Learning Stochastic Behaviour from Aggregate Data

Shaojun Ma [1] **Shu Liu** [1] **Hongyuan Zha** [2] **Haomin Zhou** [1]

Abstract

Learning nonlinear dynamics from aggregate data is a challenging problem because the full trajectory of each individual is not available, namely, the individual observed at one time may not be observed at the next time point, or the identity of individual is unavailable. This is in sharp contrast to learning dynamics with full trajectory data, on which the majority of existing methods are based. We propose a novel method using the weak form of Fokker Planck Equation (FPE) — a partial differential equation — to describe the density evolution of data in a sampled form, which is then combined with Wasserstein generative adversarial network (WGAN) in the training process. In such a sample-based framework we are able to learn the nonlinear dynamics from aggregate data without explicitly solving FPE. We demonstrate our approach in the context of a series of synthetic and real-world data sets.

1. Introduction

In the context of dynamical systems, **Aggregate data** refers to a data format in which the full trajectory of each individual modeled by the evolution of state is not available, but rather a sample from the distribution of state at a certain time point is available. Typical examples include data sets collected for DNA evolution, social gathering, density in control problems, and bird migration, during the evolution of which it is impossible to follow an individual inter-temporally. In those applications, some observed individuals at one time point may be un-observable at the next time spot, or when the individual identities are blocked or unavailable due to various technical and ethical reasons. Rather than inferring the exact information for each individual, the main objective of learning dynamics in aggregate data is to recover and predict the evolution of distribution of all individuals together. **Trajectory data**, in contrast, is a kind of data that we are able to acquire the information of each individual all the time. Although some studies also considered the case that partial trajectories are missing, the identities of those individuals, whenever they are observable, are always assumed available. For example, stock price, weather, customer behaviors and most training data sets for computer vision and natural language processing are considered as trajectory data. There are many existing models to learn dynamics of full-trajectory data. Typical ones include Hamiltonian neural networks (Greydanus et al., 2019), Hidden Markov Model (HMM) (Alshamaa et al., 2019; Eddy, 1996), Kalman Filter (KF) (Farahi & Yazdi, 2020; Harvey, 1990; Kalman, 1960) and Particle Filter (PF) (Santos et al., 2019; Djuric et al., 2003), as well as the models built upon HMM, KF and PF (Deriche et al., 2020; Fang et al., 2019; Hefny et al., 2015; Langford et al., 2009). They require full trajectories of each individual, which may not be applicable in the aggregate data situations. On the other side, only a few methods are proposed on aggregated data in the recent learning literature. In the work of Hashimoto et al. (2016), authors assumed that the hidden dynamic of particles follows a stochastic differential equation (SDE), in particular, they used a recurrent neural network to parameterize the drift term. Furthermore, Wang et al. (2018) improved traditional HMM model by using an SDE to describe the evolving process of hidden states and Singh et al. (2020) updated HMM parameters through aggregate observations.

We propose to learn the dynamics of density through the weak form of Fokker Planck Equation (FPE), which is a parabolic partial differential equation (PDE) governing many dynamical systems subject to random noise perturbations, including the typical SDE models in existing studies. Our learning is accomplished by minimizing the Wasserstein distance between predicted distribution given by FPE and the empirical distribution from data samples. Meanwhile we utilize neural networks to handle higher dimensional cases. More importantly, by leveraging the framework of Wasserstein Generative Adversarial Network (WGAN) (Arjovsky et al., 2017), our model is capable of approximating the distribution of samples at different time points without solving the SDE or FPE. More specifically, we treat the drift

[1]Department of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA [2]School of Data Science, Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong, Shenzhen, China, the research of Hongyuan Zha is supported in part by a grant from Shenzhen Research Institute of Big Data. Correspondence to: Shaojun Ma <shaojunma@gatech.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

coefficient, the goal of learning, in the FPE as a generator, and the test function in the weak form of FPE as a discriminator. In other words, our method can also be regarded as a data-driven method to estimate transport coefficient in FPE, which corresponds to the drift terms in SDEs. Additionally, though we treat diffusion term as a constant in our model, it is straightforward to generalize it to be a neural network as well, which can be an extension of this work. We would like to mention that several methods of solving SDE and FPE (Weinan et al., 2017; Beck et al., 2018; Li et al., 2019) adopt opposite ways to our method, they utilize neural networks to estimate the distribution $P(x,t)$ with given drift and diffusion terms.

In conclusion, our contributions are: 1) We develop an algorithm that learns the drift term of a SDE via minimizing the Wasserstein discrepancy between the observed aggregate data and our generated data. 2) By leveraging a *weak* form of FPE, we are able to compute the Wasserstein distance directly without solving the FPE. 3) Finally, we demonstrate the accuracy and the effectiveness of our algorithm via several synthetic and real-world examples.

2. Proposed Method

2.1. Fokker Planck Equation for the density evolution

We assume the individuals evolve in a pattern in the space $\mathbb{R}^D$ as shown in Figure 1. One example satisfying such process is the stochastic differential equation(SDE), which is also known as the Itô process (Øksendal, 2003): $d\boldsymbol{X}_t = g(\boldsymbol{X}_t,t)dt + \sigma d\boldsymbol{W}_t$. Here $d\boldsymbol{X}_t$ represents an infinitesimal change of $\{\boldsymbol{X}_t\}$ along with time increment dt, $g(\cdot,t) = (g^1(\cdot,t),...,g^D(\cdot,t))^T$ is the drift term (drifting vector field) that drives the dynamics of SDE, σ is the diffusion constant, $\{\boldsymbol{W}_t\}$ is the standard Brownian Motion.

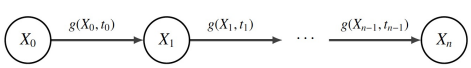


Figure 1. State model of the stochastic process X_t

The probability density of $\{\boldsymbol{X}_t\}$ is governed by the Fokker Planck Equation(FPE) (Risken & Caugheyz, 1991):

Lemma 1. *Suppose $\{\boldsymbol{X}_t\}$ solves the SDE $d\boldsymbol{X}_t = g(\boldsymbol{X}_t,t)dt + \sigma d\boldsymbol{W}_t$, denote $p(\cdot,t)$ as the probability density of the random variable $\boldsymbol{X}_t$. Then $p(x,t)$ solves the following equation:*

$$\frac{\partial p(\boldsymbol{x},t)}{\partial t} = \sum_{i=1}^{D} -\frac{\partial}{\partial x_i}\left[g^i(\boldsymbol{x},t)p(\boldsymbol{x},t)\right] + \frac{1}{2}\sigma^2\sum_{i=1}^{D}\frac{\partial^2}{\partial {x_i}^2}p(\boldsymbol{x},t). \quad (1)$$

As a linear evolution PDE, FPE describes the evolution of density function of the stochastic process driven by a SDE. Due to this reason, FPE plays a crucial role in stochastic calculus, statistical physics and modeling (Nelson, 1985; Qi & Majda, 2016; Risken, 1989). Its importance is also drawing more attention among statistic and machine learning communities (Liu & Wang, 2016; Pavon et al., 2018; Rezende & Mohamed, 2015). In this paper, we utilize the weak form of FPE as a basis to study hidden dynamics of the time evolving aggregated data without solving FPE.

Our task can be described as: assume that the individuals evolve with the process indicated by Figure 1, which can be simulated by Itô process. Then given observations $\boldsymbol{x}_t$ along time axis, we aim to recover the drift coefficient $g(\boldsymbol{x},t)$ in FPE, and thus we are able to recover and predict the density evolution of such dynamic. For simplicity we treat $g(\boldsymbol{x},t)$ as a function uncorrelated to time t, namely, $g(\boldsymbol{x},t) = g(\boldsymbol{x})$. Notice that though evolving process of individuals can be simulated by Itô process, in reality since we lose identity information of individuals, the observed data become aggregate data, thus we need a new way other than traditional methods to study the swarm's distribution.

We also remark that in the work of Hashimoto et al. (2016), based on Jordan-Kinderlehrer-Otto (JKO) (Jordan et al., 1998) theorem, they utilize RNN to approximate potential function and measure Sinkhorn distance (an approximation to Wasserstein distance). In our work, we assume that the density follows Fokker Planck equation, but we don't solve it directly. Instead, we take the weak form of Fokker Planck equation and compute everything in sample form, which coincides with a similar form of WGAN at the observations. Particularly, we treat FPE as the dynamic regularizer for the marginal fitting problem, therefore is fundamentally different from previous methods. As a byproduct, our numerical scheme allows to freely choose the time step Δt, which is not restricted to the given time stamp of observations. Δt is used to control the error bound.

2.2. Weak Form of Fokker Planck Equation

Given FPE stated in Lemma 1, if we multiply a test function $f \in H_0^1(\mathbb{R}^D)$ on both sides of the FPE, where $H_0^1(\mathbb{R}^D)$ denote the Sobolev space. Integration on both sides:

$$\int \frac{\partial p}{\partial t} f(\boldsymbol{x})d\boldsymbol{x} = \int \sum_{i=1}^{D} -\frac{\partial}{\partial x_i}\left[g^i(\boldsymbol{x})p(\boldsymbol{x},t)\right] f(\boldsymbol{x})d\boldsymbol{x} + \frac{1}{2}\sigma^2 \int \sum_{i=1}^{D}\frac{\partial^2}{\partial {x_i}^2}p(\boldsymbol{x},t)f(\boldsymbol{x})d\boldsymbol{x}.$$

$$\int \frac{\partial p}{\partial t} f(\boldsymbol{x}) d\boldsymbol{x} = \int \sum_{i=1}^{D} g^i(\boldsymbol{x}) \frac{\partial}{\partial x_i} f(\boldsymbol{x}) p(\boldsymbol{x}, t) d\boldsymbol{x} + \frac{1}{2}\sigma^2 \int \sum_{i=1}^{D} \frac{\partial^2}{\partial {x_i}^2} f(\boldsymbol{x}) p(\boldsymbol{x}, t) d\boldsymbol{x}.$$

The first advantage of weak solution is that the solution of a PDE usually requires strong regularity and thus may not exist in the classical sense for a certain group of equations, however, the weak solution has fewer regularity requirements and thus their existence are guaranteed for a much larger classes of equations. The second advantage is that the weak formulation may provide new perspectives for numerically solving PDEs (Zienkiewicz & Cheung, 1971; Sirignano & Spiliopoulos, 2018; Zang et al., 2019).

Suppose the observed samples at time points t_{m-1} and t_m follow the true densities $\hat{p}(\cdot, t_{m-1})$ and $\hat{p}(\cdot, t_m)$ respectively. Let's consider the following SDE:

$$d\tilde{\boldsymbol{X}}_t = g_\omega(\tilde{\boldsymbol{X}}_t)dt + \sigma d\boldsymbol{W}_t, \quad \text{where} \quad t_{m-1} \le t \le t_m, \quad \tilde{\boldsymbol{X}}_{t_{m-1}} \sim \hat{p}(\cdot, t_{m-1}). \tag{2}$$

Here g_ω is an approximation to the real drift term g. In our research, we treat g_ω as a neural network with parameters ω. Stochastic process $\tilde{\boldsymbol{X}}_t$ has a density function, denoted by $\tilde{p}(\cdot, t)$, which is different from the observed density. Hence, it is natural to compute and minimize the discrepancy between the approximated density $\tilde{p}(\cdot, t_m)$ and true density $\hat{p}(\cdot, t_m)$, within which we optimize g_ω and thus recover the true drift term g.

In our research, we choose the Wasserstein-1 distance as our discrepancy function (Villani, 2008) (Arjovsky et al., 2017). Applying Kantorovich-Rubinstein duality (Villani, 2008) leads to $W_1(\hat{p}(\cdot, t_m), \tilde{p}(\cdot, t_m)) =$

$$\sup_{\|\nabla f\| \le 1} \left\{ \mathbb{E}_{\boldsymbol{x}_r \sim \hat{p}(\boldsymbol{x}, t_m)}[f(\boldsymbol{x}_r)] - \mathbb{E}_{\boldsymbol{x}_g \sim \tilde{p}(\boldsymbol{x}, t_m)}[f(\boldsymbol{x}_g)] \right\}.$$

The first term $\mathbb{E}_{\boldsymbol{x}_r \sim \hat{p}(\boldsymbol{x}, t_m)}[f(\boldsymbol{x}_r)]$ can be conveniently computed by Monte-Carlo method since we are already provided with the real data points $\boldsymbol{x}_r \sim \hat{p}(\cdot, t_m)$. To evaluate the second term, we first approximate $\tilde{p}(\cdot, t_m)$ by trapezoidal rule (Atkinson, 2008): $\tilde{p}(\boldsymbol{x}, t_m) \approx$

$$\hat{p}(\boldsymbol{x}, t_{m-1}) + \frac{\Delta t}{2} \left(\frac{\partial \hat{p}(\boldsymbol{x}, t_{m-1})}{\partial t} + \frac{\partial \tilde{p}(\boldsymbol{x}, t_m)}{\partial t} \right), \tag{3}$$

where $\Delta t = t_m - t_{m-1}$. Then we compute:

$$\mathbb{E}_{\boldsymbol{x}_g \sim \tilde{p}(\cdot, t_m)}[f(\boldsymbol{x}_g)] \approx \int f(\boldsymbol{x}) \hat{p}(\boldsymbol{x}, t_{m-1}) d\boldsymbol{x} + \frac{\Delta t}{2} \left(\int \frac{\partial \hat{p}(\boldsymbol{x}, t_{m-1})}{\partial t} f(\boldsymbol{x}) d\boldsymbol{x} + \int \frac{\partial \tilde{p}(\boldsymbol{x}, t_m)}{\partial t} f(\boldsymbol{x}) d\boldsymbol{x} \right). \tag{4}$$

In the above Equation (4), the second and the third term on the right-hand side can be reformulated via the weak form of FPE. This gives us a new formulation for $W_1(\hat{p}(\cdot, t_m), \tilde{p}(\cdot, t_m))$, which can by computed by using Monte-Carlo method. In fact, the first and the second terms in (4) can be directly computed via data points from $\hat{p}(\cdot, t_{m-1})$. For the third term, we need to generate samples from $\tilde{p}(\cdot, t_m)$. To achieve this, we apply Euler-Maruyama scheme (Kloeden & Platen, 2013) to SDE (2) in order to acquire our desired samples $\tilde{\boldsymbol{x}}_{t_m}$:

$$\tilde{\boldsymbol{x}}_{t_m} = \hat{\boldsymbol{x}}_{t_{m-1}} + g_\omega(\hat{\boldsymbol{x}}_{t_{m-1}})\Delta t + \sigma\sqrt{\Delta t}\boldsymbol{z}, \quad \text{where} \quad \boldsymbol{z} \sim \mathcal{N}(0, I), \quad \hat{\boldsymbol{x}}_{t_{m-1}} \sim \hat{p}(\cdot, t_{m-1}). \tag{5}$$

Here $\mathcal{N}(0, I)$ is the standard Gaussian distribution on $\mathbb{R}^D$. Now we summarize these results in Proposition 1:

Proposition 1. *For a set of points $X = \{\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(N)}\}$ in $\mathbb{R}^D$. We denote $\mathcal{F}_f(X)$ as:*

$$\frac{1}{N} \sum_{k=1}^{N} \left(\sum_{i=1}^{D} g_\omega^i(\boldsymbol{x}^{(k)}) \frac{\partial}{\partial x_i} f(\boldsymbol{x}^{(k)}) + \frac{1}{2}\sigma^2 \sum_{i=1}^{D} \frac{\partial^2}{\partial x_i^2} f(\boldsymbol{x}^{(k)}) \right),$$

then at time point t_m, the Wasserstein distance between $\hat{p}(\cdot, t_m)$ and $\tilde{p}(\cdot, t_m)$ can be approximated by:

$$W_1(\hat{p}(\cdot, t_m), \tilde{p}(\cdot, t_m)) \approx \sup_{\|\nabla f\| \le 1} \left\{ \frac{1}{N} \sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_m}^{(k)}) - \frac{1}{N} \sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_{m-1}}^{(k)}) - \frac{\Delta t}{2} \left(\mathcal{F}_f(\hat{X}_{m-1}) + \mathcal{F}_f(\tilde{X}_m) \right) \right\}.$$

Here $\{\hat{\boldsymbol{x}}_{t_{m-1}}^{(k)}\} \sim \hat{p}(\cdot, t_{m-1})$, $\{\hat{\boldsymbol{x}}_{t_m}^{(k)}\} \sim \hat{p}(\cdot, t_m)$. We denote $\hat{X}_{m-1} = \{\hat{\boldsymbol{x}}_{t_{m-1}}^{(1)}, ..., \hat{\boldsymbol{x}}_{t_{m-1}}^{(N)}\}$, $\tilde{X}_m = \{\tilde{\boldsymbol{x}}_{t_m}^{(1)}, ..., \tilde{\boldsymbol{x}}_{t_m}^{(N)}\}$, where each $\tilde{\boldsymbol{x}}_{t_m}^{(k)}$ is computed by Euler-Maruyama scheme.

2.3. Wasserstein Distance on Time Series

In real cases, it is not realistic to observe the data at arbitrary two consecutive time nodes, especially when Δt is small. To make our model more flexible, we extend our formulation so that we are able to plug in observed data at arbitrary time points. To be more precise, suppose we observe data set $\hat{X}_{t_n} = \{\hat{\boldsymbol{x}}_{t_n}^{(1)}, ..., \hat{\boldsymbol{x}}_{t_n}^{(N)}\}$ at $J+1$ different time points $t_0, t_1, ..., t_J$. And we denote the generated data set as $\tilde{X}_{t_n} = \{\tilde{\boldsymbol{x}}_{t_n}^{(1)}, ..., \tilde{\boldsymbol{x}}_{t_n}^{(N)}\}$, here each $\tilde{\boldsymbol{x}}_{t_n}^{(\cdot)}$ is derived from the n-step Euler-Maruyama scheme:

$$\tilde{\boldsymbol{x}}_{t_j} = \tilde{\boldsymbol{x}}_{t_{j-1}} + g_\omega(\tilde{\boldsymbol{x}}_{t_{j-1}})\Delta t + \sigma\sqrt{\Delta t}\boldsymbol{z}, \quad \text{where} \quad \boldsymbol{z} \sim \mathcal{N}(0, I), \quad 0 \le j \le n, \quad \tilde{\boldsymbol{x}}_{t_0} \sim \hat{p}(\cdot, t_0). \tag{6}$$

Let us denote $\tilde{p}(\cdot, t)$ as the solution to FPE (1) with g replaced by g_ω and with initial condition $\tilde{p}(\cdot, t_0) = \hat{p}(\cdot, t_0)$,

then the approximation formula for evaluating the Wasserstein distance $W_1(\hat{p}(\cdot,t_n),\tilde{p}(\cdot,t_n))$ is provided in the following proposition:

Proposition 2. *Suppose we keep all the notations defined as above, then we have the approximation:*

$$W_1(\hat{p}(\cdot,t_n),\tilde{p}(\cdot,t_n)) \approx \sup_{\|\nabla f\|\le 1}\Bigg\{\frac{1}{N}\sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f(\hat{\boldsymbol{x}}_{t_0}^{(k)}) - \frac{\Delta t}{2}\Bigg(\mathcal{F}_f(\hat{X}_0)+\mathcal{F}_f(\tilde{X}_n) + 2\sum_{s=1}^{n-1}\mathcal{F}_f(\tilde{X}_s)\Bigg)\Bigg\}.$$

Minimizing the Objective Function: Base on Proposition 2, we obtain objective function by summing up the accumulated Wasserstein distances among J observations along the time axis. Thus, our goal is to minimize the following objection function:

$$\min_{g_\omega}\Bigg\{\sum_{n=1}^{J}\sup_{\|\nabla f_n\|\le 1}\Bigg\{\frac{1}{N}\sum_{k=1}^{N} f_n(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f_n(\hat{\boldsymbol{x}}_{t_0}^{(k)}) - \frac{\Delta t}{2}\Bigg(\mathcal{F}_{f_n}(\hat{X}_0)+\mathcal{F}_{f_n}(\tilde{X}_n) + 2\sum_{s=1}^{n-1}\mathcal{F}_{f_n}(\tilde{X}_s)\Bigg)\Bigg\}\Bigg\}.$$

Notice that since we have observations on J distinct time points, for each time point we compute Wasserstein distance with the help of the dual function f_n, thus we involve J test functions in total. In our actual implementation, we will choose these dual functions as neural networks. We call our algorithm Fokker Planck Process(FPP), the entire procedure is shown in Algorithm 1. We also provide an error analysis in Appendix.

Remark 1. *When the time interval $\Delta t = t_j - t_i$ between two observations at X_i and $X_j (i<j)$ is large. In order to guarantee the accuracy of $\tilde{X}_s$, we can separate Δt into multiple smaller intervals, namely, $\Delta t = Kh$, where K the number of intervals and h is the interval length. Then we evaluate (6) on the finer meshes to obtain more accurate samples $\{\tilde{x}_s^{(1)},...,\tilde{x}_s^{(N)}\}$ at specific time s.*

Remark 2. *The drift function recovered by our framework may not be unique, see Section 4 for more details.*

3. Experiments

In this section, we evaluate our model on various synthetic and realistic data sets by employing Algorithm 1. We generate samples $\tilde{\boldsymbol{x}}_t$ and make all predictions base on Equation (5) starting with $\hat{\boldsymbol{x}}_0$.

Baselines: We compare our model with two recently proposed methods. One model (NN) adopts recurrent neural network(RNN) to learn dynamics directly from observations of aggregate data (Hashimoto et al., 2016). The other one model (LEGEND) learns dynamics in a HMM framework (Wang et al., 2018). The baselines in our experiments are two typical representatives that have state-of-the-art performance on learning aggregate data. Furthermore, though we simulate the evolving process of the data as a SDE, which is on the same track with NN, as mentioned before, NN trains its RNN via optimizing Sinkhorn distance (Cuturi, 2013), our model starts with a view of weak form of PDE, focuses more on WGAN framework and easier computation.

Algorithm 1 Fokker Planck Process Algorithm

Require: Initialize f_{θ_n} $(1\le n\le J)$, g_ω
Require: Set ϵ_{f_n} as the inner loop learning rate for f_{θ_n} and ϵ_g as the outer loop learning rate for g_ω
for # training iterations **do**
  for k steps **do**
    for observed time t_s in $\{t_1,...,t_J\}$ **do**
      Compute the generated data set $\tilde{X}_{t_s}$ from Euler-Maruyama scheme (6) for $1\le s\le J$
      Acquire data sets $\hat{X}_{t_s}=\{\hat{\boldsymbol{x}}_{t_s}^{(1)},...,\hat{\boldsymbol{x}}_{t_s}^{(N)}\}$ from real distribution $\hat{p}(\cdot,t_s)$ for $1\le s\le J$
    end for
    For each dual function f_{θ_n}, compute: $\mathcal{F}_n = \mathcal{F}_{f_{\theta_n}}(\hat{X}_{t_0})+\mathcal{F}_{f_{\theta_n}}(\tilde{X}_{t_n})+2\sum_{s=1}^{n-1}\mathcal{F}_{f_{\theta_n}}(\tilde{X}_{t_s})$
    Update each f_{θ_n} by:
    $\theta_n \leftarrow \theta_n + \epsilon_{f_n}\nabla_\theta\Big(\frac{1}{N}\sum_{k=1}^{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N}\sum_{k=1}^{N} f_{\theta_n}(\hat{\boldsymbol{x}}_{t_0}^{(k)}) - \frac{\Delta t}{2}\mathcal{F}_n\Big)$
  end for
  Update g_ω by:
  $\omega \leftarrow \omega - \epsilon_g\nabla_\omega\Big(\sum_{n=1}^{J}\big(\frac{1}{N}f_{\theta_n}(\hat{\boldsymbol{x}}_{t_n}^{(k)}) - \frac{1}{N}f_{\theta_n}(\hat{\boldsymbol{x}}_{t_0}^{(k)}) - \frac{\Delta t}{2}\mathcal{F}_n\big)\Big)$
end for

3.1. Synthetic Data

We first evaluate our model on three synthetic data sets which are generated by three artificial dynamics: **Synthetic-1**, **Synthetic-2** and **Synthetic-3**.

Experiment Setup: In all synthetic data experiments, we set the drift term g and the discriminator f as two simple fully-connected networks. The g network has one hidden layer and the f network has three hidden layers. Each layer has 32 nodes for both g and f. The only one activation function we choose is Tanh. Notice that since we need to calculate $\frac{\partial^2 f}{\partial x^2}$, the activation function of f must be twice differentiable to avoid loss of weight gradient. In terms of

training process, we use the Adam optimizer (Kingma & Ba, 2014) with learning rate 10^{-4}. Furthermore, we use spectral normalization to realize $\|\nabla f\| \leq 1$(Miyato et al., 2018). We initialize the weights with Xavier initialization(Glorot & Bengio, 2010) and train our model by Algorithm 1. We set the data size at each time point is $N = 2000$, treat 1200 data points as the training set and the other 800 data points as the test set, Δt is set to be 0.01.

Synthetic-1:

$$\hat{\boldsymbol{x}}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0),$$
$$\hat{\boldsymbol{x}}_{t+\Delta t} = \hat{\boldsymbol{x}}_t - (\boldsymbol{A}\hat{\boldsymbol{x}}_t + \boldsymbol{b})\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1).$$

In Synthetic-1, the data is following a simple linear dynamic, we set $\boldsymbol{A} = [(4,0),(0,1)], \boldsymbol{b} = [-12, \ -12]^T$, $\sigma = 1$, $\boldsymbol{\Sigma}_0 = \boldsymbol{I_2}$. We utilize true $\boldsymbol{x}_0$, $\boldsymbol{x}_{20}$ and $\boldsymbol{x}_{200}$ in training process and predict the distributions of $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{50}$ and $\boldsymbol{x}_{500}$. As visualized in Figure 2, from (a) to (c), the generated data(blue) covers all areas of ground truth(red), the original Gaussian distribution converges to the target Gaussian distribution as we expect.

Synthetic-2:

$$\hat{\boldsymbol{x}}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0), \quad \hat{\boldsymbol{x}}_{t+\Delta t} = \hat{\boldsymbol{x}}_t - \boldsymbol{G}\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1),$$

where $\boldsymbol{G}$ is given in Appendix.

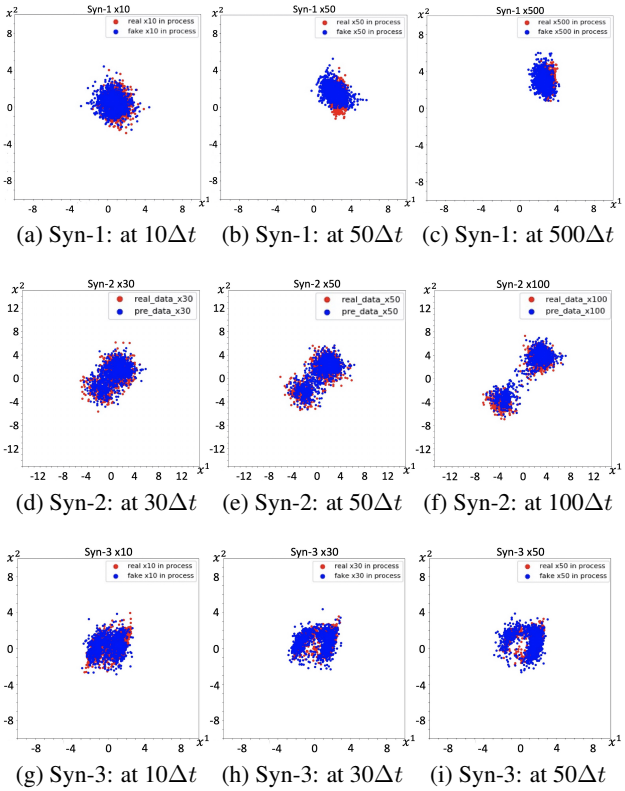
In Synthetic-2, the data is following a complex nonlinear dynamic. We let $\sigma = \sigma_1 = \sigma_2 = 4$, $\mu_1 = [12, 15]^T$ and $\mu_2 = [-15, -15]^T$(defined in Appendix). We utilize true $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{40}$ and $\boldsymbol{x}_{80}$ in training process and predict $\boldsymbol{x}_{30}$, $\boldsymbol{x}_{50}$ and $\boldsymbol{x}_{100}$. The results are shown in Figure 2, from (d) to (f), the generated data(blue) covers all areas of ground truth(red), generated samples split and converge to a mixed Gaussian as the ground truth suggests.

Synthetic-3 (Nonlinear Van der Pol oscillator (Li, 2018):)

$$\hat{\boldsymbol{x}}_0 \sim \mathcal{N}(0, \boldsymbol{\Sigma}_0),$$
$$\hat{x}^1_{t+\Delta t} = \hat{x}^1_t + 10\left(\hat{x}^2_t - \frac{1}{3}(\hat{x}^1_t)^3 + \hat{x}^1_t\right)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1),$$
$$\hat{x}^2_{t+\Delta t} = \hat{x}^2_t + 3(1 - \hat{x}^1_t)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1).$$

In Synthetic-3, we let $\sigma = 1$ and utilize true $\boldsymbol{x}_3$, $\boldsymbol{x}_7$ and $\boldsymbol{x}_{20}$ in training process then predict the distributions of $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{30}$ and $\boldsymbol{x}_{50}$. As presented in Figure 2, from (g) to (i), the generated data(blue) covers all areas of ground truth(red), the distributions we predict are following the true stochastic oscillator's pattern.

Remark 3: In Syn-2 and Syn-3, $\hat{x}^i_t$ represents the i-th dimension of $\hat{\boldsymbol{x}}_t$. **We further state that** in Syn-1 and Syn-3, the training data is coming from the same $\boldsymbol{x}_0$ respectively. In Syn-2 the training data is coming from different $\boldsymbol{x}_0$, namely,



(a) Syn-1: at $10\Delta t$ (b) Syn-1: at $50\Delta t$ (c) Syn-1: at $500\Delta t$

(d) Syn-2: at $30\Delta t$ (e) Syn-2: at $50\Delta t$ (f) Syn-2: at $100\Delta t$

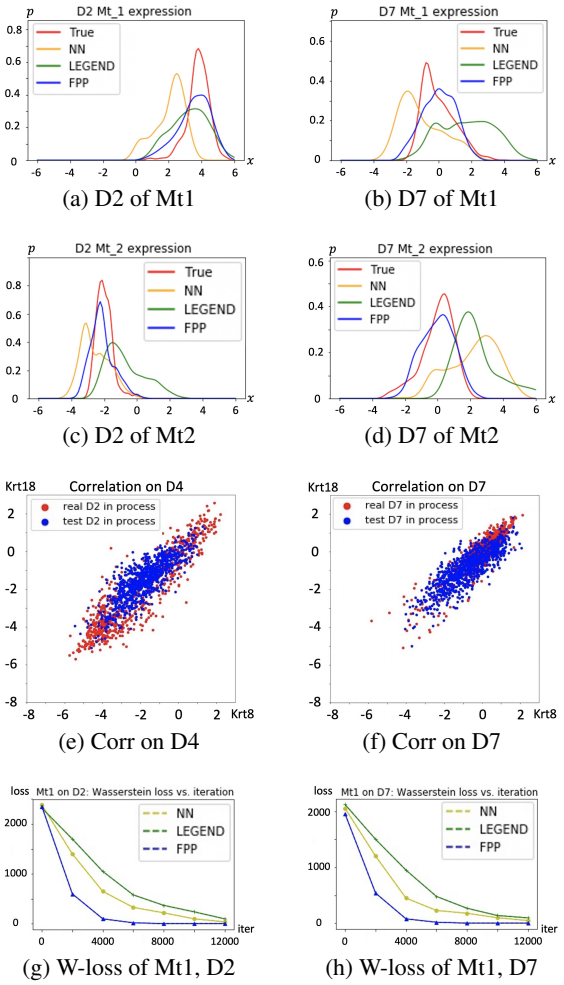
(g) Syn-3: at $10\Delta t$ (h) Syn-3: at $30\Delta t$ (i) Syn-3: at $50\Delta t$

Figure 2. Comparison of generated data(blue) and ground truth(red) of Synthetic-1((a) to (c)), Synthetic-2((d) to (f)) and Synthetic-3((g) to (i)). In each case, it finally converges to a stationary distribution.

the training data $\boldsymbol{x}_{10}$, $\boldsymbol{x}_{40}$ and $\boldsymbol{x}_{80}$ are generated from three different sets of $\boldsymbol{x}_0$. We also consider cases in higher dimensions: D = 6 and 10. To be more precise, we couple three 2-D dynamical systems to create the 6-D dynamical system and five 2-D systems to create the 10-D example. We compare our model with the two baseline models by using Wasserstein distance as error metric for the low-dimensional (D = 2) and high-dimensional (D = 6, 10) cases. As reported in Table 1, our model achieves lower Wasserstein error than the two baseline models in all cases. Clearly all the drift functions in the synthetic data sets cause the change of the distributions. In Section 4 we discuss a special case when the drift term does not change the distribution.

3.2. Realistic Data – RNA Sequence of Single Cell

In this section, we evaluate our model on a realistic biology data set called Single-cell RNA-seq(Klein et al., 2015), which is typically used for learning the evolvement of cell differentiation. The cell population begins to differentiate at day 0 (D0). Single-cell RNA-seq observations are then sampled at day 0 (D0), day 2 (D2), day 4 (D4) and day 7 (D7). At each time point, the expression of 24,175 genes



(a) D2 of Mt1 (b) D7 of Mt1

(c) D2 of Mt2 (d) D7 of Mt2

(e) Corr on D4 (f) Corr on D7

(g) W-loss of Mt1, D2 (h) W-loss of Mt1, D7

Figure 3. (a) to (d): The performance comparisions among different models on D2 and D7 of Mt1 and Mt2. (e) and (f): True (red) and predicted (blue) correlations between Mt1(x-axis) and Mt2(y-axis) on D2 (left) and D7 (right). (g) and (h): Wasserstein loss of Mt1 on D2 and D7 vs iterations.

of several hundreds cells are measured (933, 303, 683 and 798 cells on D0, D2, D4 and D7 respectively). Notice that there is only whole group's distribution but no trajectory information of each gene on different days. We pick 10 gene markers out of 24,175 to make a 10 dimensional data set. In the first task we treat gene expression at D0, D4 and D7 as training data to learn the hidden dynamic and predict the distribution of gene expression at D2. In the second task we train the model with gene expression at D0, D2 and D4, then predict the distribution of gene expression at D7. We plot the prediction results of two out of ten markers, i.e. Mt1 and Mt2 in Figure 3.

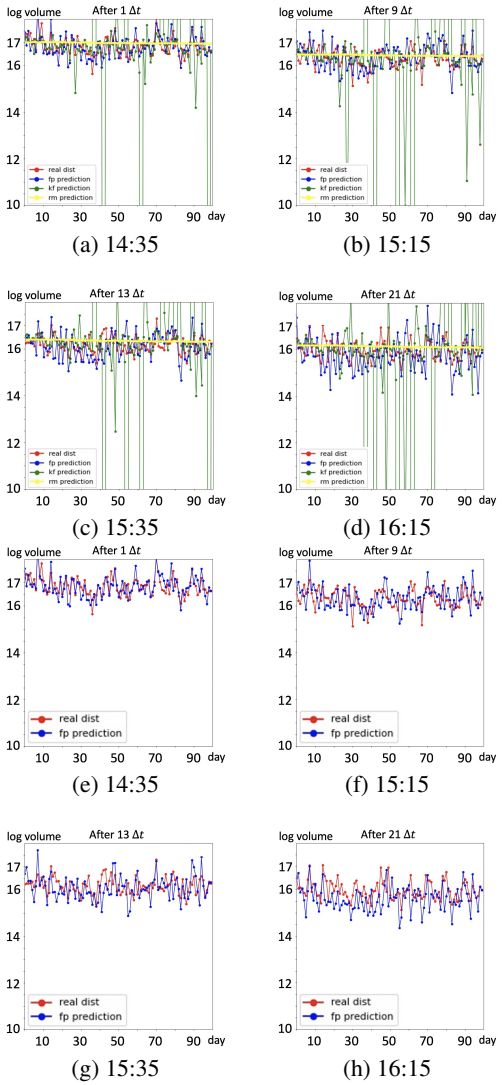
Experiment Setup: We set both f and g as fully connected three-hidden-layers neural networks, each layer has 64 nodes. The only activation function we choose is Tanh.

Table 1. The Wasserstein error of different models on Synthetic-1/2/3 and RNA-sequence data sets.

Data	Task	Dimension	NN	LEGEND	Ours
Syn–1	$\boldsymbol{x}_{50}$	2	1.37	0.44	**0.05**
		6	4.79	2.32	**0.06**
		10	9.13	2.89	**0.10**
	$\boldsymbol{x}_{500}$	2	0.84	0.18	**0.03**
		6	3.28	0.30	**0.03**
		10	8.05	1.79	**0.09**
Syn–2	$\boldsymbol{x}_{50}$	2	4.72	2.84	**0.02**
		6	6.47	5.33	**0.14**
		10	12.58	7.21	**0.22**
	$\boldsymbol{x}_{100}$	2	3.83	2.98	**0.04**
		6	8.83	3.17	**0.19**
		10	14.11	5.65	**0.32**
Syn–3	$\boldsymbol{x}_{30}$	2	4.13	1.29	**0.08**
		6	6.40	3.16	**0.17**
		10	11.76	8.53	**0.25**
	$\boldsymbol{x}_{50}$	2	3.05	0.87	**0.12**
		6	6.72	1.52	**0.16**
		10	9.81	3.55	**0.23**
RNA-Mt1	D2	10	33.86	10.28	**4.23**
	D7	10	12.69	7.21	**2.92**
RNA-Mt2	D2	10	31.45	13.32	**4.04**
	D7	10	11.58	7.89	**1.50**

The other setups of neural networks and training process are the same with the ones we use in Synthetic data. Notice that in realistic cases, Δt and $T/\Delta t$ become hyperparameters, here we choose $\Delta t = 0.05$, $T/\Delta t = 35$, which means the data evolves $10\Delta t$ from D0 to D2 , then $10\Delta t$ from D2 to D4 and finally $15\Delta t$ from D4 to D7. For preprocessing, we apply standard normalization procedures (Hicks et al., 2015) to correct batch effects and use non-negative matrix factorization to impute missing expression levels(Hashimoto et al., 2016; Wang et al., 2018).

Results: As shown in Table 1, when compared to other baselines, our model achieves lower Wasserstein error on both Mt1 and Mt2 data, which proves that our model is capable of learning the hidden dynamics of the two studied gene expressions. In Figure 3 (a) to (d), we visualized the predicted distributions of the two genes. The distributions of Mt1 and Mt2 predicted by our model (curves in blue) are closer to the true distributions (curves in red) on both D2 and D7. Furthermore, our model precisely indicates the correlations between Krt8 and Krt18 on D4 and D7, as shown in Figure 3 (e) and (f), which also demonstrates the effectiveness of our model since closer to the true correlation represents better performance (more results in Appendix). In Figure 3 (g) and (h), we see the training process of our model is easier with least computation time.



(a) 14:35 (b) 15:15 (c) 15:35 (d) 16:15 (e) 14:35 (f) 15:15 (g) 15:35 (h) 16:15

Figure 4. (a) to (d): Group A: with full trajectory of training data, predictions of traded volume in next 100 days, RM(yellow) fails to capture the regularities of traded volume in time series, kalman filter based model(green) fails to capture noise information and make reasonable predictions, our model(blue) is able to seize the movements of traded volume and yield better predictions. (e) to (h): Group B: predictions of our model without full trajectory.

3.3. Realistic Data – Daily Trading Volume

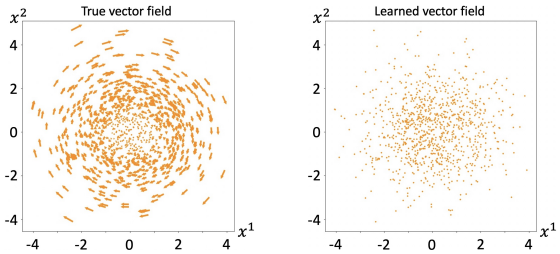
In this section we would like to demonstrate the performance of our model in financial area. Trading volume is the total quantity of shares or contracts traded for specified securities such as stocks, bonds, options contracts, future contracts and all types of commodities. It can be measured on any type of security traded during a trading day or a specified time period. In our case, daily volume of trade is measured on stocks. Predicting traded volume is an essential component in financial research since the traded volume, as a basic component or input of other financial algorithms, tells investors the market's activity and liquidity. The data set we use is the historical traded volume of the stock "JPM". The data covers period from January 2018 to January 2020 and is obtained from Bloomberg. Each day from 14:30 to 20:55, we have 1 observation every 5 minutes, totally 78 observations everyday. Our task is described as follows: we treat historical traded volume at 14:30, 14:40, 15:05, 15:20 and 16:20, namely, $\boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{x}_7, \boldsymbol{x}_{10}, \boldsymbol{x}_{22}$ as training data, each time point includes 730 samples. Then for next 100 days we predict traded volume at 14:35, 15:15, 15:35 and 16:15, namely, $\boldsymbol{x}_1, \boldsymbol{x}_9, \boldsymbol{x}_{13}, \boldsymbol{x}_{21}$. One of baselines we choose is classical rolling means(RM) method, which predicts intraday volume of a particular time interval by the average volume traded in the same interval over the past days. The other one baseline is a kalman filter based model (Chen et al., 2016) that outperforms all available models in predicting intrady trading volume.

Experiment Setup: Following similar setup as we did for RNA data set, we utilize the same structures for neural networks here. For hyperparameters we set $\Delta t = 0.02$, $T/\Delta t = 22$, it takes one single Δt from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$. For preprocessing, we rescale data by taking natural logarithm of trading volume, which is a common way in trading volume research. We conduct experiments on two groups(A&B) to show advantages of our method, for group A we train our model on complete data set, in this case the data has full trajectory; for group B we manually delete some trajectories of the data, for instance, we randomly kick out some samples of $\boldsymbol{x}_0, \boldsymbol{x}_2, \boldsymbol{x}_7, \boldsymbol{x}_{10}, \boldsymbol{x}_{22}$ then follow the same procedures of training and prediction.

Results: We present prediction results in Figure 4. As shown in first four figures, with full trajectory, prediction made by RM is almost a straight line, the prediction value bouncing up and down within a very small range, thus this model cannot capture the volume movements, namely, regularities existing in the time series; prediction made by the Kalman filter based model captures the regularities better than RM model, but it fails to deal with noise component existing in the time series, thus some predictions are out of a reasonable range. Traded volume predicted by our model is closer to the real case, moreover, our model captures regularities meanwhile gives stable predictions. Furthermore, without full trajectory, Kalman filter based model fails to be applied here and RM model still fails to capture the regularities, we randomly drop half of the training samples and display predictions made by our model in last four figures of Figure 4, we see our model still works well.

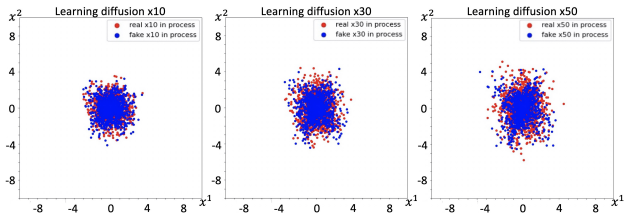
4. Discussions

In this section we discuss the limitations and extension of our model.



(a) True vector field (b) Learned vector field

Figure 5. Results of learning curl field



(a) Prediction at t_{10} (b) Prediction at t_{30} (c) Prediction at t_{50}

Figure 6. Results of learning diffusion function

The challenge for non-uniqueness: Mathematically it is impossible to recover the exact drift term of an SDE if we are only given the information of density evolution on certain time intervals, because there might be infinitely many drift functions to induce the same density evolution. More precisely, suppose $p(x,t)$ solves FPE (1), consider

$$0 = -\sum_{i=1}^{D} \frac{\partial}{\partial x_i}(u^i(\boldsymbol{x},t)p(\boldsymbol{x},t)) + \frac{\sigma^2}{2}\sum_{i=1}^{D}\frac{\partial^2}{\partial x_i^2}p(\boldsymbol{x},t).$$

One can prove, under mild assumptions, that there may be infinitely many vector fields $u(\boldsymbol{x},t) = (u^1(\boldsymbol{x},t),...,u^D(\boldsymbol{x},t))$ solving above equation. Therefore the solution to FPE (1) with drift term $g(\boldsymbol{x},t)+u(\boldsymbol{x},t)$ is still $p(\boldsymbol{x},t)$, i.e. the vector field $u(\boldsymbol{x},t)$ never affects the density evolution of the dynamic. This illustrates that given the density evolution $p(\cdot,t)$, the solution for drift term is not necessarily unique. This clearly poses an essential difficulty of determining the exact drift term from the density. In this study, the main goal is to recover the entire density evolution (i.e. interpolate the density between observation time points) and predict how the density evolves in the future. As a result, although we cannot always acquire the exact drift term of the dynamic, we can still accurately recover and predict the density evolution. This is still meaningful and may find its application in various scientific domains.

Curl field: The drift function we showed in the synthetic experiments will apparently cause the evolution of the distribution. If the drift function is a curl, namely $g = \nabla \times \boldsymbol{F}$, then the distribution does not change, under this situation we cannot learn the density evolution since our algorithm depends on the change of the whole distribution. To demonstrate this point of view, we simulate a curl field $(y,-x)$ induced by $\boldsymbol{A} = [0,10;-10,0]$ on a Gaussian distribution that mean$=(0,0)$, covariance$=[2,0;0,2]$. Here we set noise part as 0. As shown in Figure 5, true and learned vector fields are indicated in (a) and (b) respectively. We see that the learned vectors are all "points", meaning the length of the vectors is "0", the algorithm fails to recover true vector field.

Learning diffusion function: Our framework also works for learning unknown diffusion function in the Itô process. As an extension of our work, if we approximate the diffusion function with a neural network σ_η (with parameters η), we revise the operator $\mathcal{F}$ as:

$$\mathcal{F}_f(X) = \frac{1}{N}\sum_{k=1}^{N}\left(\sum_{i=1}^{D} g_\omega^i(\boldsymbol{x}^{(k)})\frac{\partial}{\partial x_i}f(\boldsymbol{x}^{(k)}) + \sum_{i=1}^{D}\left(\sum_{j=1}^{D}\frac{1}{2}(\sigma_\eta^{ij}(\boldsymbol{x}^{(k)}))^2\right)\frac{\partial^2}{\partial x_i^2}f(\boldsymbol{x}^{(k)})\right),$$

which can be derived by the same technique we used to derive Proposition 1.

We test this formulation on a synthetic data set, where we only consider diffusion influence, namely, drift term in Equation 1 is ignored. We set the ground truth of diffusion coefficient as $\boldsymbol{\sigma} = [(1,0),(0,2)]$. We design the neural network as a simple one fully connected layer with 32 nodes, then show our result in Figure 6, we see that the predictions(blue) follow the same patterns as the ground truth(red) does.

Future directions It worth mentioning that our proposed algorithm 1 requires the gradient with respect to the parameter ω of drift term g_ω (i.e. line 10 of Algorithm 1). Notice that each sample $\tilde{x}_{t_n}^{(k)}$ is computed from (6) for n steps, thus each sample $\tilde{x}_{t_n}^{(k)}$ can be treated as n compositions of drift term g_ω, which may lead to more expensive computation. However, we cann avoid direct computation of gradient ∇_ω by applying the adjoint method with Fokker-Planck equation (1) as the constraint (Pontryagin, 2018),(Zahr & Persson, 2016). This is one of our future research directions. Moreover, our model can also readily handle high dimensional cases by leveraging deep neural networks. Providing more numerical analysis such as compare trapezoidal rule and Runge-Kutta method in our scheme, as well as exploring super high dimensional applications are also appealing future directions.

5. Conclusion

In this paper, we formulate a novel method to recover the hidden dynamics from aggregate data. In particular, our

work shows one can simulate the evolving process of aggregate data as an Itô process, in order to investigate aggregate data, we derive a new model that employs the weak form of FPE as well as the framework of WGAN. Furthermore, in Appendix we prove the theoretical guarantees of the error bound of our model. Finally we demonstrate our model through experiments on three synthetic data sets and two real-world data sets.

References

Alshamaa, D., Chkeir, A., Mourad-Chehade, F., and Honeine, P. Hidden markov model for indoor trajectory tracking of elderly people. In *IEEE Sensors Applications Symposium (SAS)*, 2019.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein gan. In *arXiv preprint arXiv:1701.07875*, 2017.

Atkinson, K. E. *An introduction to numerical analysis*. John wiley & sons, 2008.

Beck, C., Becker, S., Grohs, P., Jaafari, N., and Jentzen, A. Solving stochastic differential equations and kolmogorov equations by means of deep learning. 2018.

Chen, R., Feng, Y., and Palomar, D. Forecasting intraday trading volume: a kalman filter approach. In *Available at SSRN 3101695*, 2016.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, 2013.

Deriche, M., Absa, A. A., Amin, A., and Liu, B. A novel approach for salt dome detection and tracking using a hybrid hidden markov model with an active contour model. *Journal of Electrical Systems*, 16(3):276–294, 2020.

Djuric, P. M., Kotecha, J. H., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M. F., and Miguez, J. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, 2003.

Eddy, S. R. Hidden markov models. *Current Opinion in Structural Biology*, 6(3):361–365, 1996.

Fang, Y., Wang, C., Yao, W., Zhao, X., Zhao, H., and Zha, H. On-road vehicle tracking using part-based particle filter. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4538–4552, 2019.

Farahi, F. and Yazdi, H. S. Probabilistic kalman filter for moving object tracking. *Signal Processing: Image Communication*, 82, 2020.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. *arXiv preprint arXiv:1906.01563*, 2019.

Harvey, A. C. *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, 1990.

Hashimoto, T., Gifford, D., and Jaakkola, T. Learning population-level diffusions with generative rnns. In *International Conference on Machine Learning*, pp. 2417–2426, 2016.

Hefny, A., Downey, C., and Gordon, G. J. Supervised learning for dynamical system learning. In *Neural Information Processing Systems*, 2015.

Hicks, S. C., Teng, M., and Irizarry, R. A. On the widespread and critical impact of systematic bias and batch effects in single-cell rna-seq data. *bioRxiv*, 2015.

Jordan, R., Kinderlehrer, D., and Otto, F. The variational formulation of the fokker–planck equation. *SIAM journal on mathematical analysis*, 29(1):1–17, 1998.

Kalman, R. E. A new approach to linear filtering and prediction problems. *arXiv preprint arXiv:1805.04099*, 1960.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

Klein, A., Mazutis, L., Akartuna, I., Tallapragada, N., Veres, A., Li, V., Peshkin, L., Weitz, D., and Kirschner, M. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015.

Kloeden, P. E. and Platen, E. *Numerical solution of stochastic differential equations*, volume 23. Springer Science & Business Media, 2013.

Langford, J., Salakhutdinov, R., and Zhang, T. Learning nonlinear dynamic models. In *International Conference on Machine Learning*, pp. 593–600, 2009.

Li, W., Liu, S., Zha, H., and Zhou, H. Parametric fokker-planck equation. In *Geometry science of information*, 2019.

Li, Y. A data-driven method for the steady state of randomly perturbed dynamics. *arXiv preprint arXiv:1805.04099*, 2018.

Liu, Q. and Wang, D. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Neural Information Processing Systems*, pp. 2378–2386, 2016.

Milstein, G. N. and Tretyakov, M. V. (eds.). *Stochastic numerics for mathematical physics*. Springer Science & Business Media, 2013.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Nelson, E. *Quantum fluctuations*. Princeton University Press, 1985.

Øksendal, B. Stochastic differential equations. In *Stochastic differential equations*, pp. 65–84. Springer, 2003.

Pavon, M., Tabak, E. G., and Trigila, G. The data-driven schroedinger bridge. *arXiv preprint arXiv:1806.01364*, 2018.

Pontryagin, L. S. *Mathematical theory of optimal processes*. Routledge, 2018.

Qi, D. and Majda, A. Low-dimensional reduced-order models for statistical response and uncertainty quantification: Two-layer baroclinic turbulence. *Journal of the Atmospheric Sciences*, 73(12):4609–4639, 2016.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *arXiv preprint arXiv:1505.05770*, 2015.

Risken, H. The fokker-planck equation. *Springer Series in Synergetics*, 18:4609–4639, 1989.

Risken, H. and Caugheyz, T. (eds.). *The fokker-planck equation: Methods of solution and application*. Springer, 1991.

Santos, N. P., Lobo, V., and Bernardino, A. Unmanned aerial vehicle tracking using a particle filter based approach. In *IEEE Underwater Technology (UT)*, 2019.

Singh, R., Zhang, Q., and Chen, Y. Learning hidden markov models from aggregate observations. *arXiv preprint arXiv:2011.11236*, 2020.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Wang, Y., Dai, B., Kong, L., Erfani, S. M., Bailey, J., and Zha, H. Learning deep hidden nonlinear dynamics from aggregate data. In *Uncertainty in Artificial Intelligence*, 2018.

Weinan, E., Han, J., and Jentzen, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. In *Communications in Mathematics and Statistics*, pp. 349–380, 2017.

Zahr, M. J. and Persson, P.-O. An adjoint method for a high-order discretization of deforming domain conservation laws for optimization of flow problems. *Journal of Computational Physics*, 326:516–543, 2016.

Zang, Y., Bao, G., Ye, X., and Zhou, H. Weak adversarial networks for high-dimensional partial differential equations. In *arXiv preprint arXiv:1907.08272*, 2019.

Zienkiewicz, O. and Cheung, I. *The Finite Element Method in Engineering Science*. McGraw-Hill European Publishing Programme. McGraw-Hill, 1971. ISBN 9780070941380. URL https://books.google.com/books?id=B99RAAAAMAAJ.