Deep Bayesian Nonparametric Tracking

Aonan Zhang [1] John Paisley [1]

Abstract

Time-series data often exhibit irregular behavior, making them hard to analyze and explain with a simple dynamic model. For example, information in social networks may show change-point-like bursts that then diffuse with smooth dynamics. Powerful models such as deep neural networks learn smooth functions from data, but are not as well-suited in off-the-shelf form for discovering and explaining sparse, discrete and bursty dynamic patterns. Bayesian models can do this well by encoding the appropriate probabilistic assumptions in the model prior. We propose an integration of Bayesian nonparametric methods within deep neural networks for modeling irregular patterns in time-series data. We use Bayesian nonparametrics to model change-point behavior in time, and a deep neural network to model nonlinear latent space dynamics. We compare with a non-deep linear version of the model also proposed here. Empirical evaluations demonstrates improved performance and interpretable results when tracking stock prices and Twitter trends.

1. Introduction

Irregular behaviors such as bursts and nonlinearity repeatedly show up in time-series data. For example, the past decades have seen several crashes of the stock market, in which a latent continuous-like process is interrupted by a change-point-like event (Adams & MacKay, 2007). Another example is information diffusion in social media, which exhibits initial bursts followed by smoother diffusion (Guille et al., 2013; Leskovec et al., 2009). Traditional tracking models such as the linear Kalman filter (Bishop et al., 2001) are not ideally-suited for these tasks, in part because of their incompatibility with bursts and underlying nonlinearity (Doucet & Johansen, 2009; Andrieu et al., 2010). More complex machine learning techniques such as deep neural networks (LeCun et al., 2015) usually fit smooth nonlinear functions to the data, and require more thought when handling bursty patterns in dynamic data.

On the other hand, Bayesian methods are good at handling discrete and bursty latent structures in data through explicit probabilistic modeling. Typical examples such as learning latent features (Griffiths & Ghahramani, 2011) and latent state transitions (Fox et al., 2011) have proven useful in multiple applications. Moreover, Bayesian nonparametric (BNP) methods are naturally suited for large scale learning problems due to their infinite dimensional nature. Recent inference techniques such as online learning (Hoffman et al., 2013), streaming learning (Broderick et al., 2013), and parallelization (Ge et al., 2015) further scale up these methods. To model complex and nonlinear structures, recent probabilistic generative models mimic the behavior of neural networks (Ranganath et al., 2015; Schein et al., 2016). However, these models require carefully designed inference methods. Current ongoing research largely focuses on generalizing the scope of these models (Ranganath et al., 2016; Tran et al., 2017). Other methods incorporate the power of neural networks in Bayesian models, such as the variational auto-encoder (VAE) (Kingma & Welling, 2014). However, in contrast to traditional Bayesian methods, which are good at learning interpretable patterns through latent variables, VAE's sacrifice interpretability for inference tractability.

In this paper, we aim to integrate the merits of Bayesian models and neural networks when analyzing irregular time-series data. Our strategy is to model streaming data at two different resolutions through a dynamic change-point model. To be more precise, we partition the entire time horizon into mini-batches and model the bursty dynamics between mini-batches using BNP methods. We model nonlinearity within each mini-batch with a deep neural network. In this way we are able to capture bursts and nonlinearity while also achieving interpretable results.

In Section 2 we introduce the modeling idea. In Section 3 we discuss a variational inference method. Section 4 introduces a physics-based extension and derives the predictive distributions. Section 5 demonstrates empirical results on stock and Twitter data.

[1]Department of Electrical Engineering & Data Science Institute, Columbia University, New York, USA. Correspondence to: Aonan Zhang <az2385@columbia.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

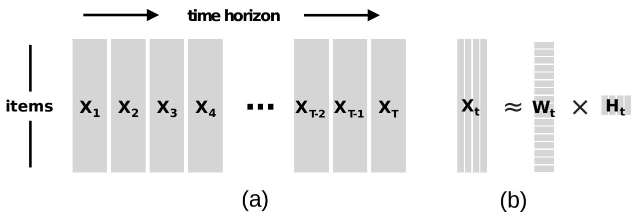

Figure 1. (a) Partitioning sequential data into time blocks. (b) Each block is modeled as a dynamic matrix factorization where the left matrix uses a gamma process for change-point detection and the right matrix is Brownian motion. Both matrices are time-evolving: the left matrix across blocks (using a jump process) and the right matrix along columns.

2. The Model

Basic linear tracking models such as the Kalman filter can be formulated as a matrix factorization problem $X \approx WH$, in which the left matrix W remains fixed and the column sequence of the right matrix H is modeled as dynamically evolving. When W is defined by the underlying physics of the problem, this is natural. But for streaming data such as stock or text data, both matrices are unknown and sequential evolution may be expected in both. In this section, we discuss our approach to the modeling problem $X \approx f(WH)$, in which W is modeled by a jump process that detects change-points in the "global" structure, while H is continuously evolving to model "local" temporal variations. We present two models: the first based on a more traditional BNP approach for which $f(\cdot)$ is the identity function, which we then extend to a neural network in a straightforward way.

2.1. Basic setup: Dynamic matrix factorization

In principle, our dynamic model is a continuous-time model. For practical application, we will group the data into blocks, where X_t is a $M \times N_t$ matrix of data at time block t. Each row could correspond to a stock or a word, and the columns contain the measured time-sequence at N_t points in block t (e.g., hourly measurements in week t). Our basic matrix factorization model is of the form $X_t \approx f(W_t H_t)$, where W_t is evolving in t and the columns of H_t are evolving (see Figure 1). In this sense, an *entire* data matrix X can be viewed as being factorized where H is one process evolving along the columns and H_t only selects the relevant submatrix of this process. The entire matrix W is changing depending on what column subset of X is being modeled, but also evolving according to a dynamic process in t.

In Section 2.2 we discuss the process for W as an infinitely divisible continuous-time jump process, then discuss the discrete time analog that we use for inference. We discuss the process for H in Section 2.3, which results in our proposed non-deep BNP tracking model (for which $f(A) = A$). We make a deep extension in Section 2.4.

2.2. Variance gamma process on W

In the continuous-time setting, we define the matrix W_t to be a Brownian motion subordinated to a gamma process. Let $R = (R_t)$ be a gamma process on state space $\mathbb{R}_+$ with shape rate a and scale c, and $Z = (Z_s)$ be a standard Brownian motion (in matrix form) with state space $\mathbb{R}^{M \times K}$. Then $W_t = Z_{R_t}$ is obtained by subordinating Z to R. Since both Z and R are Lévy processes, W is also a Lévy process, and thus can be represented as summation of independent increments and evaluated at discrete time points through simple marginalization. The gamma process is a pure jump process with occasional large jumps, and therefore the Brownian motion is also a jump process with little motion interrupted by occasional large jumps (Çınlar, 2011). We use this process as a change-point model for W_t, which will allow it to remain nearly fixed over a period of time, with occasional large jumps representing a shock in the dynamic system (e.g., caused by a market event) (Madan et al., 1998).

Mathematically, if we partition X into time blocks t, then Lévy process theory provides a simple generative process for W_t. Let Δs_t be the time lapse between block $t-1$ and t. Then $R_t - R_{t-1} \sim \text{Gam}(a\Delta s_t, c)$. Using a new variable for this difference, the discrete time evolution of W_t can be simply represented as

$$W_t \sim \mathcal{N}(W_{t-1}, \gamma_t I), \quad \gamma_t \sim \text{Gam}(a_0, c), \qquad (1)$$

where $a_0 = a\Delta s_t$, assuming a constant time shift. (We've also overloaded the normal distribution.) When the partition is over a small window of time compared with the dynamics of the process, a_0 will be small and therefore γ_t will likely be small and W_t will have little change. However, γ_t will occasionally be large, which will allow for a change in the system. As written, we assume one gamma process; generalization to row-specific gamma processes is straightforward and will allow for each data stream (e.g., stock) to have its own change-points. The BNP aspect is in inferring these change-points from the data. An example of this row-specific process for W is illustrated in the left column of Figure 2 for a rank-one factorization.

2.3. Temporal tracking in H and data generation

For the local tracking model of H we use discretized Brownian motion. If $H_{t,j}$ is the jth column of H_t, then we model its dynamics and resulting data generation as

$$H_{t,j} \sim \mathcal{N}(H_{t,j-1}, \lambda I), \quad X_t \sim \mathcal{N}(W_t H_t, \sigma^2 I), \qquad (2)$$

where λ represents the time interval between data points (assumed constant), and $H_{t,1}$ uses the last point in H_{t-1}. We've again overloaded notation of the second Gaussian distribution. The columns of H_t follow a continuous process and therefore only allows "smooth" change, although the process is nowhere differentiable.

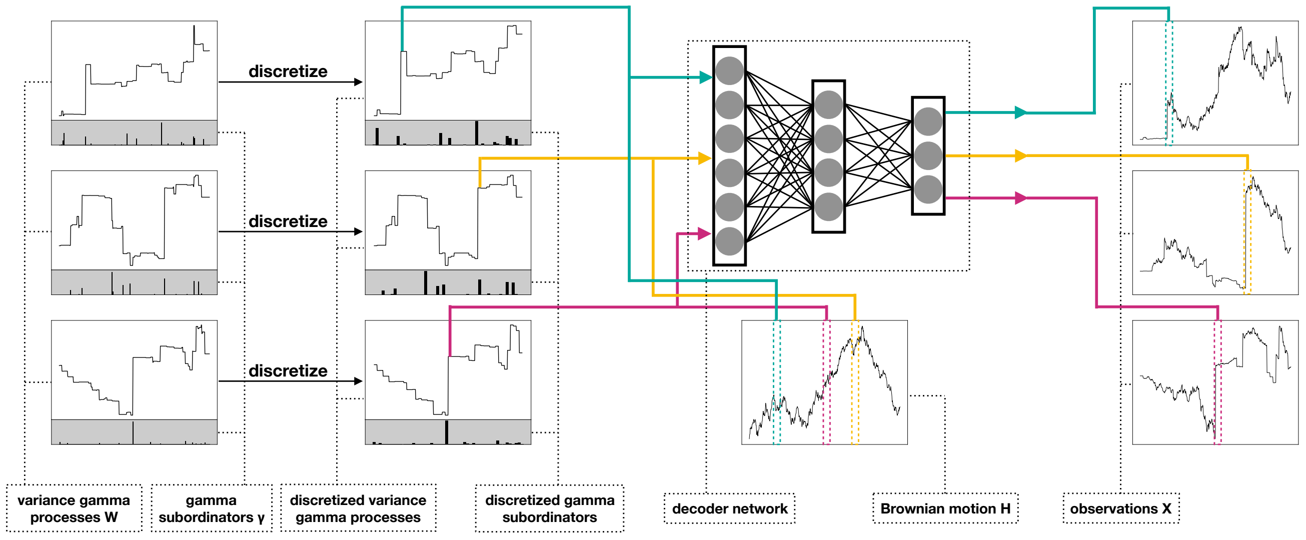

Figure 2. A rank-one example of the deep BNP tracking model for three data streams. A continuous-time variance gamma process is discretized into time blocks at finer resolutions than the large jumps in the process. A single Brownian motion multiplies these variance gamma processes and is passed into a neural network, which parameterizes the mean and covariance of the observed data streams.

2.4. Extension: A deep likelihood model

In the previous sections we proposed a linear Gaussian matrix factorization model for tracking. We next extend this to a deep model in a simple way. Here we use the variational autoencoder framework to motivate the model development, and the inference algorithm discussed later (Kingma & Welling, 2014; Krishnan et al., 2015). We use a neural network, denoted by its parameters ϕ, to define the decoder model $p_\phi(X_t|W_t, H_t)$ for block t as a Gaussian additive noise model,

$$X_t \sim \mathcal{N}\left(\mu_\phi(W_t H_t), \Sigma_\phi(W_t H_t)\right). \tag{3}$$

Here we have again overloaded the Gaussian notation. In this likelihood model, we define a multivariate Gaussian on the jth column of X_t with mean and covariance a neural network that is a function of the jth column of $W_t H_t$. The benefit of this formulation can be seen in the inference step, where we are able to easily handle the stochastic gradient of ϕ (Kingma & Welling, 2014). We discuss the design of networks μ_ϕ and Σ_ϕ in Section 3.2. We illustrate this deep model in Figure 2.

3. Variational inference

3.1. Linear Gaussian observational model

We derive variation inference algorithms for the two models proposed in Section 2. We first discuss inference for the simpler linear Gaussian model. By our choice of q distributions, the algorithm nearly reduces to an EM algorithm in which one component is the traditional Kalman filter. To approximate the full posterior, we use a factorized q distribution of the form

$$q(\gamma, W, H) = \prod_{t=1}^{T} q(\gamma_t)\delta(W_t)q(H_t). \tag{4}$$

We observe that within a time block we do not further factorize $q(H_t)$. Our q on W_t is a delta function, meaning we actually learn a point estimate of this variable. As a result, the conditional posterior $p(\gamma, H|W, X) = p(H|W, X)\prod_t p(\gamma_t|W_t)$. Our only mean-field approximation is therefore in the factorization $\prod_t q(H_t)$, which breaks dependence in the single transition across time blocks. The variational lower bound is,

$$\mathcal{L} = \mathbb{E}_q\Big[\ln \frac{p(\gamma, W, H, X)}{q(\gamma, W, H)}\Big] = \mathcal{L}_H + \mathcal{L}_{W,\gamma}. \tag{5}$$

The H portion is $\mathcal{L}_H = \sum_{t=1}^T \mathcal{L}_H^{(t)}$, where

$$\mathcal{L}_H^{(t)} = \sum_{j=1}^{N_t} \mathbb{E}\Big[\ln \frac{p(X_{t,j}|W_t, H_{t,j})p(H_{t,j}|H_{t,j-1})}{q(H_{t,j}|H_{t,j-1})}\Big]. \tag{6}$$

The W, γ portion is $\mathcal{L}_{W,\gamma} = \sum_{t=1}^T \mathcal{L}_{W,\gamma}^{(t)}$, where

$$\mathcal{L}_{W,\gamma}^{(t)} = \mathbb{E}\Big[\ln \frac{p(\gamma_t)p(W_t|W_{t-1}, \gamma_t)}{q(\gamma_t)}\Big]. \tag{7}$$

We use coordinate ascent to iterate between the following closed-form updates. All expectations are with respect to q.

Update $q(\gamma_t)$: The conditional posterior of γ_t is a generalized inverse Gaussian. Thus $q(\gamma_t) = \text{GIG}(a_t, b_t, p_t)$, where $a_t = 2c, b_t = \|W_t - W_{t-1}\|_F^2, p_t = a_0 - Md/2$.

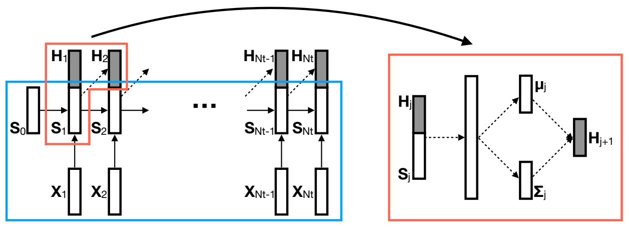

Figure 3. Left: Encoder network using an LSTM. Right: Re-parameterization and sequential sampling from $q_\theta(H_{t,j+1}|X_{t,1:j})$.

Update W_t: This is a closed-form update,

$$W_t = \left(M_1 + \frac{X_t \mathbb{E}[H_t^\top]}{\sigma^2}\right)\left(M_2 + \frac{\mathbb{E}[H_t H_t^\top]}{\sigma^2}\right)^{-1}, \quad (8)$$

$$M_1 = \mathbb{E}[\gamma_t]W_{t-1} + \mathbb{E}[\gamma_{t+1}]W_{t+1},\ M_2 = (\mathbb{E}[\gamma_t] + \mathbb{E}[\gamma_{t+1}])I.$$

Update $q(H_t)$: This is the linear Kalman filtering problem using the current value of W_t. We use the standard forward-backward (filtering-smoothing) algorithm to solve this linear dynamic system (Ghahramani & Hinton, 1996).

3.2. Extension: Variational auto-encoder model

In Section 2.4 we introduced a neural network based likelihood model $p_\phi(X_t|W_t, H_t)$ with nonlinear dependencies on the product $W_t H_t$. Variational inference for this model is non-trivial because of the clear non-conjugacy. We apply the variational auto-encoder by introducing a variational posterior conditioning on the context X,

$$q(\gamma, W, H|X) = \prod_{t=1}^{T} q(\gamma_t)\delta(W_t)q_\theta(H_t|X_t). \quad (9)$$

Note that $q_\theta(H_t|X_t)$ is a neural network encoder model for the decoder $p_\phi(X_t|W_t, H_t)$. To match the posterior structure to the prior one, we further factorize

$$q_\theta(H_t|X_t) = \prod_{j=1}^{N_t} q_\theta(H_{t,j}|H_{t,j-1}, X_t). \quad (10)$$

Encoder design. To exploit the sequential structure of the data, we use LSTM (Hochreiter & Schmidhuber, 1997) to encode X_t. The pipeline of this encoder network is shown in Figure 3. In particular, we introduce another hidden layer S_t as the LSTM output. Then we sample $H_{t,j}$ sequentially by conditioning on $(S_{t,j-1}, H_{t,j-1})$. In this case we concatenate $(S_{t,j-1}, H_{t,j-1})$ into a single vector and pass that vector to two feed-forward neural networks to obtain the Gaussian parameters $\mu_\theta(S_{t,j-1}, H_{t,j-1}), \Sigma_\theta(S_{t,j-1}, H_{t,j-1})$ as outputs[1]. Finally we sample from the distribution given above. The entire process is summarized in Algorithm (1).

[1]In our experiments we restrict Σ_ϕ to be a diagonal matrix and use neural networks to model the logarithm of each diagonal entries. We use this trick for both the encoder network and the decoder network to refrain from doing an additional projection steps, which can be time-consuming.

Algorithm 1 Sampling from $q_\theta(H_t|X_t)$

```
1: Get $S_t$ by passing $X_t$ to an LSTM.
                                        (Figure 3 blue part)
2: Sample $\widehat{H}_{t,1}$ from an initial distribution.
3: for $j = 2, \dots, N_t$ do
4:     Get parameters $(\mu_{t,j}, (\Sigma_{t,j}^{1/2}))$ for $H_{t,j}$ by passing
       $(\widehat{H}_{t,j-1}, S_{t,j-1})$ to a feed-forward network.
                                        (Figure 3 red part)
5:     Sample $\widehat{H}_{t,j} = \mu_{t,j} + \Sigma_{t,j}^{1/2}\varepsilon,\ \varepsilon \sim \mathcal{N}(0, I)$.
                                        (Figure 3 red part)
6: end for
```

Decoder design. As previously discussed, we define

$$p_\phi(X_{t,j}|W, H) = \mathcal{N}(\mu_\phi(W_t H_{t,j}), \Sigma_\phi(W_t H_{t,j})), \quad (11)$$

where parameters $\mu_\phi(W_t H_{t,j}), \Sigma_\phi(W_t H_{t,j})$ are modeled by separate feed-forward neural networks. As we will see, we can exploit this structure to simplify inference using re-parameterization.

Variational inference. Since the variational auto-encoder model is defined locally, we have a similar learning framework. We can again re-write $\mathcal{L} = \mathcal{L}_H + \mathcal{L}_{W,\gamma}$, where $\mathcal{L}_H = \sum_{t=1}^T \mathcal{L}_H^{(t)}$ and $\mathcal{L}_{W,\gamma} = \sum_{t=1}^T \mathcal{L}_{W,\gamma}^{(t)}$. While $\mathcal{L}_{W,\gamma}^{(t)}$ is as in Equation (7), the local bound is slightly different:

$$\mathcal{L}_H^{(t)} = \sum_{j=1}^{N_t} \mathbb{E}\left[\ln \frac{p_\phi(X_{t,j}|W_t, H_{t,j})p(H_{t,j}|H_{t,j-1})}{q_\theta(H_{t,j}|H_{t,j-1}, X_t)}\right]. \quad (12)$$

Update $q(\gamma_t)$: The same as the linear likelihood model.

Update W_t: This time we do not have a closed-form solution since we can only get samples for the hidden variables in the neural network. Instead we will use SGD, where

$$\nabla_{W_t}\mathcal{L} \approx \nabla_{W_t} \frac{1}{M} \sum_{j=1}^{N_t} \sum_{m=1}^{M} \ln p_\phi(X_{t,j}|W_t, \widehat{H}_{t,j}^{(m)}) + \quad (13)$$
$$\nabla_{W_t}\mathbb{E}[\ln p(W_t|W_{t-1}, \gamma_t) + \ln p(W_{t+1}|W_t, \gamma_{t+1})],$$

and $\widehat{H}_{t,j}^{(m)}$ are i.i.d. samples of $H_{t,j}$, $m = 1, \dots, M$. We observe that the first line is approximated by Monte Carlo, while the second is in closed form.

Update $q_\theta(H_t|X_t)$: We again use SGD,

$$\nabla_\theta \mathcal{L} = \sum_{j=1}^{N_t} \nabla_\theta \mathbb{E}\left[\ln \frac{p(H_{t,j}|H_{t,j-1})}{q_\theta(H_{t,j}|H_{t,j-1}, X_t)}\right]. \quad (14)$$

To estimate this gradient, we observe that we are able to move the gradient inside the integral via the log trick (see, e.g., Paisley et al. (2012)). By sampling H_t according to Algorithm 1, we are able to get an unbiased estimation of the gradient. In this case we found that using only one sample $\widehat{H}_t$ is enough to get an estimation of this direction with reasonable variance.

Update $p_\phi(X_t|W_t, H_t)$: We again use SGD to approximate $\nabla_\phi \mathcal{L}$ in a nearly identical way as updating $q_\theta(H_t|X_t)$.

4. Further discussion

We briefly discuss a straightforward motion modeling extension that can better capture latent trajectories, modeling data with missing values, and also the prediction equations we use for our experiments.

4.1. Modeling velocity and acceleration of drift

The Brownian motion H_t discussed above cannot project future trajectories in the latent space. We augment the model with standard tracking methods that imposes "Earth's physics" upon this space. Therefore, we augment H_t by tripling the number of rows and modeling the drift Δs into the future using the kinematic equations as follows,

$$\begin{aligned} H_{t,j}^{(pos)} &= H_{t,j-1}^{(pos)} + \Delta s \cdot H_{t,j-1}^{(vel)} + \frac{\Delta s^2}{2} \cdot H_{t,j-1}^{(acc)}, \\ H_{t,j}^{(vel)} &= H_{t,j-1}^{(vel)} + \Delta s \cdot H_{t,j-1}^{(acc)}, \\ H_{t,j}^{(acc)} &= e^{-\alpha \Delta s} \cdot H_{t,j-1}^{(acc)}. \end{aligned} \quad (15)$$

$\alpha > 0$ is a damping factor. In this way we are able to explicitly model the evolving position, velocity and acceleration of H according to basic physics properties. By introducing two deterministic matrices

$$G_1 = \begin{bmatrix} I & \Delta s \cdot I & \frac{1}{2}\Delta s^2 \cdot I \\ 0 & I & \Delta s \cdot I \\ 0 & 0 & e^{-\alpha \Delta s} \cdot I \end{bmatrix}, \quad G_2 = \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad (16)$$

we can rewrite the transitions and observations as

$$H_{t,j} \sim \mathcal{N}(G_1 H_{t,j-1}, \lambda I), \; X_t \sim \mathcal{N}(W_t G_2 H_t, \sigma^2 I). \quad (17)$$

We see that G_1 projects $H_{t,j-1}$ into the future. (The velocity and acceleration are directly learned from data.) G_2 picks out the current position to generate the observation. The above inference algorithms can be easily modified by inserting G_1 and G_2 at the appropriate places.

4.2. Handling missing data

If there are some missing entries in the data matrix X_t we can introduce an appropriate indicator matrix A_t. The likelihood model with drift then modifies to

$$X_t \sim \mathcal{N}(A_t W_t G_2 H_t, \sigma^2 \text{diag}(A_t) I). \quad (18)$$

Nothing changes with in remaining parts.

4.3. Prediction

There are two scenarios. The first one is "in-matrix" prediction, which aims at predicting missing values in X_t. The second is "sequential prediction," which predicts the next columns of X_t given previous column. For in-matrix prediction, we have information from both the past and the future. Given q, for model with drift, we select the position from H with G_2 and set $\widehat{X}_{t,j} = W_t G_2 \mathbb{E}_q[H_{t,j}]$. The missing values in X_t are filled with the corresponding value in $\widehat{X}_t$. For sequential prediction, for model with drift we project into the future with G_1 and then select the position from H with G_2, and predict $X_{t,j} = W_t G_2 G_1 \mathbb{E}_q[H_{t,j-1}]$.

5. Experiments

5.1. Methods and evaluations

We empirically evaluate several approaches to this matrix factorization tracking problem. We summarize their acronyms in Table 1. One immediate comparison is with a model that uses Brownian motion to model both W and H— in other words the linear model of this paper without the gamma process, but a fixed variance on W_t. We refer to this model as the collaborative Kalman filter (CKF) (Gultekin & Paisley, 2014). Furthermore, we can incorporate the dynamics of Section 4.1 on H_t, and also use the VAE approach discussed above with the CKF.

We also consider three approaches from this paper: the variance gamma process linear model, that model adding dynamics, and a VAE approach to the model with dynamics, where the dynamics are as in Section 4.1. For our experiments, we allow each row of W to have its own associated gamma process, which represents the fact that items such as stocks or words do not share the same change-points. For notational simplicity we presented the algorithm for a single variance gamma process on W, but now we use $q(\gamma_t) = \prod_{m=1}^{M} q(\gamma_{t,m})$ with straightforward modifications.

As a baseline, we compare with piece-wise linear interpolation, which predicts the next value to be the current value, or locally averages two adjacent values. For all methods, we use root-mean-square error (RMSE) as our performance metric based on predictions, as discussed in Section 4.3. For

Table 1. Methods evaluated in our experiments.

Notation	Description
Interp	Piecewise linear interpolation
CKF	Collaborative Kalman filter
CKF-drift	CKF with velocity and acceleration
CKF-VAE	CKF-drift, nonlinear VAE version
VGP	Variance gamma process model
VGP-drift	VGP with velocity and acceleration
VGP-VAE	VGP-drift, nonlinear VAE version

Table 2. Linear likelihood model predictive results for stock data set.

Method	In-matrix RMSE						Sequential RMSE					
	d=5	d=10	d=15	d=20	d=30	d=50	d=5	d=10	d=15	d=20	d=30	d=50
CKF	0.7815	0.7483	0.7198	0.6960	0.6830	0.7124	0.7103	0.5456	0.5295	0.5030	0.4933	0.4874
CKF-drift	0.7618	0.7325	0.7001	0.6672	**0.6620**	0.6853	0.6853	0.5395	0.5291	0.4982	0.4915	**0.4839**
VGP	0.7777	0.7216	0.7038	0.6678	0.6342	0.6241	0.6874	0.5356	0.5205	0.5013	0.4955	0.4845
VGP-drift	0.7416	0.7210	0.6727	0.6463	0.6247	**0.6077**	0.6760	0.5321	0.5191	0.4999	0.4933	**0.4820**
Interp	0.6704						0.4983					

Table 3. Deep likelihood model predictive results for stock data using various LSTM latent unit sizes.

Method	In-matrix	Sequential
CKF-VAE-50	0.6108±0.0245	0.4824±0.0122
CKF-VAE-200	0.5934±0.0205	0.4735±0.0082
VGP-VAE-50	0.5846±0.0132	0.4742±0.0217
VGP-VAE-200	**0.5364±0.0153**	**0.4628±0.0102**

each method we run five experiments with random initializations.

5.2. Stock market crash and recovery, 2008-2012

We present quantitative and qualitative evaluation of our model on stock market data that measures daily stock prices at opening and closing times for $M = 1429$ companies from the AMEX exchange, NASDAQ, and NYSE, giving 2.86 million total measurements from 2008-2012.[2] In particular, we partition the entire time horizon into four-week blocks. This gives $T = 50$ segments in total. Within each block we have $N_t = 40$ measurements for each stock.

For the CKF models, we set the transition variance $\lambda = 0.1$ and the likelihood variance $\sigma^2 = 1$. For models with drift, we set the damping parameter $\alpha = 0.1$. For VGP models, we set $\gamma_{t,m} \sim_{iid} \text{Gam}(1, 1)$, meaning $c = 1$ and the integral of the Lévy measure over four weeks equals 1. For the encoder part of VAE models (see Figure 3), we set the LSTM latent dimension to be 200. We set the output part of the encoder network (see the red box in Figure 3) to be one feed-forward layer followed by a RELU unit (Nair & Hinton, 2010) before splitting the outputs for μ and Σ. The decoder part contains two feed-forward layers with RELU units. For optimization we use Adam (Kingma & Ba, 2014) with learning rate 10^{-4}. In the experiments we find inference it takes around one hour of coordinate ascent for the linear models; the VAE models converge more slowly (approximately one day), but the result is significantly improved.

In order to quantitatively compare those models, we show the in-matrix prediction performance and sequential predictive performance, described in Section 4.3. For in-matrix

[2]Data source: `http://ichart.finance.yahoo.com/`

Table 4. In-matrix prediction results for encoder/decoder choices.

		Decoder		
		ff	recur	conv
Encoder	ff	0.592	0.735	>1
	recur	0.536	0.584	>1
	conv	>1	>1	>1

Table 5. In-matrix prediction results for various choices of decoder.

$f(W_t, H_t)$	$f(W_t \cdot H_t)$	$f(W_t, g(H_t))$	$f(W_t \cdot g(H_t))$
0.673	0.536	>1	>1

prediction we crop $1/5$ of the data for testing and use the rest for training. And for sequential prediction we use stock prices from 2008 to 2012 for training and predict stock prices sequentially in 2013.

Linear likelihood models. We show the predictive performance in Table 2 as a function of the factorization rank. The VGP-based models have superior performance over the models driven only by Brownian motion. We can also see an improvement in RMSE when we explicitly model the latent trajectory in H. This indicates that the latent motion trend in the market is helpful for prediction. We note that the CKF model is even slightly worse than the linear interpolation method, indicating that the stock market is very hard to track with a linear model. For sequential prediction, the performance gap is not as large as the in-matrix prediction since observations ahead of time and correlations among stocks are informative for a precise prediction.

Deep VAE models. Next we show empirical performance for VAEs. We recall that the linear models $W_t H_t$ is fed into the encoder network in our implementation. (We will compare with other approaches later.) The result is shown in Table 3, where we can see that the VAE significantly improves performance of both CKF-drift and VGP-drift models. VGP-VAE is the best among all models by exploiting both latent jumps and nonlinearity to model the data. Moreover, we observe that VAE models get better results when using a larger number of LSTM latent units S.

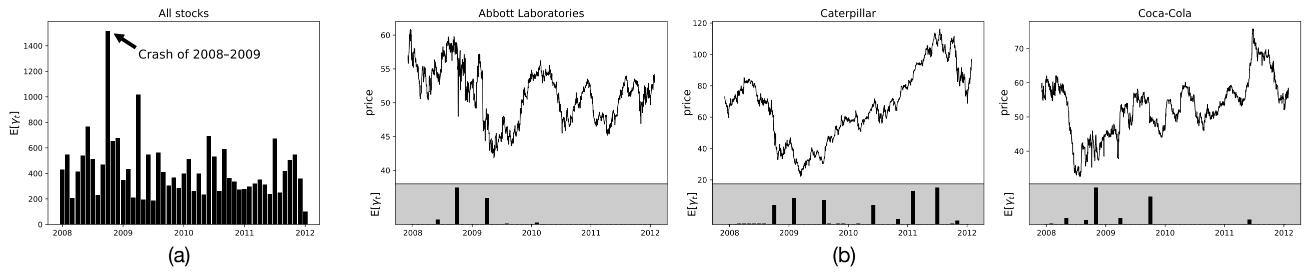

Figure 4. The jumps learned by our deep VGP-VAE model. (a) Summation of posterior means for all stocks $\sum_m \mathbb{E}[\gamma_{t,m}]$. As expected, this shows a significant jump in the market corresponding to the crash of 2008-2009. (b) the stock price for three companies and their corresponding gamma processes. Spikes indicate large jumps of their relevant locations in the row space of W_t (vectors not shown).

VAE network design. We find that the choice of encoder/decoder networks will affect the result. First, we compare three network settings for both the encoder and decoder: feed-forward (ff), recurrent (recur), and convolutional (conv). For recurrent networks we use LSTM for our experiments, and for the decoder convolutional network we use transposed convolution layers (i.e., deconvolution). We tune all networks for various layer widths and depths. Table 4 shows the best result we get for each encoder/decoder network pair. We find that using LSTM as encoder and feed-forward as decoder has the best result. This is because LSTM can exploit the sequential properties within data and effectively encode the observations into H. Since we also assume a sequential structure in $q_\theta(H_t|X_t)$, the decoder can be done in a straightforward way through a feed-forward network.

We also note that our decoder can be generalized as a function of W_t and H_t. In this case, we have multiple choices of the functional form for our decoder network. For example, we can first concatenate W_t and H_t and then feed them into a neural network (denoted as $f(W_t, H_t)$), or we can multiply them before feeding into the network (denoted as $f(W_t \cdot H_t)$, the method we've used thus far). The fact that H_t itself has a sequential structure means we can potentially exploit this by first feeding H into an LSTM (denoted as $g(H_t)$) before considering how it interacts with W_t. Table 5 shows the quantitative results for those choices. We find that feeding the linear model $W_t H_t$ directly into the neural network has the best performance. This indicates our initial BNP linear approach still has value when combined with a deep model.

Qualitative results. We also present a qualitative evaluation of the VGP-VAE model. Figure 4 shows the variance gamma process. In the left plot, we show the posterior mean of the subordinator gamma process $\mathbb{E}[\gamma_{t,m}]$ summed over all stocks, showing amount of "jumpiness" in the stocks learned by our model. We note that this should not be sparse, since in any time frame we expect a subset of stocks to have fundamental changes. However, our model does detect a significant global jump around October in 2008, which corresponds to the most recent stock crash.

In the three right plots, we show the jumps learned from three individual stocks: Abbott Laboratories (Medical & Health), Caterpillar (Energy, Industry), and Coca-Cola (Consumer Staples), in the lower gray subplots. As expected, their learned gamma processes are sparse. From the stock prices, we can see all the three stocks experienced a similar drop during the market crash. The variance gamma process does not detect stable, gradual changes in stock price because these parts can be tracked accurately by H and the VAE. The gamma process is intended to detect changepoints, where fundamental changes occur (i.e., as modeled by jumps in the latent embedding space). For example, in July 2011, Caterpillar agreed to pay a Clean Air Act penalty, which seems to have affected its location relative to other stocks in the row space of W_t.

5.3. NFL tweets

Another typical example where bursts happen naturally is in social media. Here we aim to analyze 377,164 NFL-related tweets (e.g., labeled with an #nfl hashtag) posted during the 2011-2012 season (Sinha et al., 2013). All the data were partitioned into $T = 122$ mini-batch, one per day, and for each day we aggregate all tweets into bag-of-words by hour. Thus $N_t = 24$. We remove all stop words and pick the top $M = 5000$ words by their document level tf-idf rank (Salton & Buckley, 1988). That is, our observation is a $5,000$ by (24×122) time-series matrix.

Quantitative results. We use the first $4/5$ of data for training and the rest for testing. We set $\lambda = 0.3$, $\sigma^2 = 1$, $\alpha = 0.1$, and the prior $\gamma_{t,m} \sim_{iid} \mathrm{Gam}(1, 1)$. For VAE models we set the LSTM latent dimension to be 40 and the learning rate for Adam to be 10^{-4}. The result is shown in Table 6. For CKF models we show the best result when tuning $K \in \{5, 10, 15, 20, 30, 50\}$. Again we see that VAE models have the best predictive performance due to its ability to model bursts through a Bayesian nonparametric jump process and nonlinearity through neural networks.

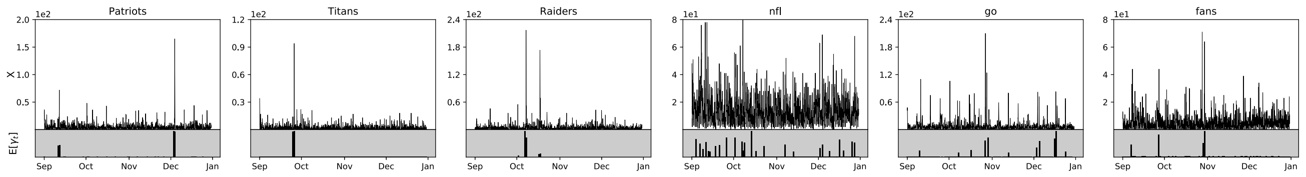

Figure 5. Top plots show raw counts for 6 different words. We can see the word frequency is very noisy in the entire period. Lower shaded plots show posterior mean of the gamma process $\mathbb{E}[\gamma_{t,m}]$ showing the jumps in word embedding locations learned by our model. Note that (i) the resolution of the plots are different, the top being hourly and the bottom per-day; (ii) the learned spikes show jumps of latent row vectors in W_t through a variance gamma process (not shown). So our method is essentially different from sparse recovery algorithms, such as soft-thresholding (Donoho, 1995).

Table 6. Sequential predictive results for NFL tweets.

Method	**Sequential RMSE**
Interp	0.481
CKF	0.478± 0.002
CKF-drift	0.475± 0.002
CKF-VAE	**0.470±0.001**
VGP	0.477± 0.002
VGP-drift	0.471± 0.003
VGP-VAE	**0.463±0.002**

Qualitative results. In Figure 5 we show the counts of various words and their learned gamma process $\mathbb{E}[\gamma_t]$ for the VGP-VAE model. The first three plots shows results for various teams: Patriots, Titans, and Raiders. The appearance of those words is very informative and bursty, and the gamma process captures the big changes that aren't able to be modeled by changes in H. Clearly we can see only a few spikes in the plot for most teams, and often there are two adjacent spikes, which indicates a very short change that doesn't persist. The last three plots show results for four less informative words: 'nfl', 'go', and 'fans'. We still learn some spikes in these words. However, the spikes are less informative and they are at a relatively smaller scale than the spikes learned from first three words.

Sensitivity results. Finally, we show a sensitivity analysis for the VGP model in the predictive performance as a function of block window size[3]. The result is shown in Figure 6, where we can clearly see that when the window size is very small we are unable to get good predictive results. In this case, the H has difficulty accurately capturing the basic dynamics because W_t is updated so frequently and many local optima exist. On the other hand, when we use very large batch size, we are still have decent predictive results. However, in this case our variance gamma process model is very coarse, so that we do not have interpretable results of change-points.

[3]We observe a similar result in the VGP-VAE model.

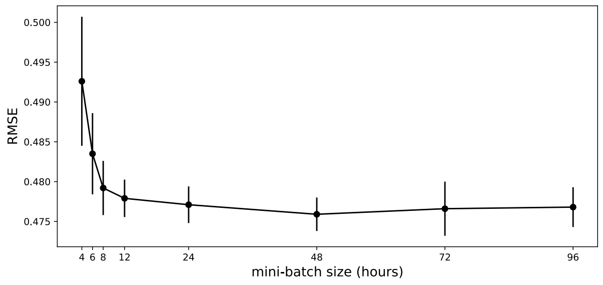

Figure 6. Sensitivity result on nfl tweet data set with various time windows. To get both good predictive and interepretable results, the best choice is to choose an intermediate discretization.

6. Conclusion

In this paper, we introduce a new method to integrate Bayesian nonparametrics and deep neural networks for time-series data. We treat a collection of data sequences, such as stock prices, as a matrix factorization problem in which there are temporal dynamics in both matrices of the factorization. For the left matrix, we use a variance gamma (jump) process to model change-points in what should otherwise be a fixed latent embedding. Columns of the right matrix follow a Brownian motion. We then extend this linear Gaussian basic model to a neural network and use the variational auto-encoder for approximate posterior inference. Our method benefits from both BNP methods (jump processes for change-point detection and explainable latent representations) and deep neural networks (non-linearity for generalized linear modeling). We empirically evaluated our method on two data sets, demonstrating predictive power and interpretability.

References

Adams, Ryan Prescott and MacKay, David JC. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

Andrieu, Christophe, Doucet, Arnaud, and Holenstein, Roman. Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical*

Methodology), 72(3):269–342, 2010.

Bishop, Gary, Welch, Greg, et al. An introduction to the kalman filter. *Proc of SIGGRAPH, Course*, 8(27599-23175):41, 2001.

Broderick, Tamara, Boyd, Nicholas, Wibisono, Andre, Wilson, Ashia C, and Jordan, Michael I. Streaming variational Bayes. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1727–1735, 2013.

Çınlar, Erhan. *Probability and stochastics*, volume 261. Springer Science & Business Media, 2011.

Donoho, David L. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

Doucet, Arnaud and Johansen, Adam M. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

Fox, Emily B, Sudderth, Erik B, Jordan, Michael I, and Willsky, Alan S. A sticky HDP-HMM with application to speaker diarization. *The Annals of Applied Statistics*, pp. 1020–1056, 2011.

Ge, Hong, Chen, Yutian, Wan, Moquan, and Ghahramani, Zoubin. Distributed inference for Dirichlet process mixture models. In *International Conference on Machine Learning (ICML)*, pp. 2276–2284, 2015.

Ghahramani, Zoubin and Hinton, Geoffrey E. Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science, 1996.

Griffiths, Thomas L and Ghahramani, Zoubin. The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12(Apr):1185–1224, 2011.

Guille, Adrien, Hacid, Hakim, Favre, Cecile, and Zighed, Djamel A. Information diffusion in online social networks: A survey. *ACM Sigmod Record*, 42(2):17–28, 2013.

Gultekin, San and Paisley, John. A collaborative Kalman filter for time-evolving dyadic processes. In *International Conference on Data Mining (ICDM)*, pp. 140–149. IEEE, 2014.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Krishnan, Rahul G, Shalit, Uri, and Sontag, David. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *nature*, 521(7553):436, 2015.

Leskovec, Jure, Backstrom, Lars, and Kleinberg, Jon. Meme-tracking and the dynamics of the news cycle. In *International conference on Knowledge discovery and data mining (KDD)*, pp. 497–506. ACM, 2009.

Madan, Dilip B, Carr, Peter P, and Chang, Eric C. The variance gamma process and option pricing. *Review of Finance*, 2(1):79–105, 1998.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, pp. 807–814, 2010.

Paisley, John, Blei, David M., and Jordan, Michael I. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning (ICML)*, 2012.

Ranganath, Rajesh, Tang, Linpeng, Charlin, Laurent, and Blei, David. Deep exponential families. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 762–771, 2015.

Ranganath, Rajesh, Tran, Dustin, and Blei, David. Hierarchical variational models. In *International Conference on Machine Learning*, pp. 324–333, 2016.

Salton, Gerard and Buckley, Christopher. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Schein, Aaron, Wallach, Hanna, and Zhou, Mingyuan. Poisson-gamma dynamical systems. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 5005–5013, 2016.

Sinha, Shiladitya, Dyer, Chris, Gimpel, Kevin, and Smith, Noah A. Predicting the NFL using Twitter. In *The European conference on machine learning & principles and practice of knowledge discovery in databases (ECML/PKDD)*, 2013.

Tran, Dustin, Ranganath, Rajesh, and Blei, David M. Deep and hierarchical implicit models. *arXiv preprint arXiv:1702.08896*, 2017.