Grn Inference

Jie Meng

April 2019

1 Main Idea

Simulating expression data with Gene Regulatory Network and iterate network to generate data which are more close to real data is a typical way to infer a GRN. However, Biological development process follows the general physical laws.For example, third law of thermodynamics. In most of simulation method, few consider the effect of state in each time point. Our research question is whether rule-based simulating process can learn Grn better.

2 Construct Environment

- Run Hopland on single cell data to get energy of each cell.
- Reduce dimension on expression data(PCA,GP-LVM,t-sne)
  - $(g_1, g_2, g_3, ...g_n) \rightarrow$ (component1,component2)
  - combine with energy: (c_1, c_2, E)
- Divide the state space (after reducing dimension) into $m * n$ cells.
- Calculate density and average energy for each cell. Then we will obtain a density matrix and a energy matrix.

$$D_{ij} = |cell - state - set_{ij}|/N$$

$$E_{ij} = E[Energy_{ij}]$$

3 Encoding Rule Based Reward Function

Three Rules

- Cell tends to transform to lower energy state.
- Cell tends to transform to higher density area.

- Distance: Bregman divergence
  Example
  - K-L divergence
  - Euclidean distance
  Gene expression data might be inhomogeneous or non-Euclidean. So, a more general metric, Bregman divergence would fit data better.
  Let ϕ: $C \rightarrow R$ be strictly convex and differentiable on C, then

$$D_\phi(\boldsymbol{x}, \boldsymbol{z}) := \phi(\boldsymbol{x}) - \phi(\boldsymbol{z}) - < \boldsymbol{x} - \boldsymbol{z}, \nabla\phi(\boldsymbol{z}) >$$

4 Reinforcement Learning

- State, s: $(s_1, s_2, ...s_n)$
- Reward: $R_{s,a} = E[R_{t+1}|S_t = s, A_t = a]$
- Return: $G_t = R_{t+1} + \gamma R_{t+2+...}$
- Q: $Q^\pi(s,a) = E[R_{t+1} + \gamma R_{t+2+...}|S_t = s, A_t = a, \pi]$
- Action

5 Encoding Policy

5.1 ODE

Policy: A first order ordinary differential equations and it's parameters.

$$\boldsymbol{h}^{'}(t) = f(\boldsymbol{h}(t)) \text{ with parameters } \boldsymbol{\theta}$$

Our goal is encode policy into network. First, get an ode with empirical formula or auto machine learning. We use forward Euler method to approximate the solution.

$$\frac{\boldsymbol{h}_t - \boldsymbol{h}_{t-1}}{\epsilon} = f(\boldsymbol{h}(t))$$

We have the forward Euler method for a given initial value

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} + \epsilon f(\boldsymbol{h}(t)), \boldsymbol{h}_0 = \boldsymbol{h}(0)$$

5.2 RNN

$$\boldsymbol{y}_t = f(\boldsymbol{y}_{t-1}, \boldsymbol{h}_t, t)$$

where $\boldsymbol{y}_t$ is output of RNN in iteration time t. $\boldsymbol{h}_t$ is input vector.

5.3 Designing RNN by discretizing ODEs

Let $t = n\epsilon$.

$$\boldsymbol{h}(n\epsilon) = \boldsymbol{h}((n-1)\epsilon) + \epsilon f(\boldsymbol{h}(n\epsilon)) \ , \ \boldsymbol{h}_0 = \boldsymbol{h}(0)$$

Let's consider a ODE for GRN inference.

$$\boldsymbol{h}^{'}(t) = tanh(\boldsymbol{A}\boldsymbol{h}_{t-1}) - \lambda \boldsymbol{h}_{t-1}$$

Discretizing it into

$$\boldsymbol{h}(n\epsilon) = \boldsymbol{h}((n-1)\epsilon) + \epsilon tanh(\boldsymbol{A}\boldsymbol{h}((n-1)\epsilon) - \lambda \boldsymbol{h}((n-1)\epsilon)$$

And then encode it as a RNN.