Model Inference from Protein Time-Course in Hematopoietic Stem Cells

Pandu Raharja Rene Schoeffel

August 4, 2016

Abstract

Stochasticity of gene expression becomes apparent when studying the dynamics of single cells rather tan a population of cells. These fluctuations in gene expression in fact reveal more information about the underlying mechanisms of transcription and translation than one could obtain from population averages.

In this paper we developed a particle filtering algorithm to infer the parameters of stochastic models from single cell time series data. In particular, we apply this method to time-lapse microscopy data of two transcription factors (`Pu.1` and `Gata.1`) in blood stem cells. These transcription are thought to play a major role in stem cell differentiation.

Our results provided several insights into the dynamics of blood stem cells maturation and specifically in the single ell environment, we managed to gain several valuable insigts on the dynamics of the interaction between two transcription factors on the outcome of cell maturation.

1 Models

1.1 Particle Filtering

1.1.1 Particle

We use particle filtering in our simulation. Particle filtering is a family of methods that use utilize the concept of *particle*. A particle $\mathcal{K}$ is defined as a triple of trajectory X, parameter set θ and assumed model $\mathcal{M}$,

$$\mathcal{K} := (X, \theta, \mathcal{M})$$

Note that a trajectory is different from the experimental data $\mathcal{D}$. A trajectory is the result of the simulation done by applying the parameters onto our model. In functional notation we would assume the i-th value of the trajectory X_i to be a function of the i-th value of the trajectory, the model and the parameters,

$$X_i = f(X_{i-1}, \mathcal{M}, \theta)$$

1.1.2 Posterior

Our posterior describes the probability of having the trajectory X and parameter θ given the observation $\mathcal{D}$,

$$P(X, \theta|\mathcal{D})$$

We could understand this as the probability of having our simulation return a given set of values *and* having the parameter set θ given that we previously observed the experimental data $\mathcal{D}$. Using Bayes' Theorem we could further expand our posterior into an update rule,

$$P(X, \theta|\mathcal{D}) = \frac{P(\mathcal{D}|X, \theta)P(X, \theta)}{P(\mathcal{D})}$$

In our simulation we know that, to compute prior $P(\mathcal{D}|X, \theta)$, only knowledge about the trajectory of the simulation is needed. Hence, we could simplify our prior,

$$P(\mathcal{D}|X, \theta) = P(\mathcal{D}|X)$$

Not that the above equation inherently assumes that $\mathcal{D}$ is only directly dependent on X and X is in turn only directly dependent on θ. Incorporating this onto our update rule, and expanding the definition of $P(X, \theta)$ using chain rule, we get

$$P(X, \theta|\mathcal{D}) = \frac{P(\mathcal{D}|X)P(X|\theta)\pi(\theta)}{P(\mathcal{D})}$$

One of the interesting aspects of our is the fact that the i-th simulation result is only dependent on previous simulation, a property known as *Markov property*. We could thus rewrite the update rule as follow,

$$P(X, \theta|\mathcal{D}) = \frac{\prod_{i=0}^{N} P(\mathcal{D}_i|X_i) \cdot P(X_0) \cdot \prod_{l=1}^{N} P(X_{[t-1,ti]}|X_i, \theta) \cdot \pi(\theta)}{P(\mathcal{D})}$$

1.1.3 Parameters

During the simulation we assumes certain parameters that would influence the trajectory of the simulation. The parameter set θ could then mathematically be understood as P-tuple containing all the parameters that are assumed in the simulation,

$$P := (P_0, P_1, \ldots, P_P)$$

1.1.4 Prior

Our prior $\pi(\theta)$ could be expanded by assuming the independent of each parameter θ_i within the parameter set θ,

$$\pi(\theta) = \prod_{i=1}^{P} \pi(\theta_i)$$

Specifically for this simulation, we assume our prior to be Gamma distributed with the parameters α and β,

$$\pi(\theta) = \prod_{i=1}^{P} Ga(\theta_i, \alpha_i, \beta_i)$$

We used Gamma distribution as our prior due to the fact that a conjugate prior of a Gamma distributed random variable is also Gamma distributed.

1.2 Simulation Process

The simulation is run in roughly following high level steps:

1. Initialization of parameters θ.
2. Input of data $\mathcal{D}$.

3. Particle filtering routine:

   (a) Generation of initial particles for step i

   $$Ki := (K_{i1}, K_{i2}, \ldots, K_{im})$$

   (b) Simulation run of each particle K_{ij}

   (c) Weighting of each particle. The weight is a function of probability of observing the data given the simulation result.

   $$w_i^k = P(D_i|X_i^k) = \mathcal{N}(D_i|X_i^k)$$

   (d) Parameter update for every K,

   $$\theta^k \propto P(\theta|X_{[to,ti]}^k)$$

4. Model comparison. This could be done by measure such as AIC, BIC, Bayes Factor etc.

1.2.1 Particle Generation

As mentioned before, A particle $\mathcal{K}$ is defined as a 3-tuple of trajectory X, parameter set θ and assumed model $\mathcal{M}$. For a specific model, j-th particle at i-th time particle is then just a tuple of trajectory and parameter set $k_{ij} = (x_i, \theta_{ij})$.

For initial time, a particle could be constructed by combining initial parameters, which were arbitrarily defined by a pre-defined prior, and initial trajectory value X_0. There are three ways to define the initial trajectory value. First is to take 0 as initial value. Second is to take the first measurement data $\mathcal{D}_0$. Third is to model initial data as normal distributed instances around $\mathcal{D}_0$, i.e. $X_0 \propto \mathcal{N}(\mathcal{D}_0)$.

M particles would then be created and for each iteration of the simulation with each particle having weight which calculated from previous iteration w_{i-1}^k. N particles would then repeatedly chosen off the M particles. The draw is done by first drawing a random $z \propto \mathcal{U}(0, 1)$. A particle k_{ij} is chosen with j denoting the smallest index so that the sum of all weight of particles with index smaller than j is larger than z, i.e.

$$\sum_{l=1}^{j} w_{i-1}^{l} \geq z$$

1.2.2 Simulation Run

Simulation is done using Gillespie's SSA algorithm. There are possible improvements that we thought might be useful in our project such as τ-leaping and Bayesian Approximation Method.

1.2.3 Particle Weighting

Upon the completion of all simulations within one iteration, the quality of each simulation (and in turn, the particle) will be quantified. This quantification is implied in the weighting of the particle at the next iteration w_i^k. We assume the Gaussian distribution of particle around the measurement time at time $t = i$. Using this assumption, we can quantify our weight in following manner,

$$w_i^k = P(\mathcal{D}_i|x_i^k) = \mathcal{N}(\mathcal{D}_i|x_i^k)$$

To generate particle for next iteration we utilize, however, the normalized particle weight,

$$w_j^k = \frac{w_j^k}{\sum_{l=1}^{M} w_j^l}$$

1.2.4 Gamma Distribution

1.2.5 Parameters Update