Isolating the perturbation response of gene regulatory networks in the presence of biological variability and technical noise

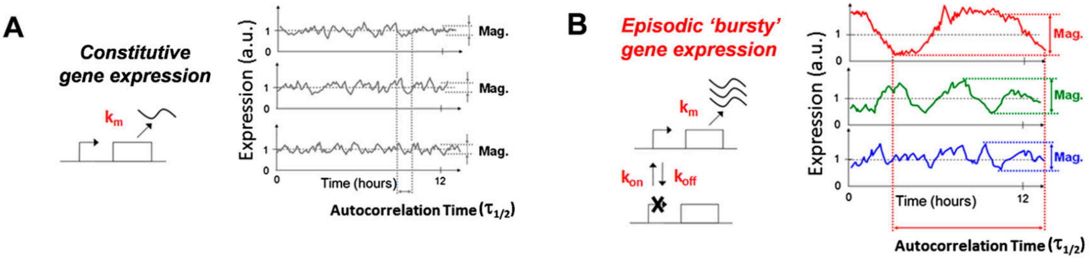
Clayton W. Seitz

May 19, 2022

Outline

Modeling stochastic biochemical reaction networks

Gene expression is stochasic and non-constitutive


A
Constitutive gene expression
k_m
Expression (a.u.)
1
0
Mag.
Time (hours)
0
12
Autocorrelation Time ($\tau_{1/2}$)
B
Episodic 'bursty' gene expression
k_m
k_{on}
k_{off}
Expression (a.u.)
1
0
Mag.
Time (hours)
0
12
Autocorrelation Time ($\tau_{1/2}$)

- If the biochemical network is known a-priori, we can build parametric dynamical models
- Bayesian inference allows us to fit parametric dynamical models without continuous dynamical trajectories

Stochastic biochemical reaction networks: the repressilator

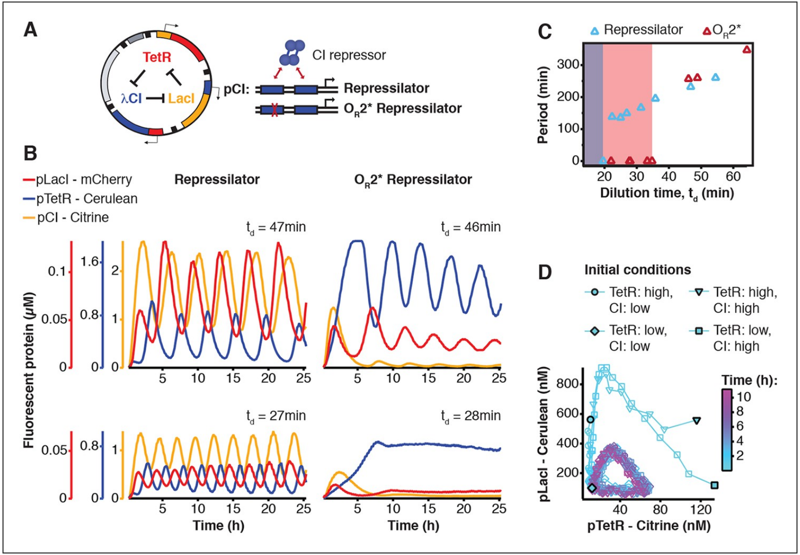


Figure 1: Niederholtmyer et al., eLife 2015

Kolmogorov's forward equation (chemical master equation)

Dynamics on biochemical reaction networks are inherently stochastic and the state space is discrete. We can only write probabilities over the state space

$$
\begin{aligned}
P(\mathrm{x}_i, t) &= \sum_j T_{ji}(\mathrm{x}_i, t|\mathrm{x}_j, t - \Delta t)P(\mathrm{x}_j, t - \Delta t) \\
&= \sum_k T_k(\mathrm{x}_i, t|\mathrm{x}_i - \nu_k, t - \Delta t)P(\mathrm{x}_i - \nu_k, t - \Delta t)
\end{aligned}
$$

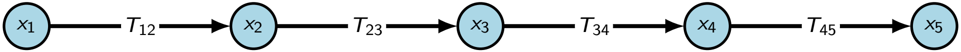
where T_k is the probability of a reaction channel k firing in the interval $(t, t + \Delta t)$.

Taking the limit $\Delta t \to 0$ one can derive the forward Kolmogorov equation or chemical master equation (CME)

$$
\frac{dP(\mathrm{x}, t|\mathrm{x}_0)}{dt} = \sum_k T_k(\mathrm{x} - \nu_k)P(\mathrm{x} - \nu_k, t) - T_k(\mathrm{x})P(\mathrm{x}, t)
$$

What about Markov models?

From another point of view, since the dynamics are Markov the state x follows the DAG



For MMs, the EM algorithm can be used for MAP estimation, but this requires time-series measurements

Time-series measurements in live cells severely limits the number of species considered simultaneously

More often than not the data we have are *ensemble snapshots*

Bayesian parameter inference using ensemble snapshots

Suppose we have a series of ensemble snapshots of an *in-vitro* population:

$$\mathrm{x} = \{\mathrm{x}_0, ..., \mathrm{x}_t\} \quad \mathrm{y} = \{\mathrm{y}_0, ..., \mathrm{y}_t\}$$

with $\mathrm{x}_t = \{x_1, ..., x_n\}$ and similarly for y. Under perfect measurements $\mathrm{x} = \mathrm{y}$

We would like to use x to fit a dynamical model $\mathcal{M}(\theta)$. Bayesian inference lets us infer θ from x while quantifying the uncertainty in our estimate:

$$P(\theta|\mathrm{x}) \propto f(\mathrm{x}|\theta)\pi(\theta) = \pi(\theta)\prod_t f(\mathrm{x}_t|\theta)$$

The likelihood $f(\mathrm{x}_t|\theta)$ is often difficult to define or intractable to compute due to the curse of dimensionality, making even MLE a challenge

Beating the curse of dimensionality for parameter inference

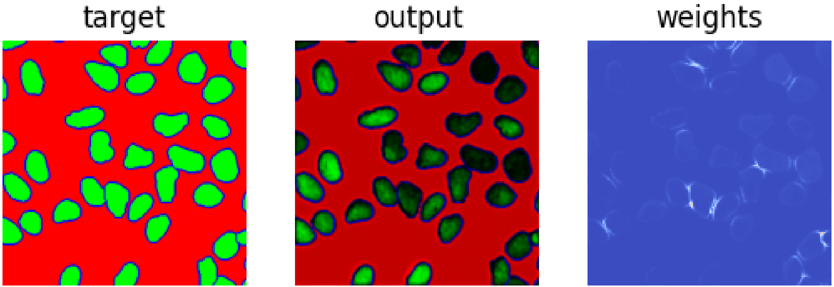
We want to avoid needing to train new networks for every parameter value θ (expensive!) then computing the likelihood of the experimental data. Instead we compute the likelihood of simulated data under set of target variational distributions trained on experimental data. This assumes simulations and experimental data are "exchangeable" when computing the posterior

Variational step: we learn N_t target distributions by training a deep network on the experimental data. In this way we have N_t variational target distributions

ABC step: We sample parameters from our prior $\theta \sim \pi(\theta)$, and produce N Monte Carlo trajectories $\mathrm{x}(t)$. We compute the likelihood of the simulated trajectory with a tolerance ϵ (with a tolerance schedule). This replaces the distance metric in ABC with a variational likelihood

Training on BBBC039 U2OS Cells

BBBC039: 200 images, 160 train + 40 validation, 256 x 256 random crop

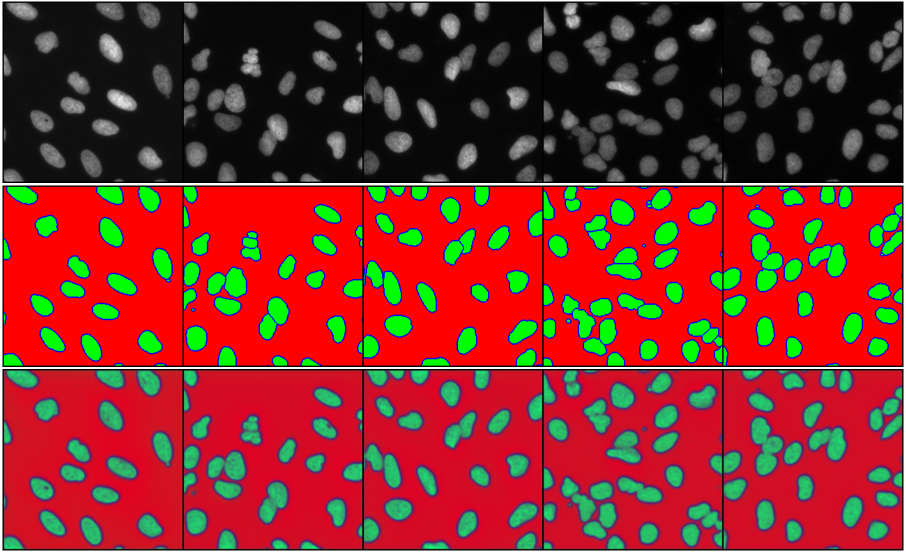


We train a 3-channel semantic segmentation model with **weighted** cross-entropy loss:

$$\mathcal{L} = \sum_{i,j} w_{ij} \log p_{ij}(\tilde{x}) = \sum_{i,j} w_{ij} \log \frac{\exp(-s_{ij}(\tilde{x}))}{\sum_{x \in \chi} \exp(-s_{ij}(\tilde{x}))}$$

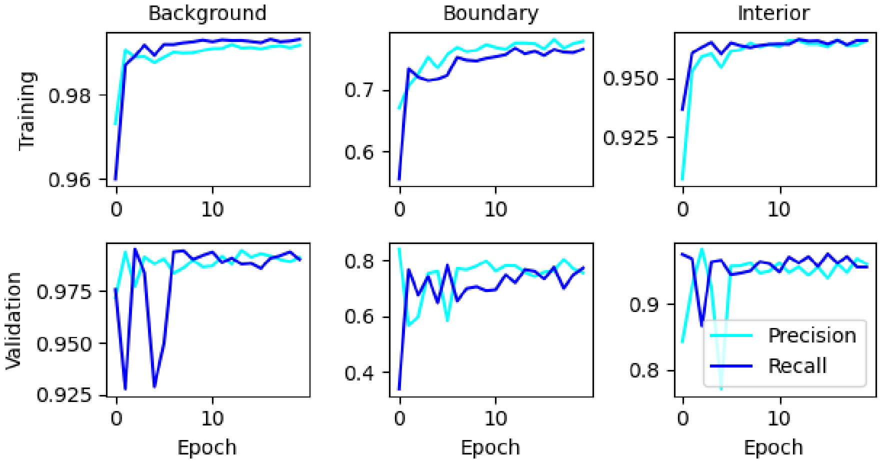
p_{ij} is the probability the model assigns a pixel to the true class $\tilde{x} \in \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}$

Training on BBBC039 U2OS Cells



Training on BBBC039 U2OS Cells

Learning rate $\eta = 0.01$, Batch-size $B = 5$ (32 train iterations, 8 validation)



References I