Connections Between DBNs and ODEs

David Merrell

June 4, 2020

1 Introduction

Both Dynamic Bayesian Networks (DBNs) and Ordinary Differential Equations (ODEs) can model the evolution of a dynamical system over time.

In this note we show a relationship between DBNs and ODEs. We use this connection to derive a DBN variant that accounts for irregular intervals between observations. We compute the MLE for its weight parameters and derive a Bayesian score for DBN architectures.

2 Background

We give a little bit of background about DBNs and ODEs. For our purposes we focus on

- Gaussian DBNs with
  - no dependencies *within* timesteps; and
  - linear dependencies between timesteps.
- Linear ODEs.

2.1 DBN essentials

Suppose we have a vector of N variables that we observe at T timepoints:

$$\vec{x}_{t_1}, \vec{x}_{t_2}, \ldots, \vec{x}_{t_T}.$$

A Gaussian DBN posits the following statistical dependencies between the observations:

$$\begin{aligned} \vec{x}_{t_1} &\sim \mathcal{N}(\vec{\mu}_1, \Sigma_1) \\ \vec{x}_{t_i} | \vec{x}_{t_{i-1}} &\sim \mathcal{N}(A\vec{x}_{t_{i-1}}, \Sigma) \qquad \forall i \in \{2, \ldots, T\} \end{aligned}$$

In other words: the values at the first timepoint are drawn from a Gaussian distribution, and we generate the subsequent timesteps by "noisy multiplication" with some matrix, A.

The matrix A defines the dynamics of a system in *discrete time*. The system's state $\vec{x}$ evolves as we repeatedly left-multiply with A. In effect, all observations are separated by uniform time intervals.

We can also think of A as an *adjacency matrix* defining a network on the state variables. There is an edge from x_i to x_j iff $A_{ij} \neq 0$.

2.2 ODE essentials

A (deterministic) linear ODE model uses the following equations to describe the evolution of vector $\vec{x}(t)$ over time:

$$\vec{x}(0) = \vec{\mu}_0$$

$$\vec{x}'(t) = A\vec{x}(t)$$

where A is some matrix. We *solve* the system of ODEs by finding a function $\vec{x}(t)$ that satisfies the equations.

It turns out that we can solve this system in full generality. Here's a hand-wavy derivation:

$$\begin{aligned} & \frac{\vec{x}(t+dt) - \vec{x}(t)}{dt} \simeq A\vec{x}(t) \\ \Rightarrow \quad & \vec{x}(t+dt) \simeq (I + dt \cdot A)\, \vec{x}(t) \\ \Rightarrow \quad & \vec{x}(t + k \cdot dt) \simeq (I + dt \cdot A)^k\, \vec{x}(t) \\ \Rightarrow \quad & \vec{x}(t_f) \simeq (I + dt \cdot A)^{\frac{t_f - t}{dt}}\, \vec{x}(t) \\ \Rightarrow \quad & \vec{x}(t_f) = \exp\left[(t_f - t)A\right] \vec{x}(t) \end{aligned}$$

That is: when we take the limit $dt \to 0$, the matrix power $(I + dt \cdot A)^{\frac{t_f - t}{dt}}$ becomes a matrix exponential $\exp[(t_f - t)A]$.

With that result and the initial condition, we arrive at the solution

$$\begin{aligned}\vec{x}(t) &= \exp[tA]\,\vec{x}(0) \\ &= (\exp A)^t \vec{x}(0).\end{aligned}$$

In summary: the matrix $\exp A$ encodes the evolution of the system as a function of time $t \in \mathbb{R}$.

3 Connecting ODEs to DBNs

We can connect ODEs to DBNs by *adding noise* to ODEs.

Suppose a system is governed by linear ODEs encoded in matrix A. However, this time assume that the system is perturbed by a random process. Under modest assumptions we get the following distribution for $\vec{x}$ after the system evolves from t_0 to t_f:

$$\vec{x}(t_f) \sim \mathcal{N}\Big((\exp A)^{t_f - t_0}\vec{x}(t_0),\ \Sigma(t_f - t_0)\Big) \tag{1}$$

where Σ is a *variance function* with the following properties:

- Σ is increasing.
- $\Sigma(0) = 0$.

For reference, $\Sigma(t) \propto t$ for a Wiener process and $\Sigma(t) \propto t^{2n+1}$ for the n^{th} integral of a Wiener process.

Equation 1 defines a conditional probability distribution $P(\vec{x}(t_f) \mid \vec{x}(t_0))$. It's very similar to the conditional probability used by linear Gaussian DBNs. However, it differs from linear Gaussian DBNs in some important ways:

- its weight parameters $(\exp A)^{t_f - t_o}$ depend on time;
- its noise parameters $\Sigma(t_f - t_0)$ depend on time.

This time-dependence could be useful in many settings. Classical DBNs don't account for non-uniform time intervals between observations, which is unrealistic for modeling e.g., physical systems.

However, there are issues when we try to build a probabilistic model from Equation 1. Specifically, it's unclear whether we can estimate the weight parameter A (or $\exp A$) from data in a computationally efficient way. It's also unclear whether we can formulate a tractable *network structure score* from Equation 1. Recall that we can think of A as a network adjacency matrix; we would like some way to assign marginal probabilities to network structures.

We are faced with a choice between model fidelity and computational tractability, represented by ODEs and DBNs respectively. Fortunately, we can get some of the time-dependence of ODEs while retaining the tractability of DBNs. We do this by using the Taylor expansion of $\exp A$:

$$\begin{aligned}
\vec{x}(t_f) &\sim \mathcal{N}\Big(\exp[(t_f - t_0)A]\vec{x}(t_0),\ \Sigma(t_f - t_0)\Big) \\
&\sim \mathcal{N}\Big([I + (t_f - t_0)A + \ldots]\vec{x}(t_0),\ \Sigma(t_f - t_0)\Big) \\
&\simeq \mathcal{N}\Big([I + (t_f - t_0)A]\vec{x}(t_0),\ \Sigma(t_f - t_0)\Big) \qquad (2)
\end{aligned}$$

Equation 2 is a first-order approximation of Equation 1. It incorporates the time elapsed between t_0 and t_f, but only relies on the first power of A. We'll refer to this sequential model as an "irregular timestep DBN," or itDBN for short.

Some algebra yields a form much closer to classic linear regression:

$$\frac{\vec{x}(t_f) - \vec{x}(t_0)}{t_f - t_0} \sim \mathcal{N}\Big(A\vec{x}(t_0),\ \frac{\Sigma(t_f - t_0)}{t_f - t_0}\Big) \qquad (3)$$

Equation 3 suggests that we can estimate A from irregularly spaced data by treating it as linear regression problem on pairs of $(x, \Delta x/\Delta t)$. The formulation differs slightly from ordinary least squares since the pairs have unequal variances and are therefore given unequal weight in the loss function.

For illustration we compute the Maximum Likelihood Estimate for A. Define the divided difference $\vec{\Delta}_i = (\vec{x}(t_i) - \vec{x}(t_{i-1}))/(t_i - t_{i-1})$ for notational

convenience. We maximize log-likelihood as follows: TODO write details

$$\mathcal{L}(A) \propto \prod_i \exp\Big[- \Big]$$

$$\Rightarrow \qquad l(A) = \sum_i$$

$$\Rightarrow \qquad \nabla l(A) =$$

Solving for $\vec{a}_v$ yields the maximum likelihood estimate:

$$\hat{\vec{a}}_v = (X^T \tilde{\Sigma}^{-1} X)^{-1} X^T \tilde{\Sigma}^{-1} \vec{\Delta}_v. \tag{4}$$

The structure of Equation 4 is intuitively satisfying. It's a variant of the classic normal equations, except that samples are weighted by the lengths of their time intervals—shorter intervals get larger weights.

Other straightforward calculations yield the MLE for σ_v^2:

$$\hat{\sigma}_v^2 = \frac{1}{T-1} (\vec{\Delta}_v - X\vec{a}_v)^T \tilde{\Sigma}^{-1} (\vec{\Delta}_v - X\vec{a}_v) \tag{5}$$

4 A Bayesian score for itDBN architectures

Priors:

$$\vec{a}_v \sim \mathcal{N}\big(0,\ (T-1)\sigma_v^2 (X^T \tilde{\Sigma}^{-1} X)^{-1}\big)$$

$$P(\sigma_v^2) \propto \frac{1}{\sigma_v^2}$$

Marginal likelihood:

$$P(\mathrm{pa}(v) \mid X) = \int_{\sigma_v^2} \int_{\vec{a}_v} P(\Delta_v \mid \mathrm{pa}(v), \vec{a}_v, \sigma_v^2) \cdot P(\vec{a}_v | \sigma_v^2) \cdot P(\sigma_v^2)$$