Predictive Model for Time-Series Data with Bayesian Non-parametrics

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

Abstract

We propose a novel model for time-series data using a Bayesian non-parametric framework. Training data is clustered into data patterns using the Indian Buffet Process, which is a flexible non-parametric model that allows each individual data point to be potentially assigned to multiple clusters. We also propose a framework to improve accuracy of online inference and extrapolation of the time-series data. We consider the scalability of the model during online inference due to evaluation of the clusters rather than the entire dataset, and scalability of the model during training by using approximate methods to learn the parameters of the non-parametric model.

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 the functions that map inputs to outputs change are traditionally called change points. While difficult to model, there is significant motivation to learn time-series data for applications such as creating generative models of the stock market for financial gain or understanding seismic patterns to prevent catastrophe.

A predictive model that identifies latent features within the data can be a powerful tool for understanding the time-series. It is intuitive to model the non stationarity of time-series data by using clustering methods to segment the time-series 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 of 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 called Kernel Learning is ideal for modeling time-series data. Kernel learning uses kernel functions to learn the covariance, which roughly gives the expected difference in outputs given the inputs, of the data in function space 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 update its parameters, cluster new data with existing patterns, and identify new patterns. The scalability of the model is a key point of concern as online learning needs to be done in real-time for maximum benefit. In addition to increasing the complexity of the model with online data, extrapolation of future data is desired. A Bayesian approach to online inference and extrapolation is appropriate in order to take advantage of the inductive biases in the model.

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 may be too heavily constrained to allow increases in model 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 a variable functions mapping inputs to outputs of a single time-series trajectory. Identification of latent patterns or features would potentially allow for higher accuracy during online inference by allowing for interpretation of latent features describing the nature of the non-stationarity in the time-

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

series. Other popular approaches to modeling time-series data find change points, which correspond to boundaries between variable functions, 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 can discover latent features in data and is most commonly used in clustering problems (for example, clustering images based on latent features). 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 learn compositions of GP kernels describing the clusters (Tong & Choi, 2017). However, this approach uses predefined kernels (such as the RBF or periodic kernels) and the compositions of kernels are limited to summation. A general kernel function that can learn complex patterns and provide interpretable results is desired. [Ref] developed a powerful Spectral Mixture (SM) kernel for GP regression which can theoretically can 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 kernel.

The non-parametric nature of the IBP means it can quickly become intractable given the complexity of the data. [Ref] performs fast inference with the IBP by using a mean variational method. 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 model. We propose an IBP clustering process to cluster time-series data. Our model then uses GP regression with a SM kernel to learn the correlations on data for each cluster. Finally, our method performs online inference by using a chi-squared goodness of fit test for clustering new data and updating model parameters.

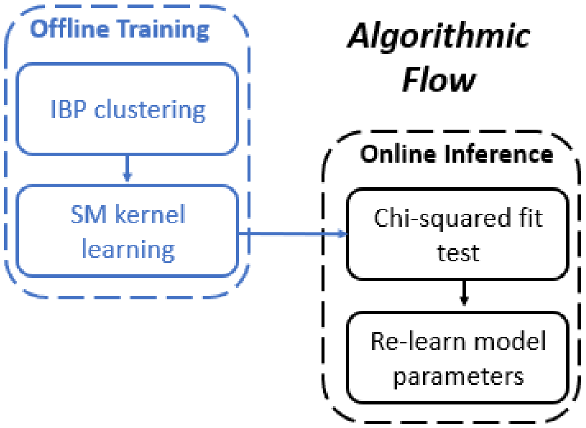

Figure 1. Algorithmic flow of the proposed time-series model. In offline training (light blue) the data is clustered into patterns with the IBP process and then GP regression is preformed over each cluster with the SM kernel. In online inference (black) the model performs chi-squared goodness of fit tests to cluster new data and then re-learns SM kernel hyperparameters for the cluster.

3.2. Indian Buffet Process (IBP) Clustering

Finding the hidden structure of time-series data using latent feature modeling is the first step in our model. 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. One interpretation of each row of matrix Z is whether each object possesses latent feature k or not.

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

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 $= \{1, ...j, ...k\}$. [Ref] derives the closed form of the IBP process [eq]?. There are two equivalent ways to generate samples from the IBP process.

Traditional clustering methods assign one data point to one cluster while 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 have more data is observed. 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.

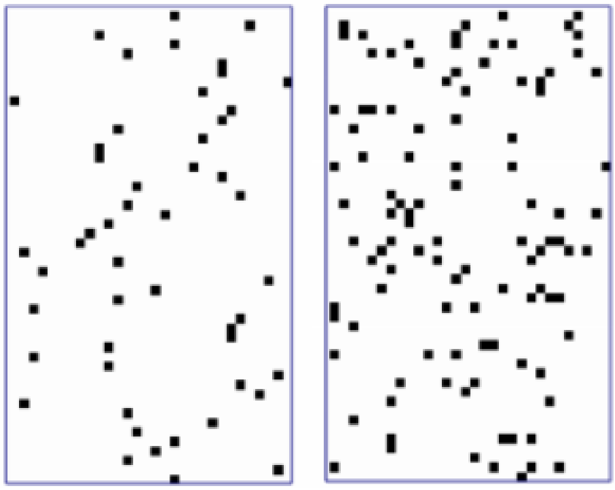

Figure 2. Comparison of traditional clustering methods with the IBP. (Left) Traditional clustering where each data point (row) can only belong to one cluster (column). (Right) In the IBP, each data point can belong to multiple clusters.

The goal of inference in the IBP is to find the posterior of Z given X . The term A is assumed to be from Gaussian distribution with variance σ_A and ϵ is white noise with variance σ_n. 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 [Ref], inference is done via Gibbs sampling by calculating the conditional distribution of Z.

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

Where $\mathbf{z}_{-i,j}$ is the feature assignments of all the other objects, $n_{-i,j}$ is the total number of other object 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 which is computationally expensive because....

[Ref] provides a fast implementation of IBP by using a mean variational method. Parameter restarts are also introduced to avoid bad initialization. Based on our empirical studies, this implementation is significantly faster than the [ref]. (Add more about mean variational method)

3.3. Spectral Mixture (SM) Kernel Learning

GP regression is used to learn each cluster from the IBP process. 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 roughly is 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.

$$\begin{bmatrix} y(x) \\ y(x_*) \end{bmatrix} = \begin{bmatrix} K + \sigma_n^2 I & K_* \\ K_* & K_{**} \end{bmatrix} \tag{1}$$

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^2 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:

$$y^* \sim \mathcal{N}(K_*^T K^{-1} y, K_{**} - K_*^T K^{-1} K_*) \tag{2}$$

A stationary kernel function is one that only depends on the norm between the inputs. 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 function is desired. [ref] The spectral density can be used to model the stationary kernel and by taking a mixture of gaussians in the spectral representation, any stationary kernel can be modeled. The inverse Fourier transform of the spectral representation is the SM kernel. The SM kernel function is defined as follows:

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

$$K(\tau) = \sum_{q=1}^{Q} w_q \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\} \cos(2\pi \tau_p \mu_q^{(p)}) \tag{5}$$

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 difference in inputs [ref]. While the the time-series data is non-stationary in nature, the clusters found by the IBP are 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.

This is not feasible in practice, so a value of Q that learns the data patterns reasonably well is empirically determined.

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

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

$$M = K + \sigma_n^2 I \quad (1)$$

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

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

$$-\frac{1}{2} \mathrm{tr}(M^{-1} \frac{\partial M}{\partial \theta}) \quad (4)$$

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.

3.4. Online Inference

In online inference, our model clusters new data observed online with existing patterns or generates new clusters for the new data. The model updates clusters kernel hyperparameters as new data is found, and is more scalable because 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 learned during the model training. The model subsequently groups the online data into the cluster with the best ”fit”. If the data does not fit any existing clusters with a predefined level of confidence then a new cluster is generated. The test statistic for the chi-squared test is defined below:

$$t_{\chi^2}^{n_y} = (y(x_*) - y)^T P^{-1} (y(x_*) - y) \quad (1)$$

$$P = K_{**} + \sigma_n^2 I \quad (2)$$

Where y is the observed output at the new input x_* online, $y(x_*)$ is the expected output at x_* given the SM kernel, K_{**} is the covariance at x_* given the SM kernel, σ_n^2 is the noise hyperparameter of the SM kernel, and n_y is the dimension of y. The test statistic $t_{\chi^2}^{n_y}$ is chi-squared distributed with n_y degrees of freedom. If we are more than 90% confident that y was generated by the learned distribution over a cluster from the training data with the SM kernel, then the new data is assigned to that cluster. If the new data is more than 90% confident to belong to multiple clusters, then we assign the new data to the cluster with the highest confidence. If the new data cannot be assigned to any cluster with 90% confidence then it is assigned to a new cluster.

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. We use point inducing methods for fast inference of the GP. 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.

4. Experimental Results

4.1. IBP Clustering

We tested the algorithm with a motorcycle crash data set commonly used for non-stationary data [Ref]. Figure 4 shows the result of IBP inference with latent features for each data point that determine clusters of the data. The IBP can discover the number of clusters in a probabilistic manner and discovers latent features which could help with clustering. There are two issues at the current stage of the experiments. First, the data only has one dimensional inputs which is not ideal for the IBP (why?). Second, the posterior of Z is the mean of the entire Monte Carlo Markov Chain (MCMC) which still requires heuristics to assign data points to clusters. Figure 4 shows results from the clustering with the IBP and clustering with fuzzy C-means [ref]. For future experiments, we plan to use the IBP as a prior for another clustering algorithm.

4.2. Kernel Learning and Chi-squared Testing

Our model fits a SM Kernel over each IBP cluster to learn the cluster patterns. Then when new data is observed online, the new data will be clustered according to the online inference described in section 3.4.

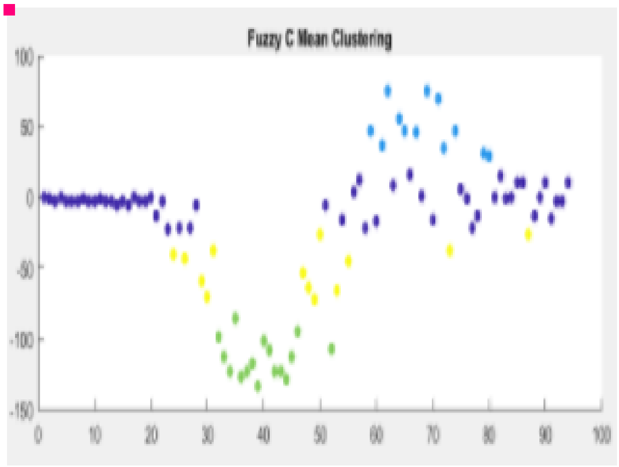

Figure 3. Comparison of traditional clustering methods with the IBP. (Left) Traditional clustering where each data point (row) can only belong to one cluster (column). (Right) In the IBP, each data point can belong to multiple clusters.

4.3. Evaluation of Model Accuracy

To measure the performance of model prediction, we plan to use root mean squared error (RMSE) on the test data to compare the proposed model to other modeling methods. We are still developing the online inference part of the model described in section 3.4. Once the online inference part of the algorithm has been developed, we can calculate the RMSE between data observed online and GP extrapolation of assigned clusters. The benchmark for the RMSE of this algorithm will be the RMSE from extrapolation of the test data without any IBP clustering. We expect our model will work well for non-stationary data given the clusters from the IBP are good.

5. Discussion

We present a novel non-parametric time-series model that (1) clusters data into underlying patterns with the IBP, (2) learns patterns and correlations of clusters using GP regression with a SM kernel, and (3) performs online inference and extrapolation through online clustering with a chi-squared test. Our experimental results so far are promising in IBP clustering compared other clustering methods with automatic discovery of cluster number and finding intuitive clusters. For future experiments we plan to validate the online inference section of the algorithm and improve the scalability of the GP regression process.

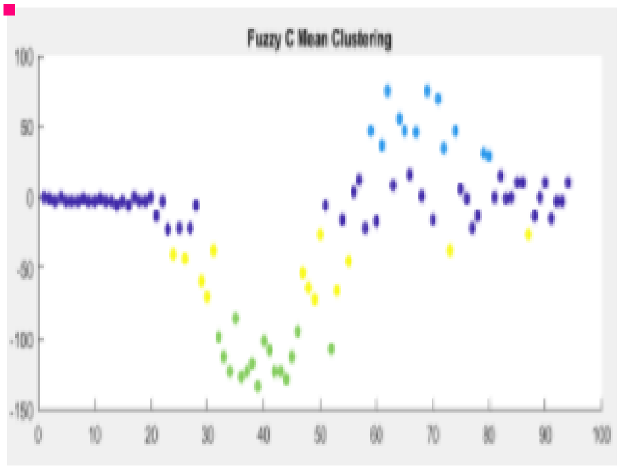

Figure 4. THIS FIGURE FOR SPACING. Comparison of traditional clustering methods with the IBP. (Left) Traditional clustering where each data point (row) can only belong to one cluster (column). (Right) In the IBP, each data point can belong to multiple clusters.

References

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.

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.

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.

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.