s_2 which consists of only of state in S'. If there exist no $S'' \subseteq S$, such that $S \subset S''$ and S'' is strongly connected, then S' is a *Strongly Connected Component*, or *SCC* in short.

Definition 1.3 (Bottom Strongly Connected Component)**.** Let $\mathcal{M} = (S, \mathbf{P}, S_{init}, AP, L)$ a DTMC and $S' \in S$ a Strongly Connected Component. S' is also a *Bottom Strongly Connected Component*, or *BSCC* for short, if and only if there exist no state $s \in S$
S' that is reachable from any state in S'.

1.1.2 Parametric Discrete-time Markov Chain

In order to generalize the stochastic matrix to encompass unknown information of the system, we introduce *parametric DTMC*. Let $\theta = (\theta_1, \ldots, \theta_n) \in [0, 1]^n$, we represent each element in matrix $\mathbf{P}$ as a polynomial function of θ. Let $\mathbf{Pol}_\theta$ be the set of polynomials $P : [0, 1]^n \to [0, 1]$. We define *parametric Discrete-Time Markov Chain*, or *pMC* for short, as follow:

Definition 1.4 (Parametric DTMC)**.** A parametric Discrete Time Markov Chain (pMC for short) is a tuple $(S, \theta, \mathbf{P}_\theta, S_{init}, AP, L)$

- S is a countable nonemty set of *states*
- θ is the set of model parameters
- $\mathbf{P} : S \times S \to \mathbf{Pol}_\theta$ is the *transition probability* function that map a transition relation between two states to a polynomial function of θ

- $S_{init} : S \to [0, 1]$ is the *initial distribution*, s.t

$$\sum_{s' \in S} S_{init}(s') = 1$$

- AP is a set of *atomic propositions*
- $L : S \to 2^{AP}$ is the labelling function on states.

A concrete assignment of θ on pMC induces a DTMC. In this project, we concern about the problem of data-informed estimation of model parameter θ.

1.1.3 Markov population model

In the scope of this project we concern *population* of a bee colony.

Definition 1.5 (Markov population model)**.**

1.1.4 Steady-state distribution

1.2 Bayesian inference

1.2.1 Bayesian parameter estimation

Let D be observed data. In statistical inference, we assume that the observed data has a probability distribution of unknown parameter θ, i.e $D \sim P(D|\theta)$. In frequentist approach, the estimation of θ based on long-run property, that is, given a large enough sample size, expected value of parameter estimation $\hat{\theta}$ is equal to θ. Therefore, frequentist approach requires to gather a large amount of data to deliver a close estimation $\hat{\theta}$. In Bayesian approach, we reuse the information *beliefs* gained from observed data to enhance the accuracy of the estimation of $\hat{\theta}$. The main advantage of Bayesian approach over frequentist approach is that it require less data to obtain an estimation $\hat{\theta}$.
The beliefs obtained from prior knowledge of model parameter θ is represented by *prior distribution* $\pi(\theta)$.
Also, we have probability distribution of observed data, given parameter θ, $P(D|\theta)$. This is also called *likelihood function.*
With Bayesian formula, we have

$$\pi(\theta|D) = \frac{P(D|\theta)\pi(\theta)}{\int_\theta P(D|\theta)\pi(\theta)d\theta}$$

$\int_\theta P(D|\theta)\pi(\theta)d\theta$ is called *marginal distribution*. $\pi(\theta|D)$ is called *posterior distribution*. Computing posterior distribution is the essential part of Bayesian inference, since it gives us the estimation of parameter θ.

1.2.2 Posterior conjugation

Conjugated posteriors are special cases of Bayesian inference, in which the prior and posterior distribution belongs to the same family of distribution. Conjugated posteriors give us significant benefits

1. Tractability: we have analytical form of posterior distribution.
2. Computationally effective: updating model parameter is of linear time to the dimension of parameter.

We consider two conjugated posterior: Binomial-Beta and Dirichlet-Multinomial

Lemma 1 (Binomial-Beta Conjugation)**.** Binomial distribution is conjugated to beta distribution.

Proof. The observed data $D = (x_1, \ldots, x_n)$ is sampled from $Binomial(k, \theta)$ function

$$P(D|\theta) = \prod_{i=1}^{n} \binom{k}{x_i} \theta^{x_i}(1-\theta)^{k-x_i}$$

The parameter θ is of $Beta(\alpha, \beta)$ distribution

$$\pi(\theta) = \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

We obtained:

$$\begin{aligned}\pi(\theta|D) &\sim P(D|\theta)\pi(\theta) \\ &\sim \theta^{\sum_{i=1}^n x_i}(1-\theta)^{nk-\sum_{i=1}^n x_i}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \theta^{\alpha-1+\sum_{i=1}^n x_i}(1-\theta)^{\beta-1+nk-\sum_{i=1}^n x_i}\end{aligned}$$

Thus, the posterior is $Beta(\alpha + \sum_{i=1}^n x_i, \beta + nk - \sum_{i=1}^n x_i)$ □

Generalize this conjugation, we also have Multinomial-Dirichlet conjugation.

Lemma 2 (Multinomial-Dirichlet Conjugation)**.** Multinomial distribution is conjugated to Dirichlet distribution.

Proof. The observed data $D = (x_1, \ldots, x_n)$ is sampled from $Multinomial(n; \theta_1, \ldots, \theta_n)$ function

$$P(x_1, \ldots, x_n | N, \theta_0, \ldots, \theta_n) = \frac{n!}{x_1! \ldots x_n!} \prod_{i=1}^{n} \theta_i^{x_i}$$

The parameter $(\theta_1, \ldots, \theta_n)$ is $Dirichlet(\alpha_1, \ldots, \alpha_n)$

$$\pi(\theta_1, \ldots, \theta_n) = \frac{1}{\mathbf{B}(\alpha_1, \ldots, \alpha_n)} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1}$$

We obtain

$$\begin{aligned} \pi(\theta_1, \ldots, \theta_n | D) &\sim P(D|\theta)\pi(\theta) \\ &\sim \prod_{i=1}^{n} \theta_i^{x_i} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} \\ &\sim \prod_{i=1}^{n} \theta_i^{\alpha_i - 1 + \sum_{i=1}^{n} x_i} \end{aligned}$$

Thus, the posterior is $Dirichlet(\alpha_1 + x_1, \ldots, \alpha_n + x_n)$ □

More detailed description in these cases can be found in [6] and [2]. We summarize the necessary results in the following table:

Likelihood	Prior	Posterior parameters
$Binomial(k, \theta)$	$Beta(\alpha, \beta)$	$\alpha' = \alpha + \sum_{i=1}^{n} x_i, \beta' = \beta + nk - \sum_{i=1}^{n} x_i$
$Multinomial(n; \theta_1, \ldots, \theta_n)$	$Dirichlet(\alpha_1, \ldots, \alpha_n)$	$\alpha'_i = \alpha_i + x_i, 1 \leq i \leq n$

1.2.3 Metropolis-Hastings algorithm

In case the posterior distribution has no analytical form or its analytical form is difficult to sample from directly, we use *Metropolis-Hastings* algorithm (*MH* in short).
Metropolis-Hastings algorithm is a *Monte Carlo Markov Chain* algorithm. In its essential, Metropolis-Hastings algorithm draws Using the MH algorithm, we can estimate the parameter by posterior mean, without knowing the analytical form of posterior distribution itself.

Algorithm 1 Metropolis-Hastings Algorithm, D is the observation data, θ is parameter

procedure METROPOLIS-HASTINGS(D, maxIteration)
    Select a proposal distribution $\pi(\theta)$
    Initialize θ
    while maxIteration not reached **do**
        $L \leftarrow P(D|\theta)$
        Draw θ' from the proposal distribution (transition kernel).
        $L' \leftarrow P(D|\theta')$
        if $\ln(L') - \ln(L) > 0$ **then**
            $\theta = \theta'$ (accept θ')
        else
            with probability ϵ accept θ' (avoiding local maxima)
        end if
    end while
end procedure

The likelihood function can be implemented as log-likelihood to avoid underflow error. Proposal distribution defines how do we proceed to the next parameter value on the parameter space; it can be of any distribution family.

1.2.4 Selection of prior distribution

In this project, since parameters are in $[0, 1]$, we select Beta distribution as proposal.
There are different methods to select the initial value of parameter P. In this project, we initialize P randomly. There are two advantages of using Markov Chain Monte Carlo in Bayesian inference:

1. Parameter transition only needs the computation of likelihood function. Therefore, Monte Carlo Markov Chain can be used in general Bayesian inference, in which we are not guaranteed to have an analytical form of posterior.

2. Specifically in Metropolis-Hastings algorithm, marginal distribution is cancelled out, thus make MH a computationally efficient algorithm.

However, MH algorithm also has a drawback; its convergence becomes slower as the dimension of parameter θ increases.

1.2.5 Bayesian Parameter Estimation

With posterior distribution $\pi(\theta|D)$ we estimate the parameter $\hat{\theta}$ using Bayesian posterior mean:

$$\hat{\theta} = \mathbf{E}[\theta] = \int_{\theta} \theta\pi(\theta|D)d\theta$$

In case we have samples from posterior distribution, for example when we use MH algorithm, the discrete form of posterior mean is used:

$$\hat{\theta} = \mathbf{E}[\theta] = \sum_{\theta} \theta\pi(\theta|D)$$

Definition 1.6 (Bayesian Credible Set)**.** Set C is a $(1\alpha)100\%$ credible set for the parameter θ if the posterior probability for θ to belong to C equals (1α).

$$P(\theta \in C|D) = \int_{C} \pi(\theta|D)d\theta = 1 - \alpha$$

In this project, we use 0.95 credible set, i.e $\alpha = 0.05$

Definition 1.7 (Highest Posterior Density credible set)**.** Highest Posterior Density $(1 - \alpha)100\%$ credible set (HDP for short) is the interval with minimum length over all Bayesian $(1 - \alpha)100\%$ Credible Set.

In this project, the HPD is calculated using PyMC3 library [5]

2 Related works

Markov chain and
Bayesian inference
Bees modeling

3 Modeling of bees colony

3.1 Biological description

We consider the collective action of a bee colony. Each bee in a colony could possibly sting after observing a threat in the surrounding environment, and

warn other bees by releasing pheromone. By sensing the pheromone released in the environment, other bees in the colony may also sting. Since stinging leads to the termination of an individual bee, it reduces the total defense capability as well. We studies how the actions of a bee changes with regarding to its surrounding the environment. There are 3 assumptions on the system:

1. Each bee release an unit amount of pheromone immediately after stinging.
2. A bee dies after stinging and releasing pheromone. In the other words, no bee can sting more than once.
3. Stinging behaviour only depends on the concentration of pheromone in the environment.

Under these assumption, a bee colony can be viewed as a set of agents (bees) interact with each other in a closed environment with the appearance of a factor *pheromone*. Each agent in the colony observe the following two factors of the environment: (1) amount of pheromone and (2) number of other agents. Afterward, the agent has probability to commit an action, namely *sting*. The agent is eliminated from environment after stinging.

3.2 Markov population models for bee colony

Assume that we have a colony of n bees initially. As aforementioned, an individual bee is terminated after it stings. Thus, at the end of experiment, the number of bees is $n' \in \{0, 1, \ldots, n\}$. We model the bee colony with a DTMC $\mathcal{M} = (S, \mathbf{P}, S_{init}, AP, L)$, such that

- $|S_{init}| = 1$
- There exists $n + 1$ BSCCs $B_0, \ldots, B_n \in S$. These BSCCs encode the population at the end of the experiment.

In a bee colony, we can also ob

- Let $P(B_i)$ be the probability of the

Semantics of Markov population models for bees colony are developed by [3]. In this report we summarize three models of bees population

1. Synchronous model

2. Asynchronous model

3. Semi-synchronous model

In the scope of this project we do not analyze the semantics of each model. Instead, the inference methods presented only uses the polynomial function of the models' steady-state distribution.

Asynchronous model

In asynchronous experiment we assume that there is almost improbable for two bees to sting simultaneously, and any bees release pheromone immediately after its death. By that observation we can assume that each bee sting at different level of pheromone.

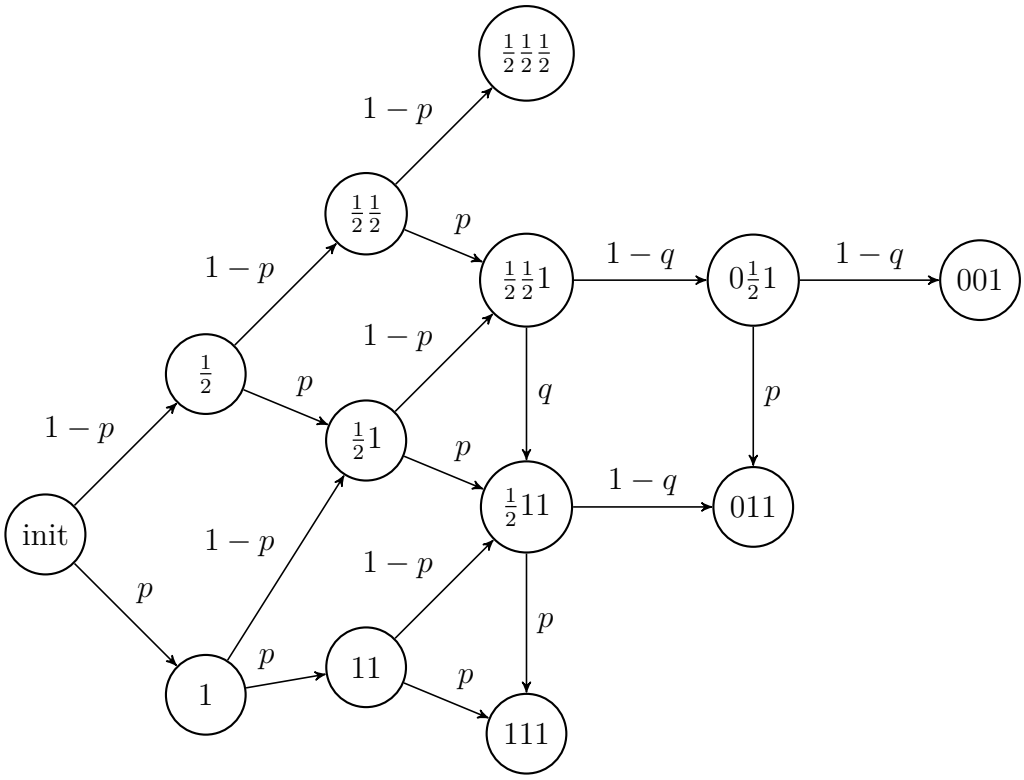

Figure 1: Example asynchronous model of 3 bees, 2 parameters

Synchronous model

In *fully synchronous* experiment we assume that the number of stinging bees is only counted after a fixed amount of time, so that without loss of generality we can assume the pheromone diffuse almost immediately among the bee colony, and each bee decide to sting or not to sting immediately after sensing the pheromone concentration.

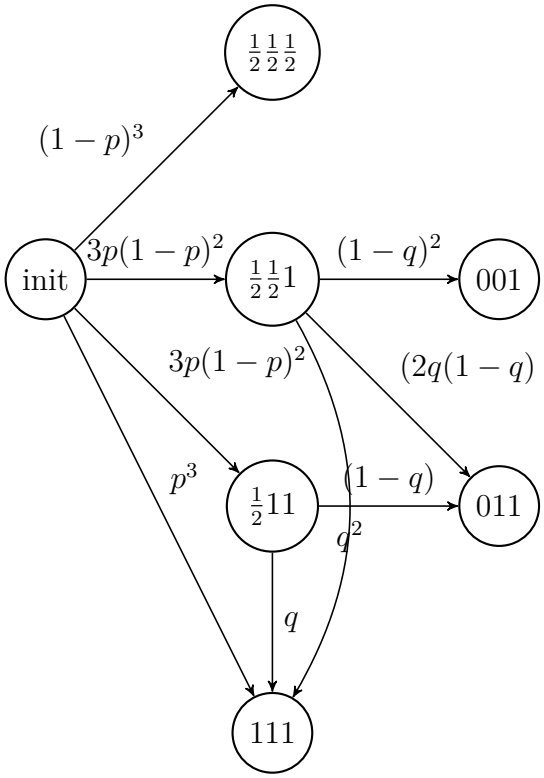

Figure 2: Example synchronous model of 3 bees, 2 parameters.

Semi-synchronous model

In *Semisynchronous model*, we assume that the behaviour is initally synchronous. That means, at the initial states we do synchronous update. From all succeeding states from initial states, the updates are of asynchronous semantics.

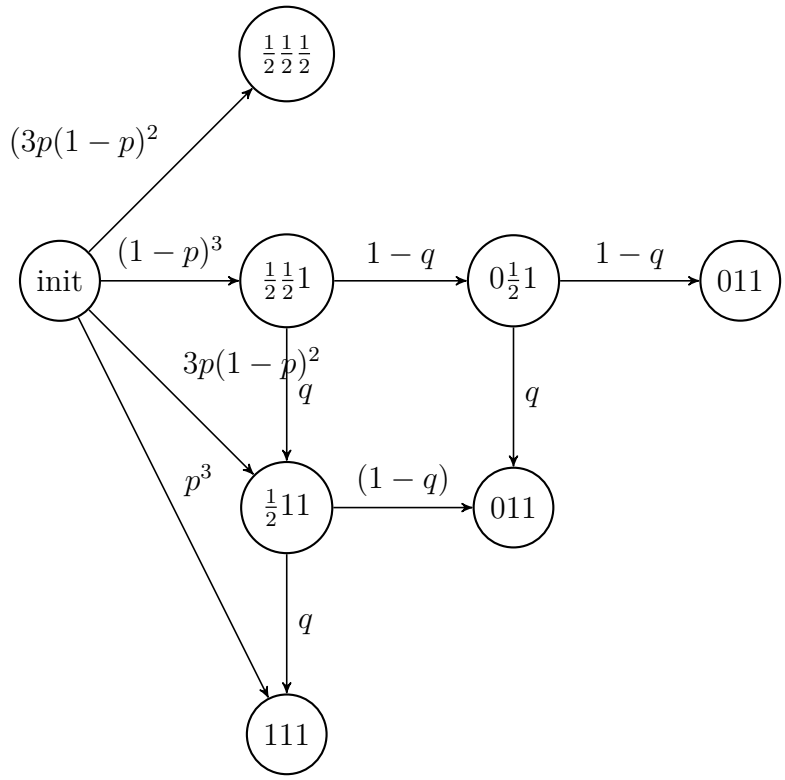

Figure 3: Example semisynchronous model of 3 bees, 2 parameters

For a population of n bees, all three types of models share two properties:

1. Has exactly one initial state, $|S_{init}| = 1$. This assumption is obvious, since at the beginning of the experiment, all bees are alive.

2. Has $n + 1$ tSCCs $(tSCC_0, \ldots, tSCC_n)$. This is because the number of dead bees cannot exceed the population size. It also follows that

$$\sum_{i=0}^{n} P(FG \quad tSCC_i) = 1$$

3.2.1 Linear model

asdf

3.2.2 Steady-state distribution

asdf

4 Bayesian parameter synthesis of Markov population model

4.1 Inference of steady state distribution

Let Using the Multinomial-Dirichlet conjugation, we can infer the steady state distribution as we observe new experiment data $D = (a_0, \ldots, a_n)$, with a_i is the number of experiment in which the population in steady-state is i

Algorithm 2 Estimation of steady-state distribution given a sample S

1: **procedure** ESTIMATE $p_i = P(tSCC_i)(S)$
2: Initialize $\alpha_0 = \alpha_1 = \ldots = \alpha_n = 1$
3: Initialize $p_0 = p_1 = \ldots = p_n = 0$
4: Update $\alpha_i = \alpha_i + a_i, 1 \leq i \leq n$
5: Update $p_i = \alpha_i / \sum_{i=1}^{n} \alpha_i, 1 \leq i \leq n$
6: **end procedure**

As later shown in the implementation with synthetic data, the Bayesian estimation of steady-state distribution converges quickly to the true parameter used for data synthesis. This method does not directly estimate the pMC model parameters. However, it can support the parameter estimation presented [3] to estimate the model parameters with narrower credible set.

4.2 Evaluation of steady -state distribution

4.3 Inference of model parameters with rational functions

We can also use Bayesian inference to estimate pMC model parameters directly. Given a pMC $\mathcal{M}$ with $\theta = (\theta_1, \ldots, \theta_k)$ as its model parameters. Since $\theta_i \in [0, 1]$, we can assume that θ_i are of $Beta(\alpha, \beta)$ distribution. Note that α, β are hyperparameters and must be selected manually. Also note that $\theta_1, \ldots, \theta_k$ are not simplex, thus it is not possible to use Dirichlet prior.
In this method, since there is no possible use of posterior conjugation, thus the hyperparameters estimation is hard and not in the scope of this project. Let $f_i = P(FG \quad tSCC_i)$ are polynomial functions of steady-state distribution, we denote $\epsilon_i = f_i(\theta)$ as evaluations of the polynomial function

with a concrete assignment of θ. The steady-state distribution has then $Multinomial(N, \epsilon = (\epsilon_0, \ldots, \epsilon_n))$ with N is the sample size of experiment data D. The posterior distribution has the following form:

$$\pi(\theta|D) \sim Multinomial(N, (\epsilon_0, \ldots, \epsilon_n))\pi(\theta_1) \ldots \pi(\theta_k)$$

As the posterior $\pi(\theta|D)$ has no analytical form, we use Metropolis-Hastings to sample from it.

Algorithm 3 Estimation of model parameters given a sample D

```
1: procedure ESTIMATE θ(S)
2:     Select hyperparameter α, β
3:     Select Beta(α, β) as proposal distribution (transition kernel) for
   θ_i, 1 ≤ i ≤ n
4:     Initialize θ
5:     while maxIteration not reached do
6:         Evaluate ϵ = (f_0(θ), ..., f_n(θ))
7:         L ← Multinomial(D|ϵ)
8:         Draw θ' from the proposal distribution.
9:         Evaluate ϵ' = (f_0(θ), ..., f_n(θ))
10:        L' ← P(D|ϵ')
11:        if ln(L') − ln(L) > 0 then
12:            θ = θ' (accept θ')
13:        else
14:            with probability ϵ accept θ' (avoiding local maxima)
15:        end if
16:    end while
17: end procedure
```

4.4 Inference of model parameter with Metropolis-Hastings algorithm

We can also use Bayesian inference to estimate pMC model parameters directly. Given a pMC $\mathcal{M}$ with $\theta = (\theta_1, \ldots, \theta_k)$ as its model parameters. Since $\theta_i \in [0, 1]$, we can assume that θ_i are of $Beta(\alpha, \beta)$ distribution. Note that α, β are hyperparameters and must be selected manually. Also note that $\theta_1, \ldots, \theta_k$ are not simplex, thus it is not possible to use Dirichlet prior.
In this method, since there is no possible use of posterior conjugation, thus

the hyperparameters estimation is hard and not in the scope of this project. Let $f_i = P(FG \quad tSCC_i)$ are polynomial functions of steady-state distribution, we denote $\epsilon_i = f_i(\theta)$ as evaluations of the polynomial function with a concrete assignment of θ. The steady-state distribution has then $Multinomial(N, \epsilon = (\epsilon_0, \ldots, \epsilon_n))$ with N is the sample size of experiment data D. The posterior distribution has the following form:

$$\pi(\theta|D) \sim Multinomial(N, (\epsilon_0, \ldots, \epsilon_n))\pi(\theta_1) \ldots \pi(\theta_k)$$

As the posterior $\pi(\theta|D)$ has no analytical form, we use Metropolis-Hastings to sample from it.

Algorithm 4 Estimation of model parameters given a sample D

procedure ESTIMATE $\theta(S)$
    Select hyperparameter α, β
    Select $Beta(\alpha, \beta)$ as proposal distribution (transition kernel) for $\theta_i, 1 \leq i \leq n$
    Initialize θ
    while maxIteration not reached **do**
        Evaluate $\epsilon = (f_0(\theta), \ldots, f_n(\theta))$
        $L \leftarrow Multinomial(D|\epsilon)$
        Draw θ' from the proposal distribution.
        Evaluate $\epsilon' = (f_0(\theta), \ldots, f_n(\theta))$
        $L' \leftarrow P(D|\epsilon')$
        if $\ln(L') - \ln(L) > 0$ **then**
            $\theta = \theta'$ (accept θ')
        else
            with probability ϵ accept θ' (avoiding local maxima)
        end if
    end while
end procedure

5 Implementation

The Bayesian inference methods are implemented in Python 3 and tested in the followin system configuration:

- Intel Core i5-8265U @ 1.60GHz

- 16GB RAM
- OpenSUSE Tumbleweed 20200427
- Anaconda 3 2019.10 for linux x86_64

The following experiment on Bayesian inference use synthetic data from semi-synchronous models of 2, 5 and 10 bees (in order to compare the performance with related work in [3]).

5.1 PRISM model and result processing

asdf

5.2 Metropolis-Hastings algorithm implementation

A script `parse_prism.py` is include in folder `models` to parse PRISM polynomial string to a Python class template. This is to avoid calling expensive function `eval()` on runtime. Source code structure:

- `docs` folder contains final report
- `impl` folder contains python implementation and PRISM results parsed to Python classes
- `examples` folder contains jupyter notebook examples.

5.3 Class design

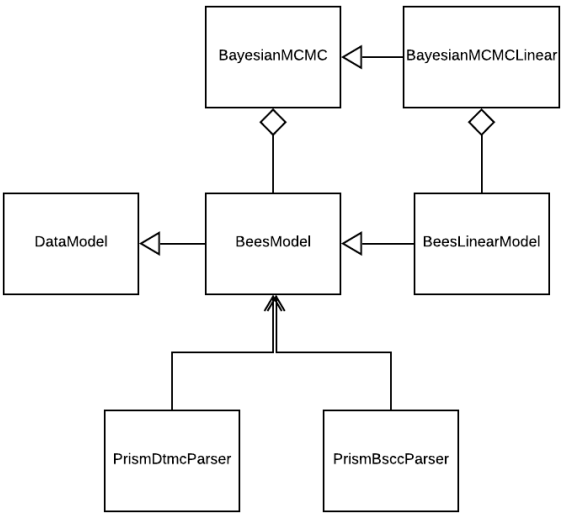

Figure 4: Class diagram of the project

6 Benchmark

6.1 Data synthesis

The data synthesis for n bees population is conducted as follow:

- Select a pMC model $\mathcal{M}$ to model the population of . Let $\bar{p} = p_1, \ldots, p_n$ is the model parameter vector.
- Assign a concrete value for $\bar{p}$, called $\bar{p}_{true}$
- From $\mathcal{M}$, deliver the rational functions of $f_i(\bar{p}) = P(FG \quad tSCC_i)$, $0 \leq i \leq n$. (using PRISM Model Checker [4])
- Evaluate concrete value $\theta_i = f_i(\bar{p}_{true})$
- Draw sample S from multinomial distribution $Multinomial(N, \theta = (\theta_0, \ldots, theta_n))$
- S is the synthetic result for the experiment 3.1

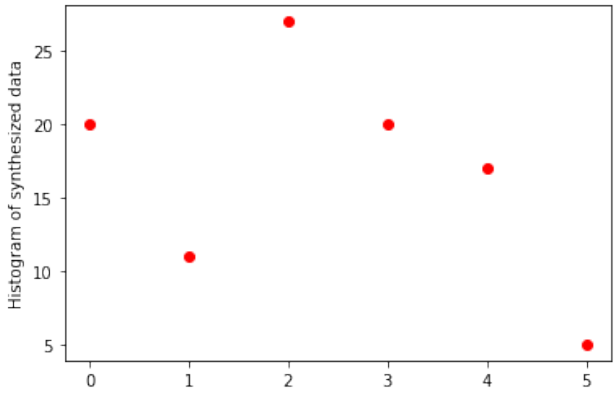

Figure 5: Steady state data synthesis, semi-synchronous model of 5 bees population, model parameters $(0.3, 0.25, 0.35, 0.45, 0.5)$

6.2 Evaluation measures

In order to evaluate the accuracy of parameter point inference, we have to select a distance measure from estimated parameter to true parameter. In this project we use two measures:

Definition 6.1 (RMSE)**.** Let $\theta = (\theta_1, \ldots, \theta_n)$ and $\hat{\theta} = (\hat{\theta_1}, \ldots, \hat{\theta_n})$ be vectors of n real numbers, the Root Mean Square Error between p and $\hat{p}$ is defined as follow:

$$RMSE(\theta, \hat{\theta}) = \sqrt{\frac{\sum_{i=1}^{n} (\theta_i - \hat{\theta_i})}{n}}$$

6.3 Comparison of DTMC sampling and rational functions evaluation

Given a bee population model $\mathcal{M}$, we have two methods to obtain steady-state distribution:

6.4 Inference of steady state distribution

This experiment is conducted on semisynchronous model of 5 bees. The polynomial functions of steady-state distribution is generated by PRISM and

parsed into Python 3 source code (`semisync_5bees.py`). The source code for this experiment is on Jupyter Notebook file `visualization.ipynb`

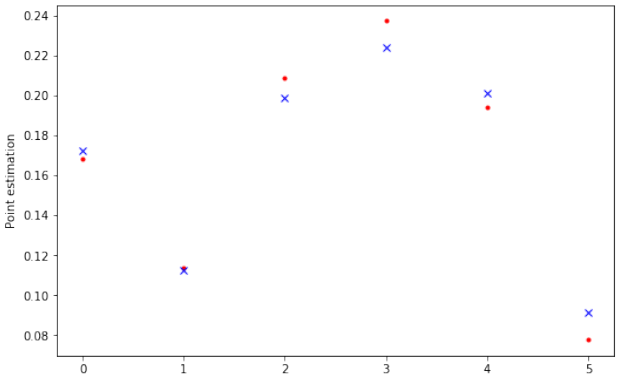

Figure 6: Bayesian estimation of steady state distribution. Blue cross is true steady state evaluation.

In order to compare with the work by [3], we also visualize the interval calculated by the following formula:

$$\theta_i \pm (z_{\alpha/2}\sqrt{\frac{\theta_i(1-\theta_i)}{N}})$$

where N is the sample size. Experiment configurations:

- true model parameter: $[0.3, 0.25, 0.35, 0.45, 0.5]$
- tSCC distribution evaluation: $[0.1680, 0.1139, 0.2089, 0.2371, 0.1940, 0.0778]$

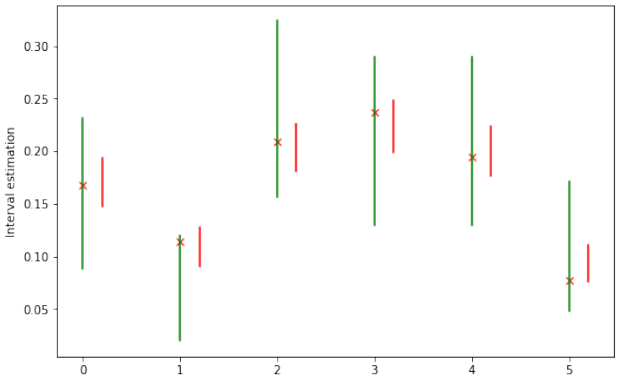

Figure 7: Comparision of Bayesian Highest Posterior Density 95% (red bar) and interval estimated using method in [3] (green bar). Red cross is true steady state evaluation.

6.5 Inference of model parameters with rational functions

This experiment is conducted in 2, 5, and 10 bees population models of 2, 5, and 10 parameters respectively. All models are semi-synchronous.

6.6 Inference of model parameters with DTMC sampling

7 Conclusion

The goal of this project is to experiment Bayesian approach on parameter inference of Markov population models. The results shows that Bayesian inference is efficient to deliver an estimation of model parameters. Furthermore, the implementation is more efficient than the work in [3] as it can work with higher number of parameters and requires less computational resources. However, there are still problems

1. MH algorithm converges slower as the number of parameter increases, thus requires longer chain of sampling the parameter space, which may increase the computational cost dramatically.

2. PRISM is still necessary to deliver polynomial functions, thus introduce an overhead, that PRISM itself is resource-hungry.

3. In this project, there is no Bayesian inference, which is designed to deal with the case when the polynomial functions are hard to synthesize, or impossible to synthesize.

However, we believe that the method can be developed further to overcome these limitations.

References

[1] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[2] Michael Baron. *Probability and statistics for computer scientists*. CRC Press, 2019.

[3] Matej Hajnal et al. "Data-Informed Parameter Synthesis for Population Markov Chains". In: *International Workshop on Hybrid Systems Biology*. Springer. 2019, pp. 147–164.

[4] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: Verification of Probabilistic Real-time Systems". In: *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. LNCS. Springer, 2011, pp. 585–591.

[5] John Salvatier, Thomas V Wieckiâ, and Christopher Fonnesbeck. "PyMC3: Python probabilistic programming framework". In: *ascl* (2016), ascl–1610.

[6] Stephen Tu. "The dirichlet-multinomial and dirichlet-categorical models for bayesian inference". In: *Computer Science Division, UC Berkeley* (2014).