infer_sde user manual

Cyril Galitzine

May 30, 2018

1 Introduction

infer_sde is a Python code that allows the inference of the distribution of the rates of a stochastic differential equation (SDE) from a single or multiple noisy observation(s) of its trajectory. It is based on a Markov chain Monte Carlo (MCMC) method which samples from the posterior rate distribution. The likelihood of the data is estimated via a particle filter method. Further details about the inference method can be found in [1]

It was initially created to infer the rates governing peroxisome dynamics [1]. In such situation, we are interested in inferring three rates k_d, k_f, γ and the standard deviation of the measurement error σ. The peroxisome count X_t at time t is governed by the following SDE:

$$dX_t = [k_d + (k_f - \gamma)\, X_t]\, dt + [k_d + (k_f + \gamma)\, X_t]^{1/2}\, dW_t \tag{1}$$

which corresponds to a birth-death-immigration (BDI) stochastic process. We simultaneously observe multiple *realizations* of this SDE by measuring the X_t time course in multiple cells. Each realization of the stochastic process is called a *replicate* in the following the biology terminology.

The code can readily be extended to other types of SDEs because of its modular object oriented design.

2 Running the code

2.1 Input files

2.1.1 sim.dat

This file is used to simulate data which is then stored the data.csv file. A typical sim.dat file is reproduced below:

```
Equation_sim BDI
Error_model_sim Normal
Nrep_sim 4
param_sim [7.75e-4,4.0e-5,4.0e-5]
error_sim [7.5]
Rate_heterogeneity_model_sim Homogeneous
Error_heterogeneity_model_sim Homogeneous
X0_sim [413]
Ntime_sim 120
T_sim 40000
```

The different lines of the file are described below:

- `Equation_sim`: Name of the SDE to simulate. Currently the following SDE are available:

  `BDI`: Eq. (1) with parameters k_d, k_f and γ (in that order). In this case the equivalent master equation is simulated via the Gillespie algorithm. This potentially avoid any error in the simulation of the SDE.

- `Error_model_sim`: Error model to use to simulate data. Currently the following models are available:

  `Normal`: Measured values x_{measured} are obtained from the true hidden values, x_{hidden} following $x_{\text{measured}} \sim \text{Normal}\left(\mu = x_{\text{hidden}}, \sigma = \texttt{error\_sim}\right)$. The standard deviation is assumed to be constant with time and the same for all replicates.

- `Nrep_sim`: Number of replicates to simulate
- `param_sim`: Value of the parameters used to simulate data. The order of the parameters has to correspond to that required by the SDE type.
- `Rate_heterogeneity_model_sim`: Heterogeneity model for the rates between replicates

  `Homogeneous`: The data of all replicates is generated with the exact same rates

  `Heterogeneous`: Rates vary between replicates following a gamma distribution for each rate. When this option is used `param_sim` does not specify directly the rates of the SDE but instead the mean and standard deviation of the gamma distribution of each rate. Assuming that the standard deviation is 10% of the rate mean, `param_sim` would become instead `[7.75e-4,7.75e-5,4.0e-5,4.0e-6,4.0e-5,4.0e-6]` of `[7.75e-4,4.0e-5,4.0e-5` for `Homogeneous` rate model.
- `Error_heterogeneity_model_sim`:

  `Homogeneous`: The data of all replicates is generated with the same error standard deviation.
- `X0_sim`: Initial value at time $t = 0$. This value should be specified for each `N_rep_sim` replicate. If it is not the initial value is determined via Poisson distribution with $\lambda =$ `X0_sim`.
- `Ntime_sim`: Number of time points (A constant Δt is used)
- `T_sim`: Overall simulation time (has to be in the same time unit as the rate specified with `param_sim`).

2.1.2 data.csv

data.csv contains the time course data that is used for the inference. It can be either generated by simulation or obtained through experimental measurement. The first few lines of typical data.csv file are reproduced below:

```
,replicate,t,x
0,0,0.0,420.0
1,0,336.1344537815126,427.0
2,0,672.2689075630252,421.0
3,0,1008.4033613445379,438.0
4,0,1344.5378151260504,456.0
5,0,1680.672268907563,427.0
6,0,2016.8067226890757,409.0
7,0,2352.9411764705883,417.0
8,0,2689.075630252101,416.0
9,0,3025.2100840336134,414.0
10,0,3361.344537815126,416.0
11,0,3697.4789915966385,400.0
12,0,4033.6134453781515,405.0
13,0,4369.747899159664,412.0
```

The first column corresponds to the index of the DataFrame is not important. It does not need to be specified. The following comma separated columns: `replicate, t, x` should always be present. The number of replicate starts at 0 and should always be specified even if there is only a single replicate.

2.1.3 inference.dat

The inference.dat file controls the inference procedure. A typical inference.dat file (the one used in example BDI_4rep_homogeneous_rates_parallel) is detailed below:

```
Simulate_data 0
Equation BDI
Ndisc 1
Error_model Normal
Nsamp 12000
Npart 1000
param [kd=3.0e-4,kf=3.0e-5,gamma=5.0e-5]
param_infer [1,1,1]
param_error [sigma_error1=4,sigma_error2=4,sigma_error3=4,sigma_error4=4]
param_error_infer [1,1,1,1]
Rate_heterogeneity_model Homogeneous
Error_heterogeneity_model Homogeneous
sd_MH 0.1
MH_step_scaling exponential
output_freq 20
```

- `Simulate_data`: 0: Read existing data.csv file (if it exists), 1: Simulate new data
- `Equation`: Name of the SDE to simulate. Currently the following SDE(s) are available:

  `BDI`: Eq. (1) with parameters k_d, k_f and γ (in that order).

- `Error_model`: Error model for the data. Currently the following models are available:

  `Normal`: Measured values x_{measured} are obtained from the true hidden values, x_{hidden} following $x_{\text{measured}} \sim \text{Normal}\left(\mu = x_{\text{hidden}}, \sigma = \texttt{error\_sim}\right)$. The standard deviation is assumed to be constant with time for each replicate.

- `Nsamp`: Number of accepted MCMC samples to calculate. MCMC sample proposals that are not accepted are not counted.
- `Npart`: Number of particles to use for the particle filter
- `param`: Initial starting value for the parameters at step 0. The parameters have to be in the correct order dictated by `Equation`.
- `param_infer`: Infer parameter of `param` 0/1 = Yes/No. In case param_infer=0 for a particular parameter, its value is assumed constant and equal to that specified in `param`.
- `param_error`: Initial starting value for the error parameter at step 0. The names specified for the error parameters do not matter. Typically one error parameter is inferred for each replicate such as here.
- `param_error_infer`: Infer parameter of `param_error` 0/1 = Yes/No. In case param_error_infer=0 for a particular parameter, its value is assumed constant and equal to that specified in `param_error`.
- `Rate_heterogeneity_model`: Heterogeneity model for the rates between replicates

  `Homogeneous`: Assume that all replicates are governed by the exact same rates.

  `Heterogeneous`: Assume that rates vary between replicates following a gamma distribution for each rate. When this option is used `param` does not specify directly the initial rate values of the SDE but instead the initial mean and standard deviation of the gamma distribution of each rate. Assuming that initially the standard deviation is 10% of the rate mean, `param_sim` would become instead `[7.75e-4,7.75e-5,4.0e-5,4.0e-6,4.0e-5,4.0e-6]` of `[7.75e-4,4.0e-5,4.0e-5` for `Homogeneous` rate model.

- `Error_heterogeneity_model`: Heterogeneity model for the measurement error between replicates

`Homogeneous`: Assumes that each replicate has a different error standard deviation. This terminology might be misleading but it used for consistency with the rate heterogeneity model. When this model is used, `param_error` needs to contain one error standard deviation for each replicate.

- `sd_MH`: Parameter controlling the step size of the Metropolis Hastings algorithm. It should be adjusted so that the average acceptance rate is around 0.3.
- `MH_step_scaling`: Metropolis Hasting stepping method. The following methods are available:

  `exponential`: New samples are generated following a log-normal distribution centered on the current value, i.e. $x^{n+1} = x^n \exp(Z)$ with $Z \sim \mathcal{N}\left(0, \sigma_{\text{MH\_step\_scaling}}\right)$. This stepping method ensures that each rate step is always properly scaled that the rates remains positive (rates are always defined to be positive).

2.2 Output files

2.2.1 out.dat

This file contains the accepted MCMC samples (rejected samples are not recorded) for each of the parameter listed in the inference.dat file. It also contains the overall likelihood of the data. The out.dat file obtained with the inference.dat file shown above looks like this: (in small fonts to show everything...)

```
iter kd kf gamma sigma_error1 sigma_error2 sigma_error3 sigma_error4 L
0.0000000000000000000e+00 3.2054073011677429 41e-04 2.8342542821711178 51e-05 5.2809021762677247 25e-05 4.1792746179815170 04e+00 3.5950008237830157 75e+00 3.6077743656066072 48e+00 4.0968776718231518 79e+00 -2.2041202
1.0000000000000000000e+00 3.1894959113848566 71e-04 2.9335804092259893 32e-05 5.4785720291663009 60e-05 4.9041978557527450 28e+00 3.6076821555437161 85e+00 3.7829852961157133 25e+00 4.2502753443571368 31e+00 -2.1917030
2.0000000000000000000e+00 3.4100042453214917 53e-04 2.8767209237567300 6e-05 4.6705605552829011 82e-05 4.9908949253182193 22e+00 3.1743307732701846 47e+00 3.2924759423927056 90e+00 4.0061536974714417 88e+00 -2.1199709
3.0000000000000000000e+00 3.8684743761341962 73e-04 2.9054210014291774 60e-05 4.4057387496900989 53e-05 5.8017274559905578 36e+00 2.9313937942559995 52e+00 3.3271878390224189 91e+00 3.3927262163377576 17e+00 -2.0761692
4.0000000000000000000e+00 3.4851989509707901 86e-04 3.3923974410632436 39e-05 4.8743835524713703 64e-05 6.0236829842909536 13e+00 3.2234903755456754 13e+00 3.2689009612150368 16e+00 3.4152516329769251 34e+00 -2.0313431
5.0000000000000000000e+00 3.0317637790267773 67e-04 3.2413324289436500 30e-05 4.4462493263500976 53e-05 6.2482627050576571 29e+00 3.1065586591444533 94e+00 2.8598089574606762 04e+00 3.7358162110639079 32e+00 -2.0316791
6.0000000000000000000e+00 3.1087109514668566 80e-04 3.5061068307923711 41e-05 4.1323339614801312 13e-05 6.5647051286465112 34e+00 3.2124026075456946 22e+00 2.6802823604842478 39e+00 3.8644621027023133 84e+00 -1.9890472
7.0000000000000000000e+00 3.3993138430005897 43e-04 3.6968239008383815 867e-05 4.0840732933896634 43e-05 7.5978698263201129 85e+00 4.0722444872434273 58e+00 2.8784984801832278 48e+00 3.9173490704675053 56e+00 -1.8880584
8.0000000000000000000e+00 3.0020888423089536 84e-04 3.3267823555692627 52e-05 3.3439099471303606 29e-05 7.8631160326132798 49e+00 4.7065881996171956 56e+00 2.8892209267246644 62e+00 3.7597656419639675 64e+00 -1.8880778
9.0000000000000000000e+00 2.9547109360788642 02e-04 3.5477886611518366 63e-05 3.5632022466064409 57e-05 8.2270713299338282 81e+00 4.7224093520465464 38e+00 2.9230401678038222 53e+00 3.9458904101259784 09e+00 -1.8699286
1.0000000000000000000e+01 3.2290930623677191 08e-04 3.9897326970502018 86e-05 4.4368199198051052 05e-05 7.9766153905986945 11e+00 5.0406076450982766 35e+00 3.1840903246276237 58e+00 4.1181434684363162 90e+00 -1.8401422
```

The columns correspond to the samples for rates specified in `parama` and `param_error` (in the same order). The last column is the log likelihood calculated for that particular set of parameters.

3 Running the code

3.1 Overall workflow

3.2 Simulating data with infer_sde_serial.py

To simulate data, remove any existing data.csv file and run the serial inference code: `python infer_sde_serial.py` . You should have all the *.py files in the directory where you execute this command as well as the following required input files: `sim.dat`, `inference.dat`. If you do not wish to perform inference simply stop the executing of the code after a few steps once you are sure that data.csv has been generated.

You can plot the trajectories that you just generated in the `data.csv` file by running `python plot_traj.py` which will produce a plot: `data.png`

It is also likely that you will have to modify the code in time_series.py to generate simulated data that conforms to your desired specifications This is because simulating data according to sim.dat can only produce a very narrow set of data types. For instance, it cannot generate replicates with different numbers of time points or overall times.

3.3 Running the serial inference code with infer_sde_serial.py

The inference code is executed by typing `python infer_sde_serial.py` You should have all the *.py files in the directory where you execute this command as well as the following required input files: `sim.dat`, `inference.dat` of course the data file `data.csv`. When the code runs, it will output for each accepted sample the value of the parameters and measurement errors, the log likelihood (L), the max log likelihood (called Lmin) and the average acceptance ratio AR. The value of `sd_MH` in `inference.dat` should be adjusted so that the acceptance ratio is around 0.3 (i.e. the optimum acceptance ratio for the Metropolis Hastings algorithm).

3.4 Running the parallel inference code with infer_sde_parallel.py

The inference code was parallelized across replicates using a python version of the MPI library mpi4py. This allows us to split the calculation of the overall log likelihood across multiple CPUs to speed up the sampling procedure (but only when multiple replicates are considered). The N_{rep} replicates are distributed as uniformly as possible across CPUs. Each CPU then calculates the sum of the log likelihoods of all the replicates that are assigned to it. The log likelihoods sums are then summed again across cpus to calculate the overall log likelihood. When using N_{CPU} to run the inference, one CPU (the master CPU) will not be performing calculations. As such the calculation will be distributed among the remaining $N_{\text{CPU}} - 1$ CPUs. For instance, if want to infer the rates for $N_{\text{rep}} = 4$ replicates and use $N_{\text{CPU}} = 5$, each CPU will calculate the likelihood of one replicate which leads to the optimal speedup. If $N_{\text{CPU}} = 4$ are used for the same 4 replicates, the replicates will be split as follows across CPUs: $\overbrace{\text{Rep1}}^{\text{CPU 2}} \mid \overbrace{\text{Rep2}}^{\text{CPU 3}} \mid \overbrace{\text{Rep3}, \text{Rep4}}^{\text{CPU 4}}$ The fourth CPU will calculate the likelihood of two replicates) while CPU 1, the master CPU, is not involved in the calculation of likelihoods. If $N_{\text{CPU}} = 3$, we would obtain the following: $\overbrace{\text{Rep1}, \text{Rep2}}^{\text{CPU 2}} \mid \overbrace{\text{Rep3}, \text{Rep4}}^{\text{CPU 3}}$. $N_{\text{CPU}} = 2$ should not be employed since the it will less efficient than the serial code the due to the master CPU.

To run the inference with, e.g. NCPUS = 4 the following should be executed:

```
mpirun -np 4 python3 infer_sde_parallel.py
```

The inputs and outputs of the parallel code are identical to those of the serial code.

3.5 Analyzing results with analyze_mcmc.py

The inference code (serial or parallel) produces an `out.dat` file with the posterior rate samples. Plots of the distribution of the rates can be obtained by running `python analyze_mcmc.py`. This produces density (*_density.pdf) and correlation (*_corr.pdf) plots for the inferred parameters. The sampling frequency of the parameters (`frequency` variable) in analyze_mcmc.py should be adjusted so that the consecutive samples are not too correlated which otherwise results in an inaccurate estimation of the densities.

4 Test cases

4.1 BDI (birth death immigration) SDE: $dX_t = [k_d + (k_f - \gamma)\, X_t]\, dt + [k_d + (k_f + \gamma)\, X_t]^{1/2}\, dW_t$

4.1.1 1 replicate, serial (EXAMPLES/BDI_1rep)

All the input, output files and results of the inference can be found in folder EXAMPLES/BDI_1rep. Data was simulated according to the following sim.dat for a single replicate:

```
Equation_sim BDI
Error_model_sim Normal
Nrep_sim 1
Rate_variation_model_sim None
param_sim [7.75e-4,4.0e-5,4.0e-5]
error_sim [6.0]
Rate_heterogeneity_model_sim Homogeneous
Error_heterogeneity_model_sim Homogeneous
X0_sim [413]
Ntime_sim 120
T_sim 40000
```

The serial inference code, `infer_sde_serial.py`, was run for 24 hours and the results were analyzed with `analyze_mcmc.py`.

4.1.2 4 replicates, Homogeneous rates, serial (EXAMPLES/BDI_4rep_homogeneous_rates_serial)

All the input, output files and results of the inference can be found in folder EXAMPLES/BDI_1rep. Data was simulated according to the following sim.dat for 4 replicates (homogeneous rates):

```
Equation_sim BDI
Error_model_sim Normal
Nrep_sim 4
Rate_variation_model_sim None
param_sim [7.75e-4,4.0e-5,4.0e-5]
error_sim [7.5]
Rate_heterogeneity_model_sim Homogeneous
Error_heterogeneity_model_sim Homogeneous
X0_sim [413]
Ntime_sim 120
T_sim 40000
```

The serial inference code, `infer_sde_serial.py`, was run for 24 hours and the results were analyzed with `analyze_mcmc.py`.

4.1.3 4 replicates, Homogeneous rates, parallel (EXAMPLES/BDI_4rep_homogeneous_rates_parallel)

The same data as for case BDI_4rep_homogeneous_rates_serial was used. The parallel inference code was run for 24 hours using 5 CPUs with `mpirun -np 5 python infer_sde_parallel.py`. Results were then analyzed with `analyze_mcmc.py`.

4.1.4 4 replicates, Heterogeneous rates, parallel (EXAMPLES/BDI_4rep_heterogeneous_rates_parallel)

All the input, output files and results of the inference can be found in folder EXAMPLES/BDI_4rep_heterogeneous_rates_parallel. Data was simulated according to the following sim.dat for 4 replicates (heterogeneous rates between replicates):

```
Equation_sim BDI
Error_model_sim Normal
Nrep_sim 4
Rate_variation_model_sim None
param_sim [2.75e-4,2.75e-5,3.0e-5,3.0e-6,5.0e-5,5.0e-6]
error_sim [7.5]
Rate_heterogeneity_model_sim Heterogeneous
Error_heterogeneity_model_sim Homogeneous
```

```
X0_sim [413]
Ntime_sim 120
T_sim 40000
```

References

[1] C. Galitzine, Pierre M. Jean Beltran, I. M. Cristea, and O. Vitek. Statistical inference of peroxisome dynamics. In Benjamin J. Raphael, editor, *RECOMB 2018*, pages 54–74. Springer, 2018.