Bayesian Non-parametric Model for Time-Series Data

Chawannut Prommin, cp626 CP626@CORNELL.EDU
Serena Li, sl2327 SL2327@CORNELL.EDU
Yutao Han, yh675 YH675@CORNELL.EDU

Abstract

We propose a novel Bayesian non-parametric framework for time-series data modeling with pattern discovery and online inference. We experiment with using the Indian Buffet Process and the infinite Hidden Markov Model for automatic pattern or cluster discovery. Our model then uses a novel framework for online inference of the time-series data using Gaussian Process regression with a Spectral Mixture kernel function and hypothesis testing. We also implement approximate inference for Gaussian Processes to make our model more scalable and allow for online parameter updates.

1. Introduction

Time-series data in several domains is highly volatile and difficult to model (for example, stock prices fluctuate wildly and are difficult to model accurately) due to their often non-stationary nature. Essentially, different locations in input space produce outputs described with variable functions. The distinct locations where functions that map inputs to outputs change are traditionally called change points. While difficult to model, there is significant motivation to learn patterns in time-series data for applications such as creating generative models of the stock market for financial gain or understanding seismic patterns for earthquake detection.

A flexible non-parametric model that identifies underlying patterns within the data can be a powerful tool for understanding time-series. It is intuitive to model the non-stationarity of time-series data by using clustering methods to segment the data into different patterns that correspond to the varying functions (with respect to input space) that model the outputs. Non-parametric models are good candidates to model time-series data because they allow the complexity of the model to grow as more data is observed, which is central to modeling the non-stationarity of the data. Furthermore, non-parametric models do not have the limitations of physics-based models that make assumptions about the data such as optimizing trajectory distance, which can constrain the model and produce unrealistic results. The class of non-parametric models using Kernel learning methods are ideal for modeling time-series data (Rasmussen & Williams, 2006). Kernel learning uses kernel functions to learn the covariance, which determines the difference in outputs given the inputs, of the data in function space which allows for the flexibility of a non-parametric model, while also capturing uncertainty in a probabilistic manner.

Given a model of time-series data, the ability for online inference is desired. As more data is observed online, the model should be able to automatically update its parameters, cluster new data with existing patterns, and identify new patterns. The ability to associate new observed data with previously identified patterns has many potential uses (for example, when analyzing seismic data, matching new data with previously observed patterns in real time can be useful for detecting earthquakes). The scalability of the model is also a key point of concern as online learning needs to be done in real-time for practical applications.

2. Related Work

Previous work has modeled time-series data using purely non-parametric models (Hensman et al., 2014) and a combination of parametric and non-parametric models (Duvenaud et al., 2013). Parametric models generally require less data to learn, but do not have the inherent flexibility of non-parametric models. Parametric models tend to be too heavily constrained to allow the model to increase in complexity given new observed data. Data-driven approaches have modeled time-series data with Gaussian Processes (GP) and Dirichlet Process Gaussian Process (DPGP) (Ross & Dy, 2013). These approaches have had success in clustering multiple trajectories of time-series data, but do not identify clusters of points belonging to a pattern, which represent variable functions mapping inputs to outputs of a single time-series trajectory. Identification of latent patterns or features could potentially allow for higher accuracy during online inference by allowing for interpretation of latent

ORIE 6741 Final Report Fall 2017, Cornell University, Ithaca, NY, 14850

features describing the nature of the non-stationarity in the time-series. Other popular approaches to modeling time-series data find change points with a run-length variable and models the variable functions between change points accordingly (Saatci et al., 2010). A drawback of this type of approach is that the prior over the distribution of change points is not appropriate for modeling real data and a more statistically rigorous approach is desired. Approaches that find change points can be rather manual and we would prefer to automatically discover variations in underlying functions describing the data.

The Indian Buffet Process (IBP) is a non-parametric model that discovers latent features in data and is most commonly used in clustering problems (for example, clustering images based on latent features) (Yildirim, 2012). The IBP allows for each individual data point to be assigned multiple latent features which allows for more flexibility than traditional clustering approaches. There has been application of the IBP to clustering multiple time-series trajectories and learning compositions of GP kernels describing the clusters (Tong & Choi, 2017). However, this approach uses predefined kernels (such as the Radial Basis Function (RBF) or periodic kernels) and the compositions of kernels are limited to summation. A more general kernel function that can learn complex patterns and provide interpretable results is desired. (Wilson & Adams, 2013) developed a powerful Spectral Mixture (SM) kernel for GP regression which can theoretically model any stationary kernel. The SM kernel has shown promising results in learning complex patterns and in performing accurate extrapolation. In addition, the SM kernel allows for direct analysis of the spectral density of the kernel, which facilitates interpretability of the model.

In addition to the IBP, the infinite Hidden Markov Model (iHMM) has been used to cluster time-series data into segments representing variable functions between change points (Gael et al., 2008). The iHMM allows a prior over an infinite number of clusters and automatically discovers the number of clusters. While the iHMM is able discover clusters in time-series data and transition probabilities between the clusters, it does not learn the underlying patterns in the data and is too computationally expensive for online inference due to the need to perform inference over the entire dataset. The ability to discover the parameters of underlying patterns in the clusters and perform online inference is desired.

Modeling with GPs can also cause scalability problems due to the complexity of inverting the kernel matrix; the community has developed approximation methods that exploit kernel matrix structure for fast inference (Wilson & Nickisch, 2015).

3. Methodology

3.1. Contribution

Figure 1 shows the algorithmic flow of our proposed model. Our innovation is a novel three stage non-parametric framework for modeling time-series data with application of the SM kernel to online inference. In stage I, our model automatically discovers the number of underlying patterns and clusters data points. In stage II, GP regression is then performed over each individual discovered cluster. In stage III, the model performs online inference by using a chi-squared goodness of fit test for clustering new observed data into the patterns previously discovered and learned in stages I and II, and updates model parameters online.

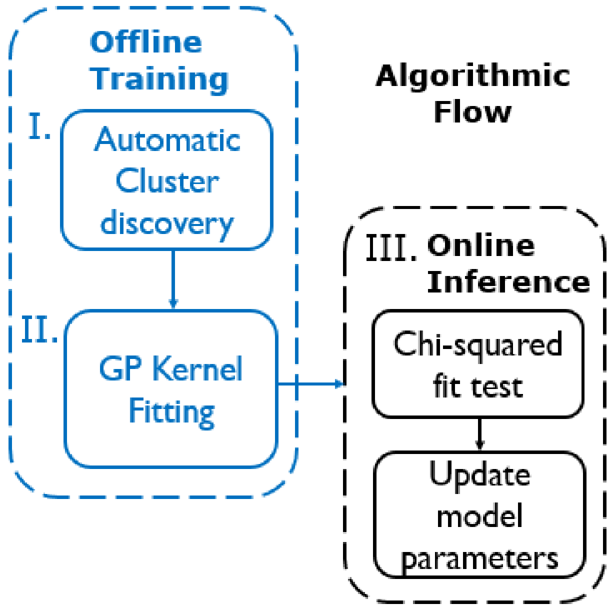

Figure 1. Algorithmic flow of the proposed time-series model. In offline training (light blue) (I) the number of clusters are automatically discovered and (II) the cluster patterns are learned with GP regression with a SM kernel function. (III) In online inference (black) the model performs chi-squared goodness of fit tests to cluster new data and then re-learns kernel hyperparameters for the cluster.

We compare two competing approaches to offline training in this paper. In the first approach, we used the IBP to initially discover the number of clusters. We then used an iterative Gibbs sampling clustering process with GP's to cluster the time-series data. In this approach the GP parameters of each cluster are learned simultaneously as data points are clustered. In the second approach illustrated by figure 1, we used the iHMM to discover the number of clusters and allocate data points to clusters. We then fit a GP to each cluster. In this approach the data points are first clustered with the iHMM and then the GP parameters for the clusters are learned. We compare the two approaches in terms of the number of clusters discovered and the data point classification error.

3.2. Indian Buffet Process (IBP) Discovery of Number of Clusters

Determining the number of clusters in a clustering problem is a difficult question to answer. One approach we tried was using the IBP to discover the number of clusters in a time-series dataset. Finding the hidden structure of time-series data using latent feature modeling is the first step in the IBP. In short, given a data set X with n objects and d dimensions, latent feature modeling aims to decompose X to a binary feature matrix Z with n rows and k columns, a weight matrix A with k rows and d columns, and white noise ϵ.

$$X^{nxd} = Z^{nxk} A^{kxd} + \epsilon^{nxd} \tag{1}$$

In order to do inference, we first need a prior over the binary matrix Z. The IBP is a non-parametric model which gives a prior over binary matrix Z where $z_{ij} = 1$ if object i possess latent feature j and $z_{ij} = 0$ otherwise, where objects $= \{1, ...i, ...n\}$ and features $= \{1, ...j, ...k\}$. (Griffiths & Ghahramani, 2011) derives the closed form of the IBP process. A method to generate samples from the IBP prior is to use a culinary metaphor where N customers enter the restaurant and the first customer chooses Poisson (α_P) dishes, where α_P is a hyperparameter. The i^{th} customer chooses from the previously chosen dishes with probability $\frac{m_K}{i}$, where m_K is the number of prior customers who have chosen that dish. Then, Poisson($\frac{\alpha_P}{i}$) new dishes will be sampled by the i^{th} customer.

Traditional clustering methods assign each data point to one cluster, while the IBP does not have such a constraint. Another drawback of traditional clustering methods is that the number of clusters k needs to be predefined. However, in the IBP k is not fixed and can grow overtime as we observe more data. Figure 2 shows how can we interpret the process of assigning data to clusters in terms of Z. The left of figure 2 represents traditional clustering methods where each object (row) can only belong to one cluster (column) while the right of figure 2 shows the IBP binary feature matrix Z where each object can belong to multiple clusters. For each row of cells, a filled in black cell represents that the object belongs to that cluster. For the time-series data, each data point is an object (row) which can belong to clusters (columns).

The goal of inference in the IBP is to find the posterior of Z given X . The term A is assumed to be from a Gaussian distribution with variance σ_A and ϵ is white noise with variance σ_x. The priors over σ_A and σ_x are gaussians. The posterior $p(Z|X, \sigma_a, \sigma_x)$ is calculated using Bayes rule where the likelihood $p(X|Z, \sigma_a, \sigma_x)$ is Gaussian and the prior $p(Z)$ is drawn from the IBP. In (Griffiths & Ghahramani, 2011), inference is done via Gibbs sampling by calculating the conditional distribution of Z

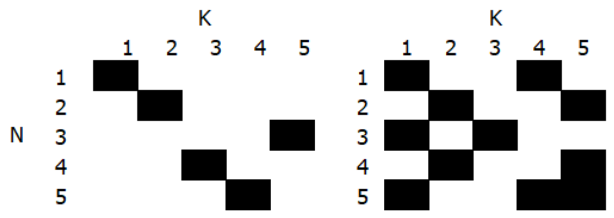

Figure 2. Comparison of traditional clustering methods with the IBP. (Left) Traditional clustering where each data point (row) can only belong to one cluster (column). (Right) In the IBP, each data point can belong to multiple clusters. The black cells represent the clusters that the object belong to. For time-series data, each data point is an object that can belong to clusters.

$$p(z_{ij} = 1|\mathbf{z}_{-i,j}) = \frac{n_{-i,j}}{N} \tag{2}$$

Where $\mathbf{z}_{-i,j}$ is the feature assignments of all the other objects, $n_{-i,j}$ is the total number of other objects that possess latent feature j, and N is the total number of objects. Metropolis-Hastings is then used to find σ_x and σ_a. We use the IBP to discover the number of clusters by using the number of latent features that are discovered. The number of clusters is set to the maxima of the posterior distribution over the number of features.

3.3. Infinite Hidden Markov Model (iHMM) Clustering

Another competing approach we used to discover the clusters is with the iHMM (Gael et al., 2008). A Hidden Markov Model (HMM) assumes latent states which are unobservable and observed states are used to infer the latent states. A HMM is fully parameterized by the transition probabilities t between the latent states, the emission probabilities ε, which are probabilities of the observed states conditioned on the hidden states, and the initial distribution π over the latent states. Figure 3 shows a HMM with observed states h_i and latent states l_i. An iHMM generalizes the number of latent states to infinity (that is, there is an infinite number of latent states that each latent state l_i can transition to) and supports a potentially infinite number of clusters in the time-series data. As opposed to the IBP approach discussed in section 3.2 which is only used to discover the number of clusters, the iHMM approach assigns points to clusters as well discovering the number of clusters.

The number of hidden states (which represents the number of clusters in the time-series data) in the iHMM is inferred using gibbs sampling as derived in (Beal et al., 2001). Unlike the IBP, the iHMM method only allows each data point to belong to one cluster. The three main hyperparameters in iHMM model, α_H, β, and γ are chosen via maximum a posteriori (MAP) estimation. Where α_H is a hyperparameter for the probability for remaining at the same state or cluster, β is a hyperparameter for the prior of π, and γ is a hyperparameter for the total number of states. α_H and β are optimized while γ is chosen based on the data. Each parameter offers an intuitive way of understanding the behavior of the iHMM in exploring new states over time. This method of clustering fits the model to the entire trajectory of the time-series data rather than considering individual data points which allows it to better model the distribution of states or clusters.

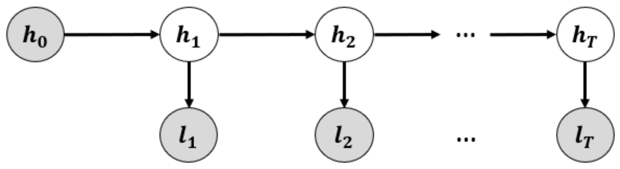

Figure 3. A schematic of the iHMM. The observed states are h_i and the latent states are l_i. The sequence of observed states h_i are used to infer the sequence of latent state transitions of l_i. In an iHMM the number of latent states are generalized to infinity.

3.4. Spectral Mixture (SM) Kernel Learning

A GP learns correlations between the output functions of inputs for given data. The outputs are not assumed to take any functional form and are a normal random vector. The kernel function of the GP describes the covariance between the outputs which is roughly the divergence of the outputs given their inputs. The kernel function has hyperparameters which determine properties of the functions drawn from the GP such as length scale or frequency. We assume white measurement noise. The distribution of output functions is given below (Rasmussen & Williams, 2006)

$$\begin{bmatrix} y(x) \\ y(x_*) \end{bmatrix} = \begin{bmatrix} K + \sigma_n^2 I & K_* \\ K_* & K_{**} \end{bmatrix} \tag{3}$$

Where x is the training input, $y(x)$ is the training output, x_* is the input where we would like to predict the output $y(x_*)$ with GP regression, σ_n is the measurement noise hyperparameter, K is the kernel function and describes the covariance points between the set of points $\{x, x\}$, K_* is the covariance between $\{x_*, x\}$, and K_{**} is the covariance between $\{x_*, x_*\}$. From now on $y(x)$ will be denoted as y, and $y(x_*)$ will be denoted as y^*. With some linear algebra the marginal distribution of y^* is derived as

$$y^* \sim \mathcal{N}(K_*^T K^{-1} y, K_{**} - K_*^T K^{-1} K_*) \tag{4}$$

Generally, kernel functions are hand-crafted to fit specific problems and require significant human time to fine tune. An alternative kernel modeling method that automatically discovers an appropriate kernel is desired. The spectral density can be used to model any stationary kernel function, which is a kernel function that only depends on the translation of the inputs, by taking a mixture of gaussians in the spectral representation (Wilson & Adams, 2013). The inverse Fourier transform of the spectral representation is the SM kernel. The SM kernel function is defined as follows

$$S(s) = \frac{1}{2}[\phi(s) + \phi(-s)] \tag{5}$$

$$K(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}) \tag{6}$$

Where $\phi(s)$ is a mixture of Q Gaussians with the q^{th} component having mean vector $\mu_q = \{\mu_q^{(1)}, ...\mu_q^{(P)}\}$ and diagonal covariance matrix with diagonal components $v_q = \{v_q^{(1)}, ...v_q^{(P)}\}$, $S(s)$ is the spectral density of the kernel function K, and τ_p is the p^{th} component of the P dimensional vector of translation in inputs (Wilson & Adams, 2013). While the time-series data is non-stationary in nature, the underlying patterns are assumed to be stationary and can be appropriately modeled by the SM kernel. As Q goes to infinity, any stationary kernel can be modeled by the SM kernel exactly.

We can learn the hyperparameters $\theta = \{w, \mu, v, \sigma_n\}$ of the kernel by maximizing the marginal likelihood of the data

$$\log p(y|\theta) \propto -y^T M^{-1} y - \log |M| \tag{7}$$

$$M = K + \sigma_n^2 I \tag{8}$$

The gradient of the log likelihood is used for optimization, and is written as

$$\frac{\partial}{\partial \theta} \log p(y|\theta) = \frac{1}{2} y^T M^{-1} \frac{\partial M}{\partial \theta} M^{-1} y - \frac{1}{2} \mathrm{tr}(M^{-1} \frac{\partial M}{\partial \theta}) \tag{9}$$

The hyperparameters of the kernel function essentially determine the inferred distribution over functions at x_*. It is naturally appropriate to continuously update θ as more data is learned for the GP to achieve the best possible fit.

3.5. Clustering Time-Series Data

3.5.1. Clustering with the IBP

In the first approach for offline training described in section 3.1, GP regression is used to cluster the data given the number of clusters discovered k from the IBP process. To cluster the data into k clusters discovered by the IBP, we first randomly initialize the data into k clusters. Then, gibbs sampling is used to find the posterior of the probability of each data point given the hyperparameters θ of each cluster. We define this posterior as

$$p(y_{obs}|D, x_*, \theta_j) \tag{10}$$

Where y_{obs} is the observed output at the new input x_*, $D = \{x_i, y_i\}_{i=1:n}$ is all of the training data, and θ_j are the kernel hyperparameters of the j^{th} cluster discovered by the IBP where $j = \{1 : k\}$. The data point is then assigned to the cluster with the highest posterior. For simple smooth patterns, the popular RBF kernel, which is primarily used for smoothing functions, would produce good results, but for more complex patterns the SM kernel is necessary to discover the pattern parameters.

3.5.2. Clustering with the iHMM

In the second approach for offline training described in section 3.1, the iHMM approach described in section 3.3 automatically discovers the number of clusters while simultaneously assigning data points to clusters. All that is left to do is use GP regression with the SM kernel over the clusters.

3.6. Online Inference

In online inference, our model clusters new data observed online with existing patterns. The model updates clusters kernel hyperparameters as new data is found, and is more scalable because kernel hyperparameters are optimized over clusters rather than the entire dataset.

When new data is observed online, the model first performs chi-squared goodness of fit tests on the observed data and each cluster found during the offline model training. The model subsequently groups the online data into the cluster with the best "fit". The test statistic for the chi-squared test is defined below (Bar-Shalom et al., 2001)

$$t_{\chi^2}^{n_y} = (y^* - y_{obs})^T P^{-1} (y^* - y_{obs}) \tag{11}$$

$$P = K_{**} + \sigma_n^2 I \tag{12}$$

Where y_{obs} is the observed output at the new input x^*, y^* is the expected output at x^* given the SM kernel, K_{**} is the covariance at x^* given the SM kernel, σ_n is the noise hyperparameter, and n_y is the dimension of y_{obs}. The test statistic $t_{\chi^2}^{n_y}$ is chi-squared distributed with n_y degrees of freedom. We assign the new data to the cluster with the highest confidence. In other words, for each cluster we find a level of confidence with hypothesis testing (using GP parameters) that the new data belongs to that cluster and assign the new data to the cluster with the highest level of confidence.

After the new points are assigned to a cluster, the hyperparameters of the SM kernel over the cluster are relearned through marginal likelihood. Rather than re-optimizing hyperparameters over the entire dataset, our model only optimizes over a subset of the data defined by the cluster, which makes the model scalable. However, huge datasets will eventually bottleneck when the clusters grow extremely large so approximation methods must be used (Wilson & Nickisch, 2015). We assign new data to clusters after a predetermined number N_{new} of new data points are observed. If N_{new} is too small, then the chi-squared test may not be representative and it will be computationally expensive to re-learn hyperparameters for each new set of clustered N_{new} points. However, if N_{new} is too large then the resolution may not be small enough to correctly cluster the new points. N_{new} is empirically found in the model.

The SM kernel also allows the model to perform short-term extrapolation of data by assuming the extrapolated points will belong the most recently assigned cluster. Naturally, this method is only accurate for short term extrapolation due to the non-stationary nature of time-series data.

3.7. Fast Inference

The most computationally expensive part of our model is the GP regression. This complexity is due to inversion of the kernel matrix K which has complexity $O(n^3)$ where n is the dimension of the data. As the number of data points increases, GP regression quickly becomes intractable. (Wilson & Nickisch, 2015) proposes to make GP's more scalable by using structured kernel interpolation (SKI) which combines point inducing methods, kronecker fast inference methods, toeplitz methods, along with local interpolation to reduce the complexity of GP regression to $O(n + g(m))$ where m is the number of inducing points and $g(m) \sim m$. The main innovation of KISS-GP is approximating a user defined kernel matrix K as

$$K = W K_{U,U} W^T \tag{13}$$

Where W is a sparse interpolation matrix and $K_{U,U}$ is the covariance of the inducing points U, which are arranged with an exploitable structure. We implement KISS-GP to make GP training faster and online parameter updates feasible. For a more detailed description of KISS-GP, we refer the reader to (Wilson & Nickisch, 2015).

4. Experimental Results

4.1. Code Libraries Used

We used the MATLAB library by Carl Edward Rasmussen written to demonstrate algorithms from (Rasmussen & Williams, 2006) along with code taken from Andrew Gordon Wilson's website (https://people.orie.cornell.edu/andrew/) for implementation of GP regression. For implementation of the iHMM we borrowed MATLAB code from Jurgen Van Gael's (https://github.com/tmills/ihmm) iHMM library. For implementation of KISS-GP we borrowed code from the Python library GPyTorch (https://github.com/jrg365/gpytorch).

4.2. Time-Series Clustering

We tested the algorithm with Well Log data (Lavielle, 2017) which is commonly used for non-stationary time-series analysis and change point detection. For GP regression, the inputs x is the time and the outputs $y(x)$ is the magnetic response.

Figure 4 shows the experimental results of offline training. Figure 4(a) shows the raw Well Log data and Figure 4(b) shows hand labeled clusters for the ground truth. We justify hand labeling the clusters for this dataset due to the obvious distinct patterns in the dataset. Figure 4(c) shows results of clustering the data using the IBP to discover k and GP regression with gibbs sampling for cluster assignment. This process can discover the number of clusters in a probabilistic manner and finds latent features which help with clustering. The result from the experiment correctly clusters most data points into intuitive patterns distinguished by the difference in the magnetic response. However, the data only has an one dimensional input which is not ideal for the IBP which is more suited to high dimensional inputs. It is possible that adding additional input features such as velocity and heading of the time-series trajectory could lead to better performance with the IBP. It is clear there are scattered points incorrectly clustered. A reason why the IBP with GP gibbs sampling method has some incorrectly clustered points is because of its dependence on initialization and issues with autocorrelation during gibbs sampling. The random initialization of data points has a significant effect on the GP functions fit through gibbs sampling, which causes inconsistent results with potentially poor clustering. Figure 4(d) shows the results of the second approach to offline training by clustering using an iHMM. Inference is done using Gibbs sampling for 2000 iterations. First, we randomly assign states to each data point with a maximum number of states k = 10. k is set to be the upper bound of the number of possible states. We also initialize hyperparameters of the iHMM using the empirical mean and variance of data. The number of states for each data point is limited to 1. It is clear that the second approach performs better in clustering the data into intuitive clusters. We believe the iHMM works better because it accounts for the entire trajectory of the time-series data during each iteration rather than each individual point. We also compared our results with standard K-means clustering shown in figure 4(e) which performed poorly, as would be expected for an algorithm that only uses geometry for clustering. For both the IBP and iHMM methods, $k = 3$ clusters are discovered and were shown in red, green, and blue. We also set $k = 3$ for K-means clustering.

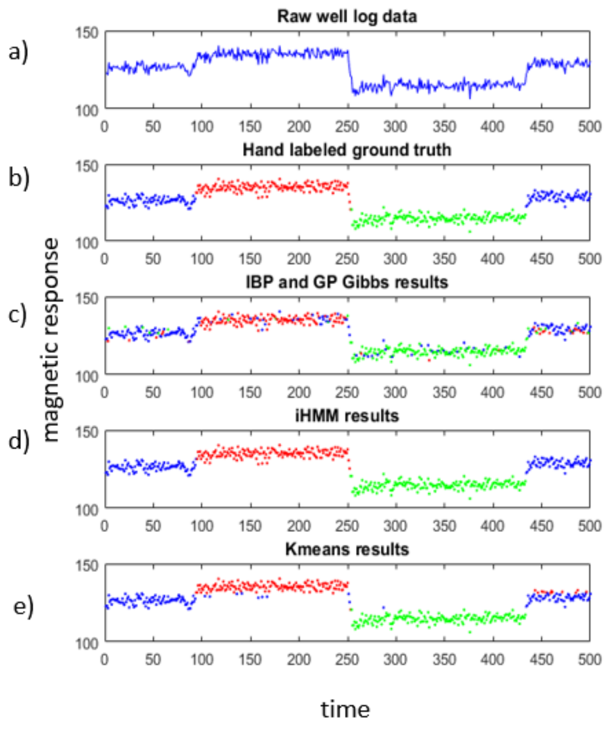

Figure 4. Comparison of clustering methods for Well Log data. **4(a)**: Raw Well Log data. **4(b)**: Hand labeled ground truth. **4(c)**: Clustering result with the IBP and GP gibbs sampling. There are some incorrectly clustered points scattered around the data. **4(d)**: Clustering result with the iHMM. Almost all of the points are clustered correctly. **4(e)**: Clustering result with K-means. Some points are incorrectly clustered due to clustering based only on geometry. All methods had 3 clusters shown in red, green, and blue.

We decided to use the iHMM approach to offline clustering for online inference experiments based on its superior performance in the experiments described in this section.

In addition to the Well Log data we also attempted to use the iHMM model to cluster more complex Bitcoin data from Kaggle (figure 5(a)) (Unknown, 2017). Our results (figure 5(b)) do not shown any intuitive patterns identified.

Table 1. Comparison of offline clustering accuracy

Method	iHMM	K-means	IBP GP
Accuracy (%)	99.4	95.6	80.8

Due to the complexity of the Bitcoin data compared to the Well Log data (which exhibits obvious patterns nearing a synthetic dataset), it is difficult to intuitively evaluate the clustering approaches on the Bitcoin data. As a result, we believe it is more reasonable to use the Well Log data to evaluate the clustering accuracy and online inference.

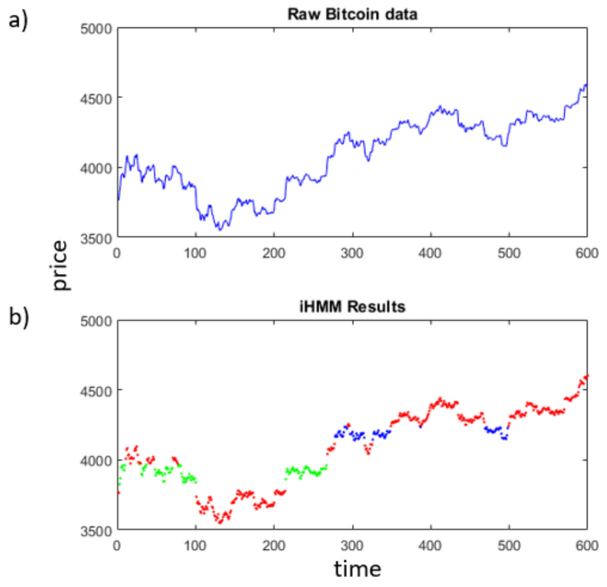

Figure 5. **5(a)**: Raw Bitcoin data. **5(b)**: Clustering results with the iHMM. Three clusters were found and are shown in red, green, and blue.

4.3. Evaluation of Offline Clustering Accuracy

For offline training, we first evaluated the number of clusters discovered by the IBP compared with the iHMM. The results (figure 4) show that both models found the true number of clusters $k = 3$. In addition, we used classification error on the data set to compare the percentage of data correctly assigned to clusters for both methods. The clustering results from the models are compared to the hand labeled ground truth (figure 4(b)). Table 1 summarizes the results for each model. The iHMM performed the best, followed K-means and then the IBP model. The good performance of the iHMM can be attributed to its ability to perform inference over the entire dataset and the poor performance of the IBP GP model can be attributed to how heavily initialization effects the final results.

4.4. Kernel Learning and Chi-squared Testing

In online inference our model first fits a SM kernel over each of the clusters discovered by the iHMM to learn the cluster patterns. We used KISS-GP for fast approximate inference when fitting the SM kernel over the learned clusters. Then, as new data is observed online, the new data is clustered according to the online inference process described in section 3.6. We tested on the full Well Log dataset (figure 6(a)). N_{new} was set to 10. We compared the performance of the popular RBF kernel function with the SM kernel function for GP regression. Each kernel was used to learn the patterns found in the time-series data with the iHMM (figure 6(b)) and then hypothesis testing was performed online to cluster observed data. Kernel hyperparameters are updated online given the newly clustered data. The results for online inference with the RBF kernel are shown in figure 6(c) and the results for online inference with the SM kernel are shown in figure 6(d). For the Well Log data, we found that the RBF and SM kernels performed at a similar level. This is likely due to the simplicity of patterns in the Well Log data clusters, which intuitively appear to be linear patterns with noise. However, we believe that for more complex data, especially periodic time-series data, the SM kernel will show significantly improved performance compared to the RBF kernel.

The parameter N_{new} also has a significant effect on the online inference results as outlined in section 3.6. One issue with using a set value for N_{new} is that points that should belong to different clusters can be assigned to one cluster if the set of points defined by N_{new} includes points from multiple clusters. For future work we would like to implement a sliding window clustering approach that can more accurately cluster new observed data.

5. Discussion

We present a novel non-parametric time-series model that (1) automatically discovers clusters of patterns representing variable functions between change points, (2) learns patterns and correlations using GP regression with a SM kernel, and (3) performs online inference through online clustering with a chi-squared test. Our experiments show promising results in automatic cluster discovery, with the iHMM, and online inference, with hypothesis testing, for simple data. Our model is able to capture the underlying patterns in the time-series data and identify when newly observed data falls into previously discovered patterns, effectively allowing the model to grow as more data is observed. We also implement KISS-GP for scalable offline training and online hyperparameter updating. The ability to accurately classify data online has potential uses in applications such as earthquake detection.

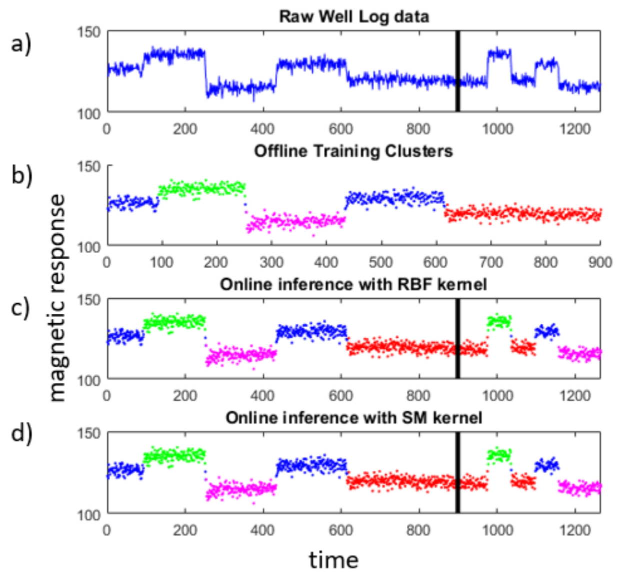

Figure 6. Online inference experimental comparison between RBF and SM kernels. **6(a)**: Full raw Well Log data. The black line indicates the cutoff of the training data. Data points to the left of the black line are used for training and data points to the right are used to test online inference. **6(b)**: Offline clustering with the iHMM. Four clusters were found and shown in red, green, blue, and purple. **6(c)**: Online inference results with the RBF kernel. **6(d)**: Online inference results with the SM kernel.

Potential future research directions include the addition of the ability to detect new clusters online and fit parameters to the newly discovered clusters, and developing an algorithm for change point prediction to complement our model that would increase the range of applications for our model by allowing for long-term extrapolation.

References

Bar-Shalom, Yaakov, Li, X. Rong, and Kirubarajan, Thiagalingam. *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001.

Beal, Matthew J., Ghahramani, Zoubin, and Rasmussen, Carl Edward. The infinite hidden markov model. In *Advances in Neural Information Processing Systems 14*, 2001.

Duvenaud, David, Lloyd, James, Grosse, Roger, Tenenbaum, Joshua, and Zoubin, Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*. JMLR: W&CP, 2013.

Gael, Jurgen Van, Saatci, Yunus, Teh, Yee Whye, and Ghahramani, Zoubin. Beam sampling for the infinite hidden markov model. In *Proceedings of the 25th International Conference on Machine Learning*. JMLR, 2008.

Griffiths, Thomas L.. and Ghahramani, Zoubin. The indian buffet process: An introduction and review. *JMLR*, 2011.

Hensman, James, Rattray, Magnus, and Lawrence, Neil D. Fast nonparametric clustering of structured time-series. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:383–393, 2014.

Lavielle, Marc. Statistics in action with r. Online, March 2017.

Rasmussen, Carl Edward and Williams, Christopher K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Ross, James C. and Dy, Jennifer G. Nonparametric mixture of gaussian process with constraints. In *Proceedings of the 30th International Conference on Machine Learning*. JMLR: W&CP, 2013.

Saatci, Yunus, Turner, Ryan, and Rasmussen, Carl Edward. Gaussian process change point models. In *Proceedings of the 27th International Conference on Machine Learning*. JMLR, 2010.

Tong, Anh and Choi, Jaesik. Discovering explainable latent covariance structure for multiple time series. 2017.

Unknown. Bitcoin historical data. Kaggle, 2017.

Wilson, Andrew Gordon and Adams, Ryan Prescott. Gaussian process kernels for pattern discovery and extrapolation. *JMLR*, 2013.

Wilson, Andrew Gordon and Nickisch, Hannes. Kernel interpolation for scalable structured gaussian process (kiss-gp). In *Proceedings of 32nd International Conference on Machine Learning*. JMLR: W&CP, 2015.

Yildirim, Ilker. Bayesian statistics: Indian buffet process. 2012.