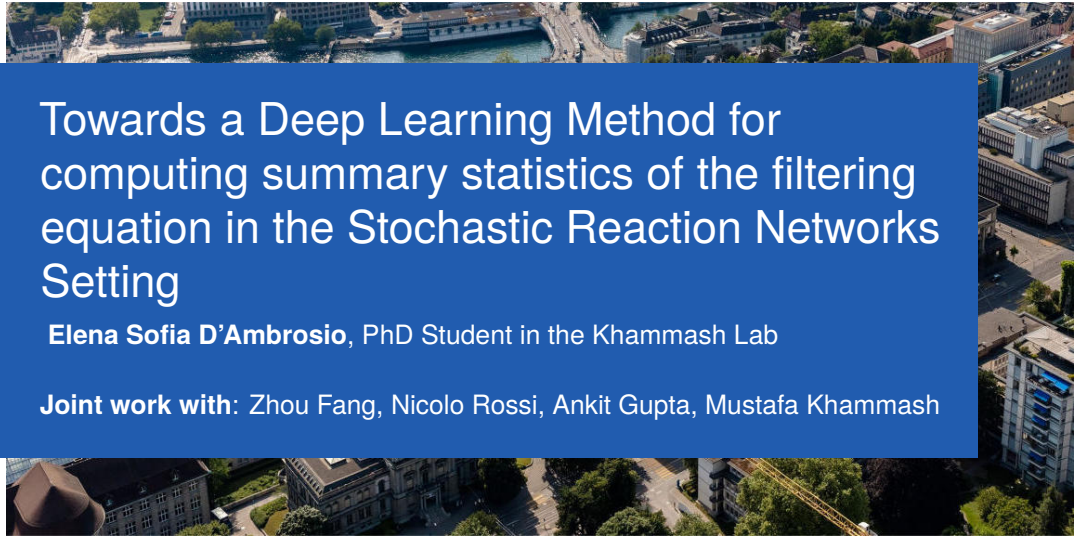
ETH zürich

Department of Biosystems Science and Engineering

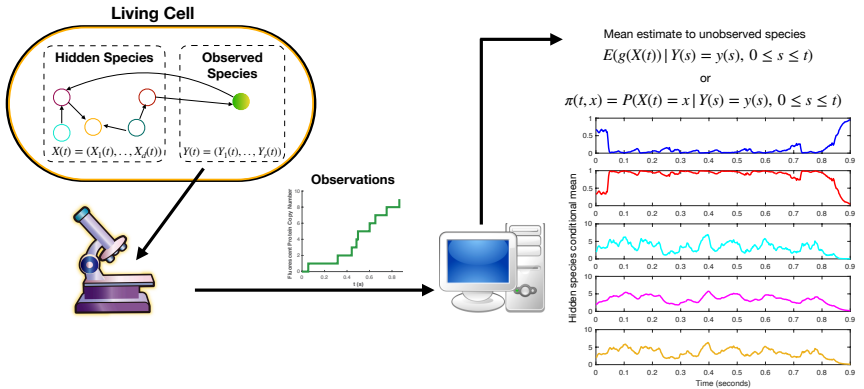


Towards a Deep Learning Method for computing summary statistics of the filtering equation in the Stochastic Reaction Networks Setting

Elena Sofia D'Ambrosio, PhD Student in the Khammash Lab

Joint work with: Zhou Fang, Nicolo Rossi, Ankit Gupta, Mustafa Khammash

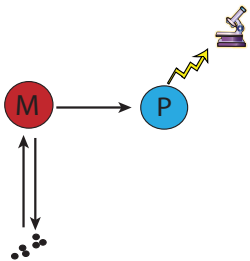
Stochastic Filtering in Biology

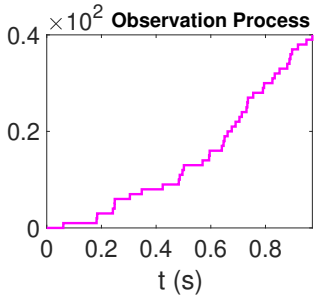


Applicability:

- to unravel cellular functions not fully understood
- inference purposes
- design feedback controllers

Drawbacks of the current methods: Kalman Filter





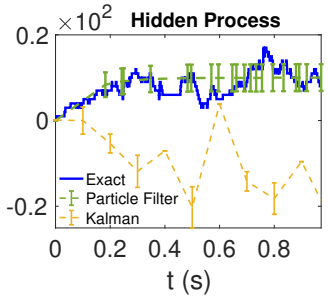
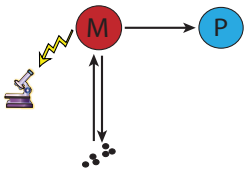


Figure: Observation and Hidden Process Trajectories and Conditional Mean and Standard Deviations Estimations

Kalman Filter (– –):

- It requires linearity of the underlying model
- Not accurate at estimating low copy number regimes: the hidden process trajectory is not even included in the standard deviation bandwidth

Drawbacks of the current methods: Particle Filter (Rathinam, JCP 2021)



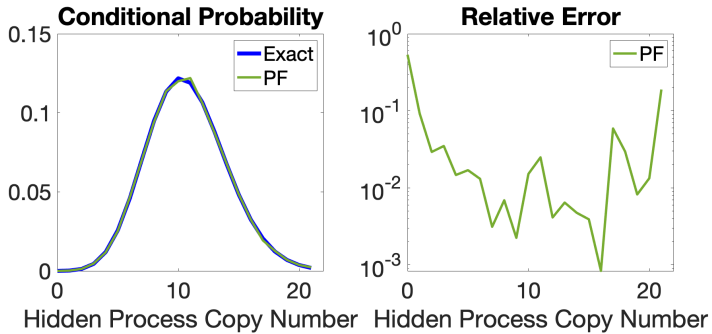


Figure: Observation and Hidden Process Trajectories and Conditional Mean and Standard Deviations Estimations

Particle Filter (—):

- Accurate at estimating summary statistics, but not probability distributions, in particular rare events probabilities

Need for New Filters for the Stochastic Reaction Networks Setting

Goal: Infer Hidden cellular states based on partial observations accurately and efficiently

Current Issues: The existing methods fail due to non-linearity, high-dimensionality, etc.

Our Solution: Provide with computational methods to compute the conditional distribution very accurately and efficiently

Filtering Problem Formulation in the Stochastic Reaction Networks Setting

- Consider an intracellular chemical reacting system that has n species $(S_1, \ldots, S_n)$ and M reactions:

$$\sum_{i=1}^{n} \xi_{ik} S_i \xrightarrow{c_k} \sum_{i=1}^{n} \xi'_{ik} S_i, \quad k = 1, \ldots M,$$

Filtering Problem Formulation in the Stochastic Reaction Networks Setting

- Consider an intracellular chemical reacting system that has n species $(S_1, \ldots, S_n)$ and M reactions:

$$\sum_{i=1}^{n} \xi_{ik} S_i \xrightarrow{c_k} \sum_{i=1}^{n} \xi'_{ik} S_i, \quad k = 1, \ldots M,$$

- $a_1, a_2, \ldots a_M$ the propensity functions and $\nu_k \triangleq \xi'_k - \xi_k$, $k = 1, \ldots, M$, the stoichiometry vectors

Filtering Problem Formulation in the Stochastic Reaction Networks Setting

- Consider an intracellular chemical reacting system that has n species $(S_1, \ldots, S_n)$ and M reactions:

$$\sum_{i=1}^{n} \xi_{ik} S_i \xrightarrow{c_k} \sum_{i=1}^{n} \xi'_{ik} S_i, \quad k = 1, \ldots M,$$

- $a_1, a_2, \ldots a_M$ the propensity functions and $\nu_k \triangleq \xi'_k - \xi_k$, $k = 1, \ldots, M$, the stoichiometry vectors
- Consider a CTMC $\mathbf{Z}(t) \in \mathbb{Z}^n_{\geq 0}$ associated with the reaction network

Filtering Problem Formulation in the Stochastic Reaction Networks Setting

- Consider an intracellular chemical reacting system that has n species $(S_1, \ldots, S_n)$ and M reactions:

$$\sum_{i=1}^{n} \xi_{ik} S_i \xrightarrow{c_k} \sum_{i=1}^{n} \xi'_{ik} S_i, \quad k = 1, \ldots M,$$

- $a_1, a_2, \ldots a_M$ the propensity functions and $\nu_k \triangleq \xi'_k - \xi_k$, $k = 1, \ldots, M$, the stoichiometry vectors
- Consider a CTMC $\mathbf{Z}(t) \in \mathbb{Z}^n_{\geq 0}$ associated with the reaction network
- We decompose the system into two sub-networks $\mathbf{Z}(t) = (\mathbf{X}(t), \mathbf{Y}(t))$:
  - $\mathbf{X}(t) \in \mathcal{X} \subseteq \mathbb{Z}^{n_1}_{\geq 0}$ hidden species copy numbers
  - $\mathbf{Y}(t) \in \mathbb{Z}^{n_2}_{\geq 0}$ (with $n_2 = n - n_1$) observed species

Filtering Problem Formulation in the Stochastic Reaction Networks Setting

- Consider an intracellular chemical reacting system that has n species $(S_1, \ldots, S_n)$ and M reactions:

$$\sum_{i=1}^{n} \xi_{ik} S_i \xrightarrow{c_k} \sum_{i=1}^{n} \xi'_{ik} S_i, \quad k = 1, \ldots M,$$

- $a_1, a_2, \ldots a_M$ the propensity functions and $\nu_k \triangleq \xi'_k - \xi_k$, $k = 1, \ldots, M$, the stoichiometry vectors
- Consider a CTMC $\mathbf{Z}(t) \in \mathbb{Z}^n_{\geq 0}$ associated with the reaction network
- We decompose the system into two sub-networks $\mathbf{Z}(t) = (\mathbf{X}(t), \mathbf{Y}(t))$:
  - $\mathbf{X}(t) \in \mathcal{X} \subseteq \mathbb{Z}^{n_1}_{\geq 0}$ hidden species copy numbers
  - $\mathbf{Y}(t) \in \mathbb{Z}^{n_2}_{\geq 0}$ (with $n_2 = n - n_1$) observed species
- We can write the Chemical Master Equation (CME):

$$\frac{dp(t,z)}{dt} = \sum_{j=1}^{M} a_j(z - \nu_j) p(t, z - \nu_j) - p(t,z) \sum_{j=1}^{M} a_j(z) \quad z \in \mathbb{Z}^n,\ t > 0$$

Filtering Equation for the Noise-Free Case (Kushner-Stratonovich)

- We can define:
  - n_1 components to define ν_k', and ν_k'' with the remaining n_2 components
  - $\mathcal{O} = \{i = 1, \ldots, M | \nu_i'' \neq 0\}$ and $\mathcal{O}^c = \mathcal{U}$
  - $\mathcal{O}_k = \{i \in \mathcal{O} | \nu_i'' = y(t_k) - y(t_k^-)\}$,

Filtering Equation for the Noise-Free Case (Kushner-Stratonovich)

- We can define:
  - n_1 components to define ν_k', and ν_k'' with the remaining n_2 components
  - $\mathcal{O} = \{i = 1, \ldots, M | \nu_i'' \neq 0\}$ and $\mathcal{O}^c = \mathcal{U}$
  - $\mathcal{O}_k = \{i \in \mathcal{O} | \nu_i'' = y(t_k) - y(t_k^-)\}$,
- Our goal is to compute $\pi(t,x) \triangleq P\{\mathbf{X}(t) = x | \mathbf{Y}(s) = y(s), 0 \leq s \leq t\}$ for any $x \in \mathbb{Z}_+^{n_1}$, and for $t_k \leq t \leq t_{k+1}$:

$$\pi'(t,x) = \sum_{j \in \mathcal{U}} \pi(t, x - \nu_j') a_j(x - \nu_j', y(t_k)) - \sum_{j \in \mathcal{U}} \pi(t,x) a_j(x, y(t_k))$$

$$- \pi(t,x) \left(a^{\mathcal{O}}(x, y(t_k)) - \sum_{x'} a^{\mathcal{O}}(x', y(t_k)) \pi(t,x') \right) \quad \forall x \in \mathbb{Z}_+^{n_1},$$

where $a^{\mathcal{O}}(x, y(t_k)) = \sum_{j \in \mathcal{O}} a_j(x, y(t_k))$.

Filtering Equation for the Noise-Free Case (Kushner-Stratonovich)

- We can define:
  - n_1 components to define ν_k', and ν_k'' with the remaining n_2 components
  - $\mathcal{O} = \{i = 1, \ldots, M | \nu_i'' \neq 0\}$ and $\mathcal{O}^c = \mathcal{U}$
  - $\mathcal{O}_k = \{i \in \mathcal{O} | \nu_i'' = y(t_k) - y(t_k^-)\}$,
- Our goal is to compute $\pi(t,x) \triangleq P\{\mathbf{X}(t) = x | \mathbf{Y}(s) = y(s), 0 \leq s \leq t\}$ for any $x \in \mathbb{Z}_+^{n_1}$, and for $t_k \leq t \leq t_{k+1}$:

$$\pi'(t,x) = \sum_{j \in \mathcal{U}} \pi(t, x - \nu_j') a_j(x - \nu_j', y(t_k)) - \sum_{j \in \mathcal{U}} \pi(t,x) a_j(x, y(t_k))$$

$$- \pi(t,x) \left(a^{\mathcal{O}}(x, y(t_k)) - \sum_{x'} a^{\mathcal{O}}(x', y(t_k)) \pi(t, x') \right) \quad \forall x \in \mathbb{Z}_+^{n_1},$$

where $a^{\mathcal{O}}(x, y(t_k)) = \sum_{j \in \mathcal{O}} a_j(x, y(t_k))$.

- At the jump times t_k, $k = 1, \ldots$:

$$\pi(t_k, x) = \frac{\sum_{l \in \mathcal{O}_k} a_l(x - \nu_l', y(t_{k-1}) \pi(t_k^-, x - \nu_l')}{\sum_{x'} \sum_{l \in \mathcal{O}_k} a_l(x', y(t_{k-1})) \pi(t_k^-, x')} \quad \forall x \in \mathbb{Z}_+^{n_1}$$

Similarities with the CME and challenges inherent with the Filtering Equation

Methods for solving the CME :

- Finite State Projection (Munsky et al., 2006, J. Chem. Phys.)
- MonteCarlo Methods (SSA, Next Reaction Method) (Gillespie and Petzold J. Chem. Phys. 2009, Anderson et al. J. Chem. Phys. 2006)
- Moment Closure Methods
- Machine Learning Algorithms (DeepCME) (Gupta et al., PLOS CB, 2022)
- Hybrid Methods (Fang et al. 2022 bioRxiv, J Hasenauer et al. J Math. Biol. 2014, Duso et al. J. Chem. Phys. 2018)

[1] Elena Sofia D'Ambrosio et al. "Filtered finite state projection method for the analysis and estimation of stochastic biochemical reaction networks". In: *bioRxiv* (2022).

Similarities with the CME and challenges inherent with the Filtering Equation

Methods for solving the CME :

- Finite State Projection (Munsky et al., 2006, J. Chem. Phys.)
- MonteCarlo Methods (SSA, Next Reaction Method) (Gillespie and Petzold J. Chem. Phys. 2009, Anderson et al. J. Chem. Phys. 2006)
- Moment Closure Methods
- Machine Learning Algorithms (DeepCME) (Gupta et al., PLOS CB, 2022)
- Hybrid Methods (Fang et al. 2022 bioRxiv, J Hasenauer et al. J Math. Biol. 2014, Duso et al. J. Chem. Phys. 2018)

Methods for solving the filtering equation:

- Filtered Finite State Projection (FFSP) (Completed work, D'Ambrosio et al. 2022, bioRxiv)[1]
- Particle Filtering Algorithms (Fang et al. SIAM 2023, Rathinam et al., J. Chem. Phys. 2021)
- Moment Closure Methods (Zechner et al., CMSB 2022)
- Machine Learning (In Progress)

[1] D'Ambrosio et al., "Filtered finite state projection method for the analysis and estimation of stochastic biochemical reaction networks".

How to address the challenges inherent with the filtering equation

Nonlinearity: $\rho(t,x) \triangleq \pi(t,x)\exp\left(-\int_{\max\{t_k|t_k\le t\}}^{t}\left[\sum_{\tilde{x}\in\mathbb{Z}_{\ge 0}^{n_1}} a^{\mathcal{O}}(\tilde{x},\mathbf{Y}(s))\pi(s,\tilde{x})\right]ds\right)$

Un-normalised Distribution satisfies the Zakai Equation

For $t_k \le t < t_{k+1}$ and $x \in \mathbb{Z}^{n_1}$:

$$\rho'(t,x) = \sum_{j\in\mathcal{U}} \rho(t,x-\nu_j')a_j(x-\nu_j',y(t_k)) - \sum_{j\in\mathcal{U}} \rho(t,x)a_j(x,y(t_k)) - \rho(t,x)a^{\mathcal{O}}(x,y(t_k)) \quad \forall x\in\mathbb{Z}_+^{n_1}$$

For $t = t_{k+1}$ and $x \in \mathbb{Z}^{n_1}$: $\rho(t_{k+1},x) = \dfrac{\sum_{j\in\mathcal{O}_{k+1}} a_j(x-\nu_j',\mathbf{Y}(t_k))\rho(t_{k+1}^-,x-\nu_j')}{\sum_{\tilde{x}}\sum_{j\in\mathcal{O}_{k+1}} a_j(\tilde{x},\mathbf{Y}(t_k))\rho(t_{k+1}^-,\tilde{x})}$

How to address the challenges inherent with the filtering equation

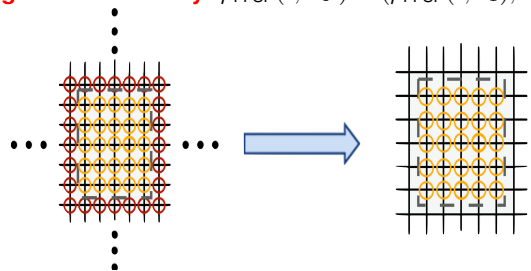
Nonlinearity: $\rho(t,x) \triangleq \pi(t,x)\exp\left(-\int_{\max\{t_k|t_k\le t\}}^{t}\left[\sum_{\tilde{x}\in\mathbb{Z}^{n_1}_{\ge 0}} a^{\mathcal{O}}(\tilde{x},\mathbf{Y}(s))\pi(s,\tilde{x})\right]ds\right)$ Un-normalised Distribution satisfies the Zakai Equation

For $t_k \le t < t_{k+1}$ and $x \in \mathbb{Z}^{n_1}$:

$$\rho'(t,x) = \sum_{j\in\mathcal{U}} \rho(t,x-\nu'_j)a_j(x-\nu'_j,y(t_k)) - \sum_{j\in\mathcal{U}} \rho(t,x)a_j(x,y(t_k)) - \rho(t,x)a^{\mathcal{O}}(x,y(t_k)) \quad \forall x\in\mathbb{Z}^{n_1}_{+}$$

For $t = t_{k+1}$ and $x \in \mathbb{Z}^{n_1}$: $\rho(t_{k+1},x) = \dfrac{\sum_{j\in\mathcal{O}_{k+1}} a_j(x-\nu'_j,\mathbf{Y}(t_k))\rho(t^-_{k+1},x-\nu'_j)}{\sum_{\tilde{x}}\sum_{j\in\mathcal{O}_{k+1}} a_j(\tilde{x},\mathbf{Y}(t_k))\rho(t^-_{k+1},\tilde{x})}$

High dimensionality: $\rho_{\text{FFSP}}(t,\mathcal{X}_P) = (\rho_{\text{FFSP}}(t,x_1),\ldots,\rho_{\text{FFSP}}(t,x_P))^\top$ finite-dimensional vector



Filtered Finite State Projection

$\dot{\rho}_{\text{FFSP}}(t,\mathcal{X}_P) = \mathbb{A}_P(\mathbf{Y}(t_k))\rho_{\text{FFSP}}(t,\mathcal{X}_P)$

$\forall k \in \mathbb{Z}_{\ge 0}$ and $\forall t \in [t_k, t_{k+1})$

$\rho_{\text{FFSP}}(t_{k+1},\mathcal{X}_P) = \mathbb{A}_{P_{\text{jump}}}(\mathbf{Y}(t_k))\rho_{\text{FFSP}}(t^-_{k+1},\mathcal{X}_P)$

Motivation: Real-Time Estimations of Cellular States

- FFSP provides accurate estimations (benchmark for validation) but suffers from slow computational speed

Motivation: Real-Time Estimations of Cellular States

- FFSP provides accurate estimations (benchmark for validation) but suffers from slow computational speed
- **Objective**: Harness the computational power of ML for real-time estimations

Motivation: Real-Time Estimations of Cellular States

- FFSP provides accurate estimations (benchmark for validation) but suffers from slow computational speed
- **Objective**: Harness the computational power of ML for real-time estimations
- Deep Learning Main Approaches for filtering and high dimensional ODEs, PDEs (including CME):

Motivation: Real-Time Estimations of Cellular States

- FFSP provides accurate estimations (benchmark for validation) but suffers from slow computational speed
- **Objective**: Harness the computational power of ML for real-time estimations
- Deep Learning Main Approaches for filtering and high dimensional ODEs, PDEs (including CME):
  - Approximating the solution of the equation with the NN during the training by comparing l.h.s. and r.h.s-Lu et al. 2021, Tang et al. 2023

Motivation: Real-Time Estimations of Cellular States

- FFSP provides accurate estimations (benchmark for validation) but suffers from slow computational speed
- **Objective**: Harness the computational power of ML for real-time estimations
- Deep Learning Main Approaches for filtering and high dimensional ODEs, PDEs (including CME):
  - Approximating the solution of the equation with the NN during the training by comparing l.h.s. and r.h.s-Lu et al. 2021, Tang et al. 2023
  - Reinforcement Learning to solve a Kolmogorov Backward Equation based on simulation data-Han et al. 2018, Gupta et al. 2021, Crisan et al. 2022, **Our Approach**

Motivation: Real-Time Estimations of Cellular States

- FFSP provides accurate estimations (benchmark for validation) but suffers from slow computational speed
- **Objective**: Harness the computational power of ML for real-time estimations
- Deep Learning Main Approaches for filtering and high dimensional ODEs, PDEs (including CME):
  - Approximating the solution of the equation with the NN during the training by comparing l.h.s. and r.h.s-Lu et al. 2021, Tang et al. 2023
  - Reinforcement Learning to solve a Kolmogorov Backward Equation based on simulation data-Han et al. 2018, Gupta et al. 2021, Crisan et al. 2022, **Our Approach**
  - Reconstruction of the dynamics such that the hidden process becomes a controlled process (by the observation) whose joint distribution coincides with the conditional distribution (**Feedback Dynamics**)- Koeppl et al. 2021

Motivation: Real-Time Estimations of Cellular States

- FFSP provides accurate estimations (benchmark for validation) but suffers from slow computational speed
- **Objective**: Harness the computational power of ML for real-time estimations
- Deep Learning Main Approaches for filtering and high dimensional ODEs, PDEs (including CME):
  - Approximating the solution of the equation with the NN during the training by comparing l.h.s. and r.h.s-Lu et al. 2021, Tang et al. 2023
  - Reinforcement Learning to solve a Kolmogorov Backward Equation based on simulation data-Han et al. 2018, Gupta et al. 2021, Crisan et al. 2022, **Our Approach**
  - Reconstruction of the dynamics such that the hidden process becomes a controlled process (by the observation) whose joint distribution coincides with the conditional distribution (**Feedback Dynamics**)- Koeppl et al. 2021
  - Learning the Mapping between the Observation Process and the Conditional Estimator through a DNN, (**Deep Filtering**)-Ghosh et al. 2022, Zhang et al. 2020, Bishop et al. 2023

Extending the DeepCME approach to solve the noise-free filtering equation

- Let $\mathbb{Z}(t) \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ be the state of a CTMC and $\mathbf{Z}(0) = z_0$

Extending the DeepCME approach to solve the noise-free filtering equation

- Let $\mathbb{Z}(t) \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ be the state of a CTMC and $\mathbf{Z}(0) = z_0$
- In the DeepCME , the goal is to compute

$$\mathbb{E}(g(\mathbf{Z}(t))) = \sum_{x \in \mathcal{Z}} g(z) p(t, z)$$

where g is a suitable real-valued function, typically called **output** function.

Extending the DeepCME approach to solve the noise-free filtering equation

- Let $\mathbb{Z}(t) \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ be the state of a CTMC and $\mathbf{Z}(0) = z_0$
- In the DeepCME , the goal is to compute

$$\mathbb{E}(g(\mathbf{Z}(t))) = \sum_{x \in \mathcal{Z}} g(z) p(t, z)$$

  where g is a suitable real-valued function, typically called **output** function.
- Given the filtration generated by $(\mathbf{Z}(t))_{t \geq 0}$, we can define a martingale in the interval $[0, T]$:

$$V_g(t, \mathbf{Z}(t)) = \mathbb{E}(g(\mathbf{Z}(T)) | \mathbf{Z}(t)), \qquad \text{such that} \qquad V_g(0, z_0) = \mathbb{E}(g(\mathbf{Z}(T)))$$

Extending the DeepCME approach to solve the noise-free filtering equation

- Let $\mathbb{Z}(t) \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ be the state of a CTMC and $\mathbf{Z}(0) = z_0$
- In the DeepCME , the goal is to compute

$$\mathbb{E}(g(\mathbf{Z}(t))) = \sum_{x \in \mathcal{Z}} g(z)p(t, z)$$

  where g is a suitable real-valued function, typically called **output** function.
- Given the filtration generated by $(\mathbf{Z}(t))_{t \geq 0}$, we can define a martingale in the interval $[0, T]$:

$$V_g(t, \mathbf{Z}(t)) = \mathbb{E}(g(\mathbf{Z}(T))|\mathbf{Z}(t)), \qquad \text{such that} \qquad V_g(0, z_0) = \mathbb{E}(g(\mathbf{Z}(T)))$$

- V_g satisfies this '**almost sure**' relationship:

$$V_g(T, \mathbf{Z}(T)) = V_g(0, Z(0)) + \sum_{k=1}^{M} \int_0^T \Delta_k V_g(t, \mathbf{Z}(t)) d\tilde{R}_k(t).$$

  where for each reaction channel $k = 1, \ldots, M$, $\Delta_k V_g(t, x) := V_g(t, x + \nu_k) - V_g(t, x)$ and $\tilde{R}_k(t) := Y_k \left(\int_0^t a_k(\mathbf{X}(s)) ds \right) - \int_0^t a_k(\mathbf{X}(s)) ds$

Extending the DeepCME approach to solve the noise-free filtering equation

- A deep neural network (DNN) approximates the following map for $z \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ and $t \geq 0$:

$$(t, z) \rightarrow V_g(t, z) = \mathbb{E}(g(\mathbf{Z}(T))|\mathbf{Z}(t) = z)$$

Extending the DeepCME approach to solve the noise-free filtering equation

- A deep neural network (DNN) approximates the following map for $z \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ and $t \geq 0$:

$$(t, z) \rightarrow V_g(t, z) = \mathbb{E}(g(\mathbf{Z}(T))|\mathbf{Z}(t) = z)$$

- The DNN is trained according to the '**almost sure**' relationship

Extending the DeepCME approach to solve the noise-free filtering equation

- A deep neural network (DNN) approximates the following map for $z \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ and $t \geq 0$:

$$(t, z) \to V_g(t, z) = \mathbb{E}(g(\mathbf{Z}(T))|\mathbf{Z}(t) = z)$$

- The DNN is trained according to the '**almost sure**' relationship

- **Our setting**: $\mathbf{Z}(t) = (\mathbf{X}(t), \mathbf{Y}(t)) \in \mathbb{Z}^n_{\geq 0}$, where $\mathbf{X}(t) \in \mathbb{Z}^{n_1}_{\geq 0}$ hidden process and $\mathbf{Y}(t) \in \mathbb{Z}^{n_2}_{\geq 0}$ observed process

Extending the DeepCME approach to solve the noise-free filtering equation

- A deep neural network (DNN) approximates the following map for $z \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ and $t \geq 0$:

$$(t, z) \to V_g(t, z) = \mathbb{E}(g(\mathbf{Z}(T))|\mathbf{Z}(t) = z)$$

- The DNN is trained according to the '**almost sure**' relationship

- **Our setting**: $\mathbf{Z}(t) = (\mathbf{X}(t), \mathbf{Y}(t)) \in \mathbb{Z}^n_{\geq 0}$, where $\mathbf{X}(t) \in \mathbb{Z}^{n_1}_{\geq 0}$ hidden process and $\mathbf{Y}(t) \in \mathbb{Z}^{n_2}_{\geq 0}$ observed process
- **Our Goal**: Compute $\mathbb{E}\left[g(\mathbf{X}(T))\middle|\mathcal{Y}_T\right]$ by finding a martingale representation

Extending the DeepCME approach to solve the noise-free filtering equation

- A deep neural network (DNN) approximates the following map for $z \in \mathcal{Z} \subseteq \mathbb{Z}^n_{\geq 0}$ and $t \geq 0$:

$$(t, z) \to V_g(t, z) = \mathbb{E}(g(\mathbf{Z}(T))|\mathbf{Z}(t) = z)$$

- The DNN is trained according to the '**almost sure**' relationship

- **Our setting**: $\mathbf{Z}(t) = (\mathbf{X}(t), \mathbf{Y}(t)) \in \mathbb{Z}^n_{\geq 0}$, where $\mathbf{X}(t) \in \mathbb{Z}^{n_1}_{\geq 0}$ hidden process and $\mathbf{Y}(t) \in \mathbb{Z}^{n_2}_{\geq 0}$ observed process
- **Our Goal**: Compute $\mathbb{E}\left[g(\mathbf{X}(T))\middle|\mathcal{Y}_T\right]$ by finding a martingale representation
- **How to achieve it**:
  - Define $(\mathbf{V}, w)$ and $M_{g,y_{[0,T]}}(t, \mathbf{V}(t), w(t)) = \mathbb{E}\left[g(\mathbf{V}(T))w(T)|(\mathbf{V}(t), w(t), y_{[0,T]})\right]$ s.t.:

$$\mathbb{E}\left[g(\mathbf{X}(T))|y_{[0,T]}\right] = \frac{\mathbb{E}\left[M_{g,y_{[0,T]}}(0, \mathbf{V}(0), w(0))|y_{[0,T]}\right]}{\mathbb{E}\left[M_{1,y_{[0,T]}}(0, \mathbf{V}(0), w(0))|y_{[0,T]}\right]}$$

  - Approximate $M_{g,y_{[0,T]}}(\cdot)$ with a DNN and train it with an '**almost sure**' relationship

Definition of $\mathbf{V}$ and w over the whole time horizon (Rathinam et al. 2021)

- Under the non-explosivity condition, we can write $\mathbf{V}(t)$ and $w(t)$ over the whole time horizon as the following:

$$\mathbf{V}(t) = \mathbf{V}(0) + \sum_{j \in \mathcal{U}} R_j \left(\int_0^t a_j \left(\mathbf{V}(s), y(s) \right) ds \right) \nu_j' + \sum_{k=1}^{m_1} \int_0^t s_{\mu_k}(s) d\tilde{R}_{\mu_k}(s)$$

$$w(t) = w(0) \exp \left\{ - \int_0^t a^{\mathcal{O}}(\mathbf{V}(s), y(s)) ds + \sum_{k=1}^{m_1} \int_0^t \ln A_{\mu_k}(s) d\tilde{R}_{\mu_k}(s) \right\},$$

Definition of $\mathbf{V}$ and w over the whole time horizon (Rathinam et al. 2021)

- Under the non-explosivity condition, we can write $\mathbf{V}(t)$ and $w(t)$ over the whole time horizon as the following:

$$\mathbf{V}(t) = \mathbf{V}(0) + \sum_{j \in \mathcal{U}} R_j \left(\int_0^t a_j \left(\mathbf{V}(s), y(s) \right) ds \right) \nu_j' + \sum_{k=1}^{m_1} \int_0^t s_{\mu_k}(s) d\tilde{R}_{\mu_k}(s)$$

$$w(t) = w(0) \exp \left\{ - \int_0^t a^{\mathcal{O}}(\mathbf{V}(s), y(s)) ds + \sum_{k=1}^{m_1} \int_0^t \ln A_{\mu_k}(s) d\tilde{R}_{\mu_k}(s) \right\},$$

- where $\{s_{\mu_k}(t), A_{\mu_k}(t)\}_{t \geq 0}$ are two stochastic processes described for each $t \geq 0$ and $k = 1, \ldots, m_1$ by the following joint probability distribution:

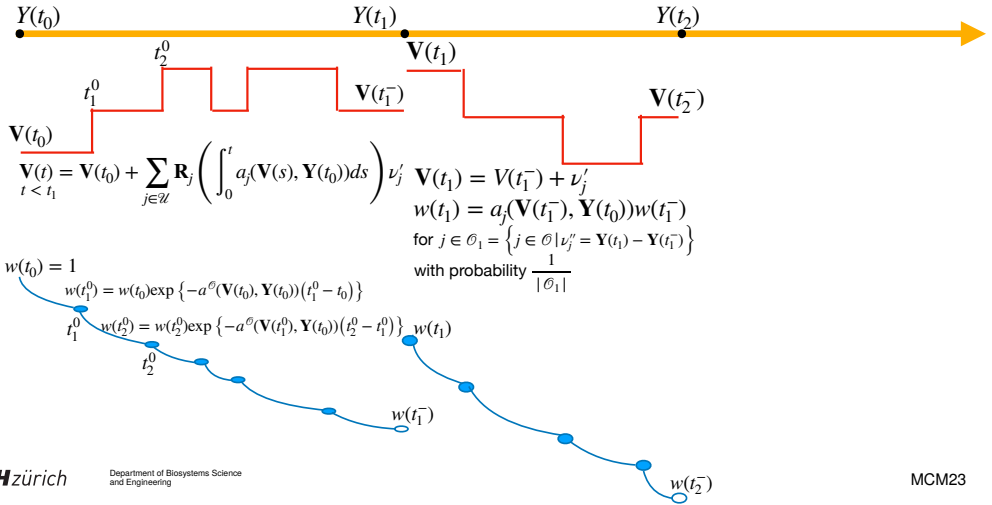
$$\mathbb{P}\left((s_{\mu_k}(t), A_{\mu_k}(t)) = (\nu_j', a_j(\mathbf{V}(t^-), y(t^-))) | \, (\mathbf{V}(t^-), y(t^-)) \right)$$

$$= \begin{cases} \frac{1}{|\tilde{\mathcal{O}}_{\mu_k}(t^-)|} & \text{if } j \in \tilde{\mathcal{O}}_{\mu_k}(t^-) \\ 0 & \text{otherwise} \end{cases}$$

being $\tilde{\mathcal{O}}_{\mu_k}(t) = \{ j \in \mathcal{O}_{\mu_k} | a_j(\mathbf{V}(t), y(t)) \neq 0 \}$.

Schematic Representation of V and w

$\mathbf{Z}(t) = (\mathbf{X}(t), \mathbf{Y}(t)) \in \mathbb{Z}^n_{\geq 0}$ $\mathbf{X}(t) \in \mathcal{X} \subseteq \mathbb{Z}^{n_1}_{\geq 0}$ hidden species $\mathbf{Y}(t) \in \mathbb{Z}^{n_2}_{\geq 0}$ observed species
$(\mathbf{V}(t), w(t))$ auxiliary processes $\mathbf{V}(t) \in \mathbb{Z}^{n_1}_{\geq 0}$ mimicking the hidden dynamics $w(t) \in \mathbb{R}_{\geq 0}$



Martingale Dynamics

Between the observation process jump times, $t_k \leq t < t_{k+1}$, $M_{g,y}$ satisfies:

$$dM_{g,y_{[0,T]}}(t, \mathbf{V}(t), w(t)) = M_{g,y_{[0,T]}}(t, \mathbf{V}(t), w(t)) \sum_{j \in \mathcal{U}} a_j(\mathbf{V}(t), y(t_k))dt$$

$$- \sum_{j \in \mathcal{U}} M_{g,y_{[0,T]}}(t, \mathbf{V}(t) + \nu'_j, w) a_j(\mathbf{V}(t), y(t_k))dt$$

$$+ \sum_{j \in \mathcal{U}} \left[M_{g,y_{[0,T]}}(t^-, \mathbf{V}(t^-) + \nu'_j, w(t^-)) - M_{g,y_{[0,T]}}(t^-, \mathbf{V}(t^-), w(t^-)) \right] dR_j(t)$$

Martingale Dynamics

Between the observation process jump times, $t_k \leq t < t_{k+1}$, $M_{g,y}$ satisfies:

$$dM_{g,y_{[0,T]}}(t, \mathbf{V}(t), w(t)) = M_{g,y_{[0,T]}}(t, \mathbf{V}(t), w(t)) \sum_{j \in \mathcal{U}} a_j(\mathbf{V}(t), y(t_k))dt$$

$$- \sum_{j \in \mathcal{U}} M_{g,y_{[0,T]}}(t, \mathbf{V}(t) + \nu_j', w) a_j(\mathbf{V}(t), y(t_k))dt$$

$$+ \sum_{j \in \mathcal{U}} \left[M_{g,y_{[0,T]}}(t^-, \mathbf{V}(t^-) + \nu_j', w(t^-)) - M_{g,y_{[0,T]}}(t^-, \mathbf{V}(t^-), w(t^-)) \right] dR_j(t)$$

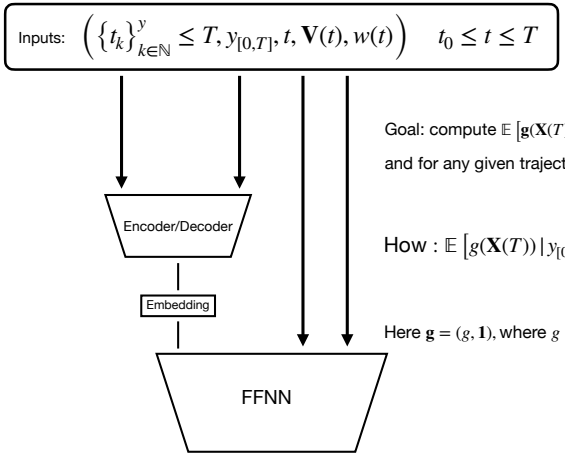
Then $M_{g,y_{[0,T]}}$ satisfies the following dynamic backward equation for $t = t_k$:

$$M_{g,y_{[0,T]}}(t_k^-, \mathbf{V}(t_k^-), w(t_k^-))$$

$$= \frac{1}{\tilde{\mathcal{O}}_{\mu_l}(t_k^-)} \sum_{j \in \tilde{\mathcal{O}}_{\mu_l}(t_k^-)} M_{g,y_{[0,T]}}(t_k, \mathbf{V}(t_k^-) + \nu_j', a_j(\mathbf{V}(t_k^-), y(t_{k-1}))w(t_k^-))$$

where $\mu_l = y(t_k) - y(t_k^-)$.

Neural Network Structure



Goal: compute $\mathbb{E}\left[\mathbf{g}(\mathbf{X}(T)) \,|\, y_{[0,T]}\right]$ for some bounded function $\mathbf{g}$ and $T > 0$

and for any given trajectory of $\mathbf{Y}(s) = y(s) \quad 0 \le s \le T,$ namely $y_{[0,T]}$

How : $\mathbb{E}\left[g(\mathbf{X}(T)) \,|\, y_{[0,T]}\right] = \dfrac{\mathbb{E}\left[M_{g,y_{[0,T]}}(0,\mathbf{V}(0), w(0) \,|\, y_{[0,T]}\right]}{\mathbb{E}\left[M_{1,y_{[0,T]}}(0,\mathbf{V}(0), w(0) \,|\, y_{[0,T]}\right]}$

Here $\mathbf{g} = (g, \mathbf{1})$, where $g : \mathbb{Z}_{\geq 0}^{n_1} \to \mathbb{R}^M$ and $\mathbf{1} : \mathbb{Z}_{\geq 0}^{n_1} \to \mathbb{R}$ constant one function

Loss Function Definition

- $M_{g,y_{[0,T]}}$ satisfies between the observation process jump times $t_k \leq t < t_{k+1}$:

$$M_{g,y_{[0,T]}}(t_{k+1}^-, \mathbf{V}(t_{k+1}^-), w(t_{k+1}^-)) = M_{g,y_{[0,T]}}(t_k, \mathbf{V}(t_k), w(t_k))$$
$$+ \sum_{l \in \mathcal{U}} \int_{t_k}^{t_{k+1}^-} A_l M_{g,y_{[0,T]}}(s) d\tilde{R}_l(s)$$

where $A_j f(t,v,w) = f(t, v + \nu_j', w) - f(t,v,w)$ with f a bounded function in the domain of A_j for $j \in \mathcal{U}$ and $\tilde{R}_j$, for $j \in \mathcal{U}$, are the centered Poisson processes.

Loss Function Definition

- $M_{g,y_{[0,T]}}$ satisfies between the observation process jump times $t_k \leq t < t_{k+1}$:

$$M_{g,y_{[0,T]}}(t_{k+1}^-, \mathbf{V}(t_{k+1}^-), w(t_{k+1}^-)) = M_{g,y_{[0,T]}}(t_k, \mathbf{V}(t_k), w(t_k))$$
$$+ \sum_{l \in \mathcal{U}} \int_{t_k}^{t_{k+1}^-} A_l M_{g,y_{[0,T]}}(s) d\tilde{R}_l(s)$$

  where $A_j f(t,v,w) = f(t, v + \nu_j', w) - f(t,v,w)$ with f a bounded function in the domain of A_j for $j \in \mathcal{U}$ and $\tilde{R}_j$, for $j \in \mathcal{U}$, are the centered Poisson processes.

- **Main Idea**: Creating the loss function according to the martingale dynamics between the jumps, and adding the costraints at the jumps

Loss Function Definition

- $M_{g,y_{[0,T]}}$ satisfies between the observation process jump times $t_k \leq t < t_{k+1}$:

$$M_{g,y_{[0,T]}}(t_{k+1}^-, \mathbf{V}(t_{k+1}^-), w(t_{k+1}^-)) = M_{g,y_{[0,T]}}(t_k, \mathbf{V}(t_k), w(t_k)) + \sum_{l \in \mathcal{U}} \int_{t_k}^{t_{k+1}^-} A_l M_{g,y_{[0,T]}}(s) d\tilde{R}_l(s)$$

  where $A_j f(t,v,w) = f(t, v + \nu'_j, w) - f(t,v,w)$ with f a bounded function in the domain of A_j for $j \in \mathcal{U}$ and $\tilde{R}_j$, for $j \in \mathcal{U}$, are the centered Poisson processes.

- **Main Idea**: Creating the loss function according to the martingale dynamics between the jumps, and adding the costraints at the jumps
- It is reasonable to think that the $NN^{\theta}_{g,y_{[0,T]}}$ minimising the loss function should be a good approximation of $M_{g,y_{[0,T]}}$

$$NN^{\theta}_{g,y_{[0,T]}} \approx M_{g,y_{[0,T]}}$$

Loss Function Definition

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathscr{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})\right]$$

$$\mathscr{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]}) := \sum_{t_{k+1}<T} \left| NN^{\theta}_{g,y_{[0,T]}}(\{t_k\}^{y}_{k\in\mathbb{N}}, y_{[0,T]}, t^{-}_{k+1}, \mathbf{V}(t^{-}_{k+1}), w(t^{-}_{k+1})) - NN^{\theta}_{g,y_{[0,T]}}(\{t_k\}^{y}_{k\in\mathbb{N}}, y_{[0,T]}, t_k, \mathbf{V}(t_k), w(t_k)) - \sum_{l\in\mathscr{U}} \int_{t_k}^{t^{-}_{k+1}} A_l NN^{\theta}_{g,y_{[0,T]}}(s) d\tilde{R}_l(s) \right|$$

Evolution of the martingale until the last jump before T

$$+ \left| g(\mathbf{V}(T))w(T)) - NN^{\theta}_{g,y_{[0,T]}}(\{t_k\}^{y}_{k\in\mathbb{N}}, y_{[0,T]}, t_{\text{last}}, \mathbf{V}(t_{\text{last}}), w(t_{\text{last}})) - \sum_{l\in\mathscr{U}} \int_{t_{\text{last}}}^{T} A_l NN^{\theta}_{g,y_{[0,T]}}(s) d\tilde{R}_l(s) \right|$$

Evolution of the martingale between the last jump and T

$$+ \sum_{t_k<T} \left| NN^{\theta}_{g,y_{[0,T]}}(\{t_k\}^{y}_{k\in\mathbb{N}}, y_{[0,T]}, t^{-}_{k}, \mathbf{V}(t^{-}_{k}), w(t^{-}_{k})) - \frac{1}{\tilde{\mathcal{O}}_{\mu_l}(t^{-}_{k})} \sum_{j\in\tilde{\mathcal{O}}_{\mu_l}(t^{-}_{k})} NN^{\theta}_{g,y_{[0,T]}}(\{t_k\}^{y}_{k\in\mathbb{N}}, y_{[0,T]}, t_k, \mathbf{V}(t^{-}_{k}) + \nu'_j, a_j(\mathbf{V}(t^{-}_{k}), y(t_{k-1}))w(t^{-}_{k})) \right|$$

Evolution of the martingale at the jump times before T

Loss Function Computation

- **Training**: Compute the loss function by sampling

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]}) \middle| \mathcal{Y}_{0:T}\right]\right]$$

where $\mathcal{Y}_{0:T}$ is the filtration generated by the observation process from time $t = 0$ to $t = T$.

Loss Function Computation

- **Training**: Compute the loss function by sampling

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]}) \middle| \mathcal{Y}_{0:T}\right]\right]$$

  where $\mathcal{Y}_{0:T}$ is the filtration generated by the observation process from time $t = 0$ to $t = T$.

- How to practically compute the expectations in the loss function given the inaccessibility to all the trajectories?

Loss Function Computation

- **Training**: Compute the loss function by sampling

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]}) \middle| \mathcal{Y}_{0:T}\right]\right]$$

  where $\mathcal{Y}_{0:T}$ is the filtration generated by the observation process from time $t = 0$ to $t = T$.
- How to practically compute the expectations in the loss function given the inaccessibility to all the trajectories?
- Strategy: Monte-Carlo methods

Loss Function Computation

- **Training**: Compute the loss function by sampling

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})\middle|\mathcal{Y}_{0:T}\right]\right]$$

  where $\mathcal{Y}_{0:T}$ is the filtration generated by the observation process from time $t = 0$ to $t = T$.
- How to practically compute the expectations in the loss function given the inaccessibility to all the trajectories?
- Strategy: Monte-Carlo methods
  - Fix a simulation of $\mathbf{Y}(t)$ for $0 \leq t \leq T$, namely $y_{[0,T]}$

Loss Function Computation

- **Training**: Compute the loss function by sampling

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})\middle|\mathcal{Y}_{0:T}\right]\right]$$

  where $\mathcal{Y}_{0:T}$ is the filtration generated by the observation process from time $t = 0$ to $t = T$.
- How to practically compute the expectations in the loss function given the inaccessibility to all the trajectories?
- Strategy: Monte-Carlo methods
  - Fix a simulation of $\mathbf{Y}(t)$ for $0 \leq t \leq T$, namely $y_{[0,T]}$
  - Sample q different trajectories of the processes $(\mathbf{V}, w)$: $(\mathbf{V}^1, w^1), (\mathbf{V}^2, w^2) \ldots, (\mathbf{V}^q, w^q)$

Loss Function Computation

- **Training**: Compute the loss function by sampling

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})\Big|\mathcal{Y}_{0:T}\right]\right]$$

  where $\mathcal{Y}_{0:T}$ is the filtration generated by the observation process from time $t = 0$ to $t = T$.
- How to practically compute the expectations in the loss function given the inaccessibility to all the trajectories?
- Strategy: Monte-Carlo methods
  - Fix a simulation of $\mathbf{Y}(t)$ for $0 \leq t \leq T$, namely $y_{[0,T]}$
  - Sample q different trajectories of the processes $(\mathbf{V}, w)$: $(\mathbf{V}^1, w^1), (\mathbf{V}^2, w^2) \ldots, (\mathbf{V}^q, w^q)$
  - Compute $\mathcal{L}(NN^{\theta}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})$ for each sampled $(\mathbf{V}, w)$ and compute $\mathbb{E}\left[\mathcal{L}(NN^{\theta}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})|\mathcal{Y}_{0:T}\right]$ by $\frac{1}{q}\sum_{j=1}^{q}\mathcal{L}(NN^{\theta}, \mathbf{V}^j_{[0,T]}, w^j_{[0,T]}, y_{[0,T]})$

Loss Function Computation

- **Training**: Compute the loss function by sampling

$$\text{Loss}(\theta) = \mathbb{E}\left[\mathbb{E}\left[\mathcal{L}(NN^{\theta}_{g,y_{[0,T]}}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})\Big|\mathcal{Y}_{0:T}\right]\right]$$

  where $\mathcal{Y}_{0:T}$ is the filtration generated by the observation process from time $t = 0$ to $t = T$.
- How to practically compute the expectations in the loss function given the inaccessibility to all the trajectories?
- Strategy: Monte-Carlo methods
  - Fix a simulation of $\mathbf{Y}(t)$ for $0 \leq t \leq T$, namely $y_{[0,T]}$
  - Sample q different trajectories of the processes $(\mathbf{V}, w)$: $(\mathbf{V}^1, w^1), (\mathbf{V}^2, w^2) \ldots, (\mathbf{V}^q, w^q)$
  - Compute $\mathcal{L}(NN^{\theta}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})$ for each sampled $(\mathbf{V}, w)$ and compute $\mathbb{E}\left[\mathcal{L}(NN^{\theta}, \mathbf{V}_{[0,T]}, w_{[0,T]}, y_{[0,T]})|\mathcal{Y}_{0:T}\right]$ by $\frac{1}{q}\sum_{j=1}^{q}\mathcal{L}(NN^{\theta}, \mathbf{V}^j_{[0,T]}, w^j_{[0,T]}, y_{[0,T]})$
  - Repeat the whole procedure for different sampled simulations of $\mathbf{Y}$ to get an approximation of the outer expectation of the loss function

Conclusion Remarks

- Stochastic Filtering is important to pave the way for **real-time** estimations of cellular states

Conclusion Remarks

- Stochastic Filtering is important to pave the way for **real-time** estimations of cellular states
- We have developed a method which exploits the power of Machine Learning to tackle high dimensional problems by extending the DeepCME

Conclusion Remarks

- Stochastic Filtering is important to pave the way for **real-time** estimations of cellular states
- We have developed a method which exploits the power of Machine Learning to tackle high dimensional problems by extending the DeepCME
- The methods is versatile and does not need simulations for the prediction step

Conclusion Remarks

- Stochastic Filtering is important to pave the way for **real-time** estimations of cellular states
- We have developed a method which exploits the power of Machine Learning to tackle high dimensional problems by extending the DeepCME
- The methods is versatile and does not need simulations for the prediction step
- Implementation (**in progress**)

Conclusion Remarks

- Stochastic Filtering is important to pave the way for **real-time** estimations of cellular states
- We have developed a method which exploits the power of Machine Learning to tackle high dimensional problems by extending the DeepCME
- The methods is versatile and does not need simulations for the prediction step
- Implementation (**in progress**)
- The current DNN can only make predictions for observation process trajectories whose maximal length cannot exceed the ones saw in the training

Conclusion Remarks

- Stochastic Filtering is important to pave the way for **real-time** estimations of cellular states
- We have developed a method which exploits the power of Machine Learning to tackle high dimensional problems by extending the DeepCME
- The methods is versatile and does not need simulations for the prediction step
- Implementation (**in progress**)
- The current DNN can only make predictions for observation process trajectories whose maximal length cannot exceed the ones saw in the training
- Try to make predictions at intermediate times $t \leq T$

References

- Ankit Gupta, Christoph Schwab, and Mustafa Khammash. “DeepCME: A deep learning framework for computing solution statistics of the chemical master equation”. In: *PLoS computational biology* 17.12 (2021), e1009623
- Elena Sofia D’Ambrosio et al. “Filtered finite state projection method for the analysis and estimation of stochastic biochemical reaction networks”. In: *bioRxiv* (2022)
- Zhou Fang, Ankit Gupta, and Mustafa Khammash. “A scalable approach for solving chemical master equations based on modularization and filtering”. In: *bioRxiv* (2022), pp. 2022–10
- Zhou Fang, Ankit Gupta, and Mustafa Khammash. “Stochastic filtering for multiscale stochastic reaction networks based on hybrid approximations”. In: *Journal of Computational Physics* 467 (2022), p. 111441
- Zhou Fang, Ankit Gupta, and Mustafa Khammash. “Stochastic filters based on hybrid approximations of multiscale stochastic reaction networks”. In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 4616–4621

ETH zürich

Thanks for the attention!

Elena Sofia D'Ambrosio
PhD Student
Control Theory And Systems Biology Laboratory
elena.dambrosio@bsse.ethz.ch

ETH Zürich
Department of Biosystems Science and Engineering
Mattenstrasse 26
4058 Basel, Switzerland
https://bsse.ethz.ch/

We acknowledge funding from the Swiss National Science Foundation under grant 182653.