Gaussian Process Change Point Models

Yunus Saatçi YS267@CAM.AC.UK
Ryan Turner RT324@CAM.AC.UK
University of Cambridge, Cambridge, CB2 1PZ United Kingdom

Carl Edward Rasmussen CER54@CAM.AC.UK
University of Cambridge, UK & Max Planck Institute for Biological Cybernetics, Tübingen, Germany

Abstract

We combine Bayesian online change point detection with Gaussian processes to create a nonparametric time series model which can handle change points. The model can be used to locate change points in an online manner; and, unlike other Bayesian online change point detection algorithms, is applicable when temporal correlations in a regime are expected. We show three variations on how to apply Gaussian processes in the change point context, each with their own advantages. We present methods to reduce the computational burden of these models and demonstrate it on several real world data sets.

1. Introduction

Nonstationarity, or changes in generative parameters, is often a key aspect of real world time series. An inability to react to regime changes can have a detrimental impact on predictive performance. Change point detection (CPD) attempts to reduce this impact by recognizing regime change events and adapting the predictive model appropriately. As a result, it can be a useful tool in a diverse set of application domains including robotics, process control, and finance. CPD is especially relevant to finance where risk resulting from parameter changes is often neglected in models. For example, Gaussian copula models used in pricing collateralized debt obligations (CDOs) had two key flaws: assuming that subprime mortgage defaults have a fixed correlation structure, and using a point estimate of these correlation parameters learned from historical data prior to the burst of the real-estate bubble (Li, 2000; Jones, 2009). Bayesian change point analysis avoids both of these problems by assuming a change point model of the parameters and integrating out the uncertainty in the parameters rather than using a point estimate.

Many of the previous Bayesian approaches to CPD have been retrospective, where the central aim is to infer change point locations in batch mode (Barry & Hartigan, 1993; Xuan & Murphy, 2007). While these methods are useful for analyzing a variety of time series datasets, they are not designed for online prediction systems that need to adapt predictions in light of incoming regime changes. Examples of such systems can include dialog systems, satellite security systems, and adaptive compression algorithms, to name a few.

The Bayesian Online CPD (BOCPD) algorithm was recently introduced by Adams & MacKay (2007), and similar work has been done by Fearnhead & Liu (2007). Central to the online predictor is the time since the last change point, namely the *run length*. One can perform exact online inference about the run length at every time step, given an *underlying predictive model* (UPM) and a *hazard function*. Given all the observations up to time t, $x_1 \dots x_t \in \mathcal{X}$, the UPM is used to compute $p(x_t|x_{(t-\tau):(t-1)}, \theta_m)$ for any $\tau \in [1, \dots, (t-1)]$. The UPM can be thought of as a simpler base model whose parameters change at every change point; for instance, the UPM could be iid Gaussian with a different mean and variance within each regime. The hazard function $H(r|\theta_h)$ describes how likely we believe a change point is given an observed run length r. Notice that through $H(r|\theta_h)$ we can specify, *a priori*, arbitrary duration distributions for parameter regimes. The only UPM considered in standard BOCPD is constructed using the assumption that the data in each segment is iid with respect to some (ideally exponential family) distribution. However, many datasets are not well

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

described by a generative model which is piecewise iid. Indeed, many temporal sequences clearly exhibit pronounced temporal smoothness within each regime. The standard BOCPD algorithm also treats its hyper-parameters, $\theta := \{\theta_h, \theta_m\}$, as fixed and known. It is clear empirically that the performance of the algorithm is highly sensitive to hyper-parameter settings.

In this paper, we first extend BOCPD by implementing UPMs which exploit temporal structure within each regime, using two time series models based on Gaussian processes (GPs). GPs can be used to model time series data directly (i.e. the mapping $\mathcal{T} \rightarrow \mathcal{X}$, where $\mathcal{T}$ is the set of time indices). This gives rise to a GP time series (GPTS) model. Alternatively, they can be used to learn the mapping between observations x_t and x_{t+1} to produce a nonlinear autoregressive (AR) model (an ARGP). GPs are an attractive choice for time series UPMs because one can integrate out the functions representing the GPTS and ARGP mappings, thus improving predictive performance. We also improve the flexibility of BOCPD by providing a principled mechanism to learn hyper-parameters from the incoming data stream without sacrificing the online nature of the algorithm. Finally, since the range of the run length variable grows linearly with time, we introduce a filtering technique to limit it to size K, where K is set beforehand to trade-off accuracy versus speed.

A similar sequential nonstationary GPTS model was introduced in Garnett et al. (2009). In this approach GPs are made robust to incoming change points through the introduction of an additional hyper-parameter in the covariance function, which represents the location at which a change point occurs inside a past window of data. The size of the window is prespecified and it is implicitly assumed that there can only be one change point inside it. Next-step predictions are improved using hyper-parameter marginalization (this includes the change point hyper-parameter). As the required integral is intractable for a GP model, Bayesian Monte Carlo (BMC) (Rasmussen & Ghahramani, 2002) is used to perform *quadrature* integration. BMC requires defining a separate GP to model the marginal likelihood surface in addition to the GP in the model itself. This is problematic because the marginal likelihood function is *positive* by definition. Hence the approximation by a GP, whose range is the entire real line, can be poor. This problem gets worse as the dimensionality of the hyper-parameter space increases, which occurs when we apply change point detection to higher-dimensional time series data.

The paper is organized as follows: In Section 2, we derive the message passing scheme used for inference in BOCPD. In Section 3, we review GPs and explain GPTS and ARGP. Learning via hyper-parameter optimization is explained in Section 4. Finally, in Section 5 we illustrate our methods on a diverse set of data sets applicable to change point analysis.

2. The BOCPD Algorithm

BOCPD calculates the posterior run length at time t, i.e. $p(r_t|x_{1:t})$, sequentially. This posterior can be used to make online predictions robust to underlying regime changes, through marginalization of the run length variable:

$$\begin{aligned} p(x_{t+1}|x_{1:t}) &= \sum_{r_t} p(x_{t+1}|x_{1:t}, r_t) p(r_t|x_{1:t}) \\ &= \sum_{r_t} p(x_{t+1}|x_t^{(r)}) p(r_t|x_{1:t}) \,, \end{aligned} \tag{1}$$

where $x_t^{(r)}$ refers to the last r_t observations of x, and $p(x_{t+1}|x_t^{(r)})$ is computed using the UPM. The run length posterior can be found by normalizing the joint likelihood: $p(r_t|x_{1:t}) = \frac{p(r_t, x_{1:t})}{\sum_{r_t} p(r_t, x_{1:t})}$. The joint likelihood can be updated online using a recursive message passing scheme

$$\begin{aligned} \gamma_t := p(r_t, x_{1:t}) &= \sum_{r_{t-1}} p(r_t, r_{t-1}, x_{1:t}) \\ &= \sum_{r_{t-1}} p(r_t, x_t|r_{t-1}, x_{1:t-1}) p(r_{t-1}, x_{1:t-1}) \\ &= \sum_{r_{t-1}} \underbrace{p(r_t|r_{t-1})}_{\text{hazard}} \underbrace{p(x_t|r_{t-1}, x_t^{(r)})}_{\text{UPM}} \underbrace{p(r_{t-1}, x_{1:t-1})}_{\gamma_{t-1}} \,. \end{aligned} \tag{2}$$

This defines a forward message passing scheme to recursively calculate γ_t from γ_{t-1}. The conditional can be restated in terms of messages as $p(r_t|x_{1:t}) \propto \gamma_t$. All the distributions mentioned so far are implicitly conditioned on the set of hyper-parameters θ.

Example BOCPD model. A simple example of BOCPD would be to use a constant hazard function $H(r|\theta_h) := \theta_h$, meaning $p(r_t = 0|r_{t-1}, \theta_h)$ is independent of r_{t-1} and is constant, giving rise, *a priori*, to geometric inter-arrival times for change points. The UPM can be set to the predictive distribution obtained when placing a Normal-Inverse-Gamma prior on iid Gaussian observations (i.e., a Student-t predictive):

$$\begin{aligned} &x_t \sim \mathcal{N}(\mu, \sigma^2), \\ &\mu \sim \mathcal{N}(\mu_0, \sigma^2/\kappa), \;\; \sigma^{-2} \sim \text{Gamma}(\alpha, \beta) \,. \end{aligned} \tag{3}$$

In this example $\theta_m := \{\mu_0, \kappa, \alpha, \beta\}$.

2.1. Improving Efficiency by Pruning

The posterior $p(r_t|x_{1:t})$ forms a vector of length t, which is problematic for long time series since it requires $t+1$ updates to propagate it to the next time step. In the parametric exponential family case the updates and predictions in the UPM run in constant time regardless of the run length. Thus total run time of a naive implementation is $\mathcal{O}(T^2)$. In practice, the run length distribution will be highly peaked. We can make the approach more efficient by *pruning* out run lengths with probability below, e.g., $\epsilon = 0.001$ or considering only the K most probable run lengths and setting the remaining probabilities to zero. These modifications run in $\mathcal{O}(T/\epsilon)$ and $\mathcal{O}(TK)$, respectively.

3. GP-based UPMs

A GP is a distribution over functions and is specified by a mean function $\mu(\cdot)$ and a covariance function $k_\xi(\cdot,\cdot)$, also called a *kernel*, (Rasmussen & Williams, 2006). The covariance function is parameterized by a set of hyper-parameters ξ which describe general properties of the functions generated from the prior, such as smoothness, length scales and amplitude. The mean function is typically set to zero. The most common covariance function is the squared-exponential, which generates smooth functions that are infinitely differentiable. It has $D+1$ hyper-parameters, where D is the dimensionality of the input space.

In GPTS the time index t is treated as the input while the time series observation x_t is the output:

$$x_t = f(t) + \epsilon_t,\ f \sim \mathcal{GP}(0, k_\xi),\ \epsilon_t \sim \mathcal{N}(0, \sigma_n^2)\,. \quad (4)$$

It can be shown that GPTS generalizes many of the classic time series models such as AR, the autoregressive moving average (ARMA) (Murray-Smith & Girard, 2001), and the Kalman filter. The GPTS UPM gives rise to a Gaussian predictive distribution:

$$p(x_t|x_{(t-\tau):(t-1)}, \theta_m) = \mathcal{N}(m_t, v_t)\,, \quad (5)$$

where

$$m_t = \mathbf{k}_\star^\top (K + \sigma_n^2 I)^{-1}\mathbf{x}\,, \quad (6)$$

$$v_t = k(x_t, x_t) - \mathbf{k}_\star^\top (K + \sigma_n^2 I)^{-1}\mathbf{k}_\star\,. \quad (7)$$

Here, $\mathbf{x} := x_{(t-\tau):(t-1)}$, $\mathbf{k}_\star := k(x_{(t-\tau):(t-1)}, x_t)$ and $K := k(x_{(t-\tau):(t-1)}, x_{(t-\tau):(t-1)})$. θ_m is given by the GP hyper-parameters $\lambda := (\xi; \sigma_n^2)$, where σ_n^2 is the noise variance. Notice that under such a model the GP hyper-parameters are assumed to be fixed *across* different regimes. If one desires to model changes in GP hyper-parameters at every change point, then the BOCPD algorithm dictates that one should integrate them out within the UPM. As a result, the UPM becomes:

$$p(x_t|\mathbf{x}, \phi) = \int p(x_t|\mathbf{x}, \lambda) p(\lambda|\mathbf{x})\ d\lambda \quad (8)$$

$$= \frac{1}{Z}\int p(x_t|\mathbf{x}, \lambda) p(\mathbf{x}|\lambda) p(\lambda|\phi)\ d\lambda\,, \quad (9)$$

where $Z := \int p(\mathbf{x}|\lambda)p(\lambda|\phi)\ d\lambda$. The expression $p(x_t|\mathbf{x}, \lambda)$ is given by (5). $p(\mathbf{x}|\lambda)$ is known as the marginal likelihood of the GP, the logarithm of which is given by:

$$-\frac{1}{2}\mathbf{x}^\top (K_\xi + \sigma_n^2 I)^{-1}\mathbf{x} - \frac{1}{2}\log|K_\xi + \sigma_n^2 I| - \frac{\tau}{2}\log 2\pi\,.$$

As the log marginal likelihood is a nonlinear function of λ, both the integrals present in (9) are intractable, even if one sets $p(\lambda|\phi)$ to be a Gaussian prior over λ. Consequently, we approximate these integrals using two methods, each with their own set of pros and cons. In the first technique we place a grid ($\{\lambda_g\}$) over a subspace of GP hyper-parameters that is assumed to be reasonable for the problem at hand (assigning uniform prior mass for each grid point). The integrals can then be approximated with sums:

$$p(x_t|\mathbf{x}, \lambda) \approx \sum_{\lambda_g} p(x_t|\mathbf{x}, \lambda_g)\left(\frac{p(\mathbf{x}|\lambda_g)}{\sum_{\lambda_g} p(\mathbf{x}|\lambda_g)}\right)\,. \quad (10)$$

Recall that it is tricky to apply more sophisticated quadrature algorithms for (9) as the target function is positive, and the interpolant runs the risk of becoming negative. The grid method does not scale with increasing dimensionality, however, it offers the opportunity to cache covariance matrices as one only considers a fixed set of hyper-parameter settings (see Section 3.1). An alternative which does scale with higher dimensionality is to use Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 1992). Say we have computed samples $\{\lambda_s\}$ representing the posterior $p(\lambda|\mathbf{x})$. Then,

$$p(x_t|\mathbf{x}, \lambda) \approx \sum_{\lambda_s} p(x_t|\mathbf{x}, \lambda_s)\,. \quad (11)$$

The samples can be updated sequentially for each run length hypothesis considered. The samples for $r_t = 0$ are straightforward as they come from the Gaussian prior $p(\lambda|\phi) = \mathcal{N}(\phi_m, \phi_v)$: we can trivially obtain iid samples at this stage. As the posterior $p(\lambda|x_{(t-\tau):(t-1)})$, represented by samples $\{\lambda_s^{(t-1)}\}$, will look similar to $p(\lambda|x_{(t-\tau):t})$, we can initialize the HMC sampler at $\{\lambda_s^{(t-1)}\}$ and run it for a short number of iterations for each sample. In practice, we have found

that 4 *trajectories* with a mean of 11 *leapfrog steps* give respectable results. Note that other Sequential Monte Carlo (SMC) methods could be used also, though we have not explored these options.

In an ARGP (Quiñonero-Candela et al., 2003) of order p, the past p values $x_{t-p:t-1}$ are taken as the GP input while the output is x_t:

$$x_t = f(x_{t-p:t-1}) + \epsilon_t, \qquad (12)$$
$$f \sim \mathcal{GP}(0, k),\ \epsilon_t \sim \mathcal{N}(0, \sigma^2)\,.$$

The ARGP can have more complex dynamics than the GPTS, but unlike GPTS it is restricted to the discrete time domain.

3.1. Methods to Improve Execution Time

Efficient computation is more tricky with GPs than iid models, and the computational bottleneck is the posterior predictive from the UPM $p(x_t|r_{t-1}, x_t^{(r)})$. Prediction in GPs is $\mathcal{O}(T^3)$ with T training points due to computing the inverse of the covariance matrix. Therefore, if we naively recomputed the GP every time step it would run $\mathcal{O}(T^5)$. Fortunately, we can apply *rank 1 updates* (Scholkopf & Smola, 2001) to the predictions since the training data only changes by one point every time step, lowering the computation to $\mathcal{O}(T^4)$. If pruning is applied the complexity can be reduced further to $\mathcal{O}(T\tilde{R}^2/\epsilon)$ or $\mathcal{O}(TK\tilde{R}^2)$, where $\tilde{R}$ is the typical max run length not pruned out.

Toeplitz Matrix Methods for GPTS If we are operating in a discrete-time setting, the covariance matrices involved for GPTS become *Toeplitz*. We can use a recursive relationship similar to the one used in solving Yule-Walker equations (see Golub & Van-Loan (1996)) in order to compute $K^{-1}\mathbf{k}_\star$ for all time steps in one $\mathcal{O}(T^2)$ sweep.[1]

Horizontal and Vertical Rank 1 Updates The precision matrix for prediction can be rebuilt from scratch every time step using *vertical updates*, or the precision matrices for each run length can be stored from the last step updating with *horizontal updates*.[2] Horizontal updates are compatible with arbitrary pruning while vertical updates are not.

[1] The method is identical to the Yule-Walker method, except that at the end we negate and reverse the result. This works because the inverse of a Toeplitz matrix is persymmetric.

[2] In horizontal updating $\Lambda_{t,r} = \text{Rank1Update}(\Lambda_{t-1,r})$, while in vertical it is $\Lambda_{t,r} = \text{Rank1Update}(\Lambda_{t,r-1})$, where $\Lambda_{t,r}$ is the precision matrix for the UPM at time t and run length r.

Algorithm 1 Learning (Lines marked with $\star$) and Run Length Estimation in BOCPD. Lines marked with † directly call the UPM.

function `getLikelihood` {All multiplications in function are element-wise}
$(\boldsymbol{\gamma}_0, \partial_h\boldsymbol{\gamma}_0, \partial_m\boldsymbol{\gamma}_0) \leftarrow (1, 0, 0)$ {Initialize the recursion, set hazard and UPM deriv. to 0}
† Initialize $\mathcal{S}$ to sufficient statistics of UPM prior
Define $\tilde{\boldsymbol{\gamma}}_t$ as $\boldsymbol{\gamma}_t[2\!:\!t+1]$
for $t = 1\!:\!T$ **do**
  † $\boldsymbol{\pi}_t^{(r)} \leftarrow p(x_t|\mathcal{S})$
  $\mathbf{h} \leftarrow H(1\!:\!t)$
  $\tilde{\boldsymbol{\gamma}}_t \leftarrow \boldsymbol{\gamma}_{t-1}\boldsymbol{\pi}_t^{(r)}(1-\mathbf{h})$ {Update the messages, no new change point, (3)}
  $\star\, \partial_h\tilde{\boldsymbol{\gamma}}_t \leftarrow \boldsymbol{\pi}_t^{(r)}(\partial_h\boldsymbol{\gamma}_{t-1}(1-\mathbf{h}) - \boldsymbol{\gamma}_{t-1}\partial_h\mathbf{h})$
  $\star\, \partial_m\tilde{\boldsymbol{\gamma}}_t \leftarrow (1-\mathbf{h})(\partial_m\boldsymbol{\gamma}_{t-1}\boldsymbol{\pi}_t^{(r)} + \boldsymbol{\gamma}_{t-1}\partial_m\boldsymbol{\pi}_t^{(r)})$
  $\boldsymbol{\gamma}_t[1] \leftarrow \sum \boldsymbol{\gamma}_{t-1}\boldsymbol{\pi}_t^{(r)}\mathbf{h}$ {Update the messages, there is a new change point, (3)}
  $\star\, \partial_h\boldsymbol{\gamma}_t[1] \leftarrow \sum \boldsymbol{\pi}_t^{(r)}(\partial_h\boldsymbol{\gamma}_{t-1}\mathbf{h} + \boldsymbol{\gamma}_{t-1}\partial_h\mathbf{h})$
  $\star\, \partial_m\boldsymbol{\gamma}_t[1] \leftarrow \sum \mathbf{h}(\partial_m\boldsymbol{\gamma}_{t-1}\boldsymbol{\pi}_t^{(r)} + \boldsymbol{\gamma}_{t-1}\partial_m\boldsymbol{\pi}_t^{(r)})$
  $p(r_t|x_{1:t}) \leftarrow \text{normalize}\,\boldsymbol{\gamma}_t$
  † Update sufficient statistics of posteriors $\mathcal{S}$
end for
$p(x_{1:T}) \leftarrow \sum \boldsymbol{\gamma}_T$ {1×1 Calculate the Evidence, message normalization constant}
$\star\, \partial p(x_{1:T}) \leftarrow (\sum \partial_h\boldsymbol{\gamma}_T, \sum \partial_m\boldsymbol{\gamma}_T)$
Return $(p(x_{1:T}), \partial p(x_{1:T}))$

4. Hyper-parameter Learning

It is possible to evaluate the (log) marginal likelihood of the BOCPD model at time T, as it can be decomposed into the one-step-ahead predictive likelihoods (see (1)):

$$\log p(x_{1:T}|\theta) \;=\; \sum_{t=1}^{T} \log p(x_t|x_{1:t-1}, \theta). \qquad (13)$$

Hence, we can compute the derivatives of the log marginal likelihood using the derivatives of the one-step-ahead predictive likelihoods. These derivatives can be found in the same recursive manner as the predictive likelihoods. Using the derivatives of the UPM, $\frac{\partial}{\partial\theta_m}p(x_t|r_{t-1}, x_t^{(r)}, \theta_m)$, and those of the hazard function, $\frac{\partial}{\partial\theta_h}p(r_t|r_{t-1}, \theta_h)$, the derivatives of the one-step-ahead predictors can be propagated forward using the chain rule, as shown in Algorithm (1). The derivatives with respect to the hyper-parameters can be plugged into a conjugate gradient optimizer to perform hyper-parameter learning. Alternatively, hyper-parameters can be marginalized out inside the UPM as in (9).

5. Results

We use three variants of our algorithm in the results: change point detection with an ARGP UPM (ARGP-CP), a GPTS UPM (GPTS-CP) with fixed GP hyper-parameters, and a nonstationary GP UPM that allows the GP hyper-parameters to change at every change point (NSGP). The variants are compared to ARGP, GPTS, and the Kalman filter on four real world data sets. We also compare them to the time independent model (TIM), modeling the data as iid normal, as a baseline. To quantify the performance of each of these models we evaluate them on test data using the one-step-ahead negative log predictive likelihood (NLL) in nats/observation and mean-square-error (MSE).[3]

The results shown all use the constant hazard function. Likewise, all the GPs use the squared-exponential covariance function.

5.1. Nile Data

We first consider the Nile data set,[4] which has been used to test many change point methods (Garnett et al., 2009). The data set is a record of the lowest annual water levels on the Nile river during 622–1284 measured at the island of Roda, near Cairo, Egypt. There is domain knowledge suggesting a change point in year 715 due to an upgrade in ancient sensor technology to the nilometer.

We trained the (hyper) parameters of all the models on data from the first 200 years (622–821). The predictive performance was evaluated on the following years, 822–1284. The run length posterior of NSGP on the Nile can be seen in Figure 1(b). The installation of the nilometer is the most visually noticeable change in the time series. We also see that the only change point the NSGP is completely confident in occurs shortly after the nilometer installation.

Quantitative results in predictive performance are shown in Table 1. We see from the performance of GPTS-CP that the change point capability does not help GPTS, but it does help ARGP. The Kalman filter does reasonably well on this data set. By analyzing the scatter plots of the Nile data it can be seen that the system is quite linear, satisfying the assumption of the Kalman filter, but possibly the noise level in the system is changing over time. This allows the NSGP to detect the changes and outperform both GPTS-CP and the Kalman filter. The error bars are larger than we would like, but not unexpected given that only 462 points are available after removing the 200 used in training.

5.2. Bee Waggle Dance Data

Honey bees perform what is known as a waggle dance on honeycombs. The three stage dance is used to communicate with other honey bees about the location of pollen and water. Ethologists are interested in identifying the change point from one stage to another to further decode the signals bees send to one another. The bee data set contains six videos of sequences of bee waggle dances.[5] The video files have been preprocessed to extract the bee's position and head-angle at each frame. While many in the literature have looked at the cosine and sine of the angle, we chose to analyze angle *differences*.

We illustrate applying NSGP to "sequence 1" of the bee data in Figure 1(a); training on the first 250 frames (four change points) and testing on the remaining 807 frames (15 change points). We see that the NSGP identifies the structure in the data despite only observing four different dance stages during the training period. Figure 1(a) also shows the output of the CUSUM algorithm when applied to the bee data; we fit the parameters as recommended by Grigg & Spiegelhalter (2008). The CUSUM appears to alert far too many times after a change point or not at all. Additionally it does not provide a run length distribution; so we do not know if CUSUM is referring to a recent change point or an old one when it alerts. The CUSUM can not correct for a mistaken alert after observing more data as BOCPD can by simply adjusting the run length distribution.

In the quantitative results we find the ARGP-CP does significantly better than the other methods. The GPTS-CP and NSGP do not do as well possibly because the x and y positions over time can be explained well by a stationary GPTS model.

5.3. Snowfall Data

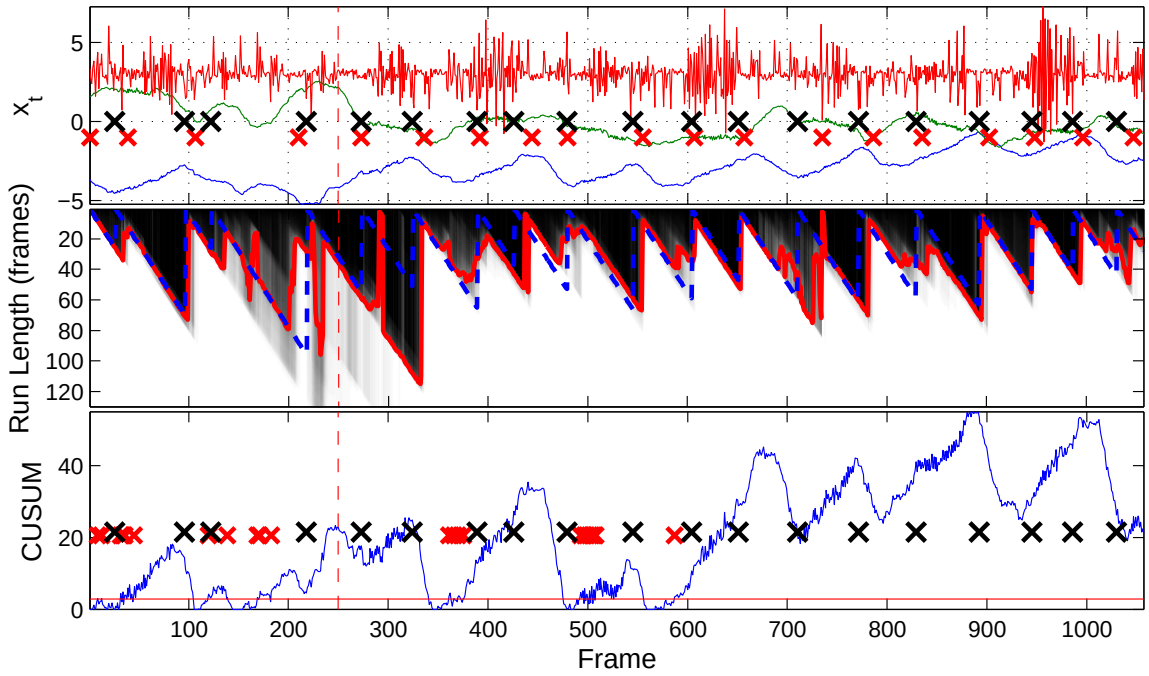
We also used historical daily snowfall data in Whistler, BC, Canada,[6] to evaluate our change point models. The models were trained on two years of data. We evaluated the models' ability to predict next day snowfall using 35 years of test data. A probabilistic model of the next day snowfall is of great interest to local skiers. In this data set, being able to adapt to different noise levels is key: there may be highly volatile

[3] MSE is given by $\sum_t \|\mu_t - x_t\|^2$ where μ_t is the predictive mean.

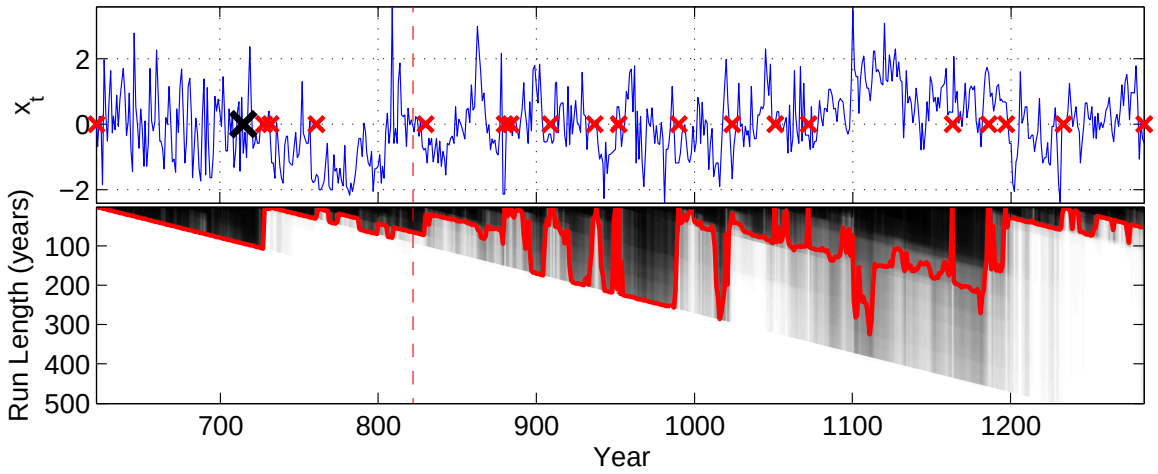
[4] http://lib.stat.cmu.edu/S/beran

[5] http://www.cc.gatech.edu/~borg/ijcv_psslds/

[6] http://www.climate.weatheroffice.ec.gc.ca/ (Whistler Roundhouse station, identifier 1108906).



(a) **Bee Waggle Dance**: The output BOCPD, video sequence one of six. **Top**: Due to the multivariate nature of the time series we artificially separated them in plotting for increased readability. The time series are the bee's: x coordinate (blue), y coordinate (green), and angle (red). The vertical dashed red line represents the boundary between train and test sets. Note that there are only four true change points in training yet NSGP can still identify the structure successfully. **Middle**: The true run length is marked in dashed blue to complement the posterior median in solid red. The large black cross marks the labeled switches in the bee's dance. The small red crosses mark alert locations where the probability of a change point under the run length posterior since the last alert exceeds 0.95. Using a conservative alert threshold of 0.95 we are able to match the true change points quite well while still keeping a low time to alert. **Bottom**: We show the common frequentist change point detector CUSUM as a comparison. The solid blue line is the CUSUM statistic, the horizontal solid red line is the alert threshold set to attain a false positive rate of 5%, the red crosses are when the CUSUM crosses the threshold, and the black crosses are again the true change points.



(b) **Nile Record:** The output NSGP, 622–1284. **Top**: The large black cross marks the installation of the nilometer in 715. The small red crosses mark alert locations where the probability of a change point under the NSGP posterior since the last alert exceeds 0.50. Given that the threshold is only 0.50 the NSGP is fairly liberal in alerting. **Bottom**: The run length CDF and its median (solid red).

Figure 1. Applying NSGP to the Nile and Bee data sets.

Table 1. Variants of Gaussian process BOCPD, marked with a ♡, compared against other methods on the Nile data, bee data, and snow data. The results are provided with 95% error bars and the p-value testing the null hypothesis that methods are equivalent to the best performing method, according to NLL, using a one sided t-test.

Method	Negative Log Likelihood	p-value	MSE	p-value
Nile Data (200 Training Points, 462 Test Points)				
GPTS	1.19±0.0548	0.196	0.579±0.0976	0.356
♡ GPTS-CP	1.19±0.0548	0.167	0.583±0.0989	0.335
ARGP	1.18±0.0510	0.202	0.568±0.0940	0.410
♡ ARGP-CP	$\mathbf{1.15 \pm 0.0555}$	N/A	$\mathbf{0.553 \pm 0.0962}$	N/A
Kalman	1.17±0.0508	0.361	0.562±0.121	0.453
TIM	1.49±0.0714	<0.001	1.16±0.161	<0.001
♡ NSGP (grid)	1.15±0.0655	0.490	0.585±0.0988	0.321
Bee Waggle Dance Data (250 Training Points, 807 Test Points)				
GPTS	8.02±0.504	<0.001	8.44±0.745	<0.001
♡ GPTS-CP	4.54±0.188	<0.001	3.13±0.241	<0.001
ARGP	4.35±0.167	0.007	2.98±0.224	0.008
♡ ARGP-CP	$\mathbf{4.07 \pm 0.150}$	N/A	$\mathbf{2.62 \pm 0.195}$	N/A
Kalman	4.39±0.176	0.002	2.93±0.215	0.016
TIM	4.54±0.177	<0.001	3.25±0.237	<0.001
♡ NSGP (HMC)	4.19±0.212	<0.001	3.17±0.230	<0.001
Whistler Snowfall Data (500 Training Points, 13380 Test Points)				
GPTS	1.48±0.0455	<0.001	0.780±0.0333	<0.001
♡ GPTS-CP	1.17±0.0183	<0.001	0.689±0.0294	<0.001
ARGP	1.31±0.0395	<0.001	0.637±0.0268	0.143
♡ ARGP-CP	-0.604±0.0385	<0.001	0.750e±0.0315	<0.001
Kalman	1.28±0.0373	<0.001	$\mathbf{0.614 \pm 0.0254}$	0.589
TIM	1.47±0.0284	<0.001	1.01±0.0387	<0.001
♡ NSGP (grid)	$\mathbf{-1.98 \pm 0.0561}$	N/A	0.618±0.0242	N/A

snowfall during a storm and then no snow in between storms. Hence, the NSGP has an advantage in being able to adapt its noise level.

5.4. Industry Portfolio Data

We also tried a multivariate data set: the "30 industry portfolios" data set,[7] which was also used in the context of change point detection by Xuan & Murphy (2007). The data consists of daily returns of 30 different industry specific portfolios from 1963 to 2009. The portfolios consist of NYSE, AMEX, and NASDAQ stocks from industries such as food, oil, telecoms, etc.

In Figure 2, we show that the change points found coincide with significant events with regard to the stock market: the climax of the Internet bubble, the burst of the Internet bubble, and the 2004 presidential election. The methods used in Xuan & Murphy (2007) did not find any correspondence with historical events. For financial returns the iid assumption within a regime is quite reasonable, so as expected the GP finds very long covariance length scales, attributing the variability within each regime to noise.

[7] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/det_30_ind_port.html

6. Conclusion

We develop inference techniques to extend Gaussian process time series models, GPTS and ARGP, to account for change points, by combining them with a change point model designed to handle nonstationary time series data. We also show how to do hyper-parameter learning in these models using hyper-parameter optimization, via maximizing the marginal likelihood. We have illustrated the use of BOCPD on a diverse set of real world examples including multivariate data sets (bee data and industry), and ones with ground truth change points (bee data).

Acknowledgments

We thank Finale Doshi-Velez, Steven Bottone, and Zoubin Ghahramani for their useful suggestions. This work was supported by Rockwell Collins, Inc.

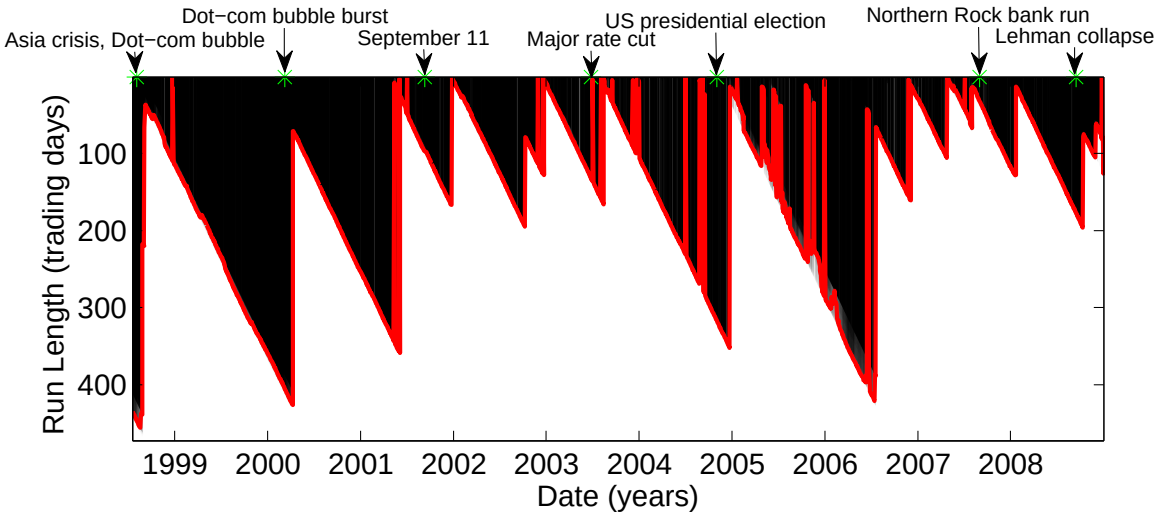


Figure 2. **Industry Portfolios**: BOCPD run length distribution in 1998–2008. The shading represents the CDF of the run length distribution, while the solid red line represents the median of the distribution. Areas of a quick transition from black (CDF of zero) to white (CDF of one) indicate a sharply peaked run length distribution. Many events of market impact create change points. Some of the other change points correspond to minor rallies or rate cuts but not a historical event. Note that we plot the filtering distribution, so the vertical spikes are change points BOCPD thinks are plausible but then realizes are false upon receiving more data.

References

Adams, R. P. and MacKay, D. J. C. Bayesian online changepoint detection. Technical report, University of Cambridge, Cambridge, UK, 2007.

Barry, D. and Hartigan, J. A. A Bayesian analysis of change point problems. *Journal of the American Statistical Association*, 88:309–319, 1993.

Duane, S., Kennedy, A. D., Pendelton, P. J., and Roweth, D. Hybrid Monte Carlo. *Physics Letters B, Volume 195, Issue 2, p. 216–222*, 195(2):216–222, 1987.

Fearnhead, P. and Liu, Z. Online inference for multiple changepoint problems. *Journal of the Royal Statistical Society*, 69:589–605, 2007.

Garnett, R., Osborne, M., and Roberts, S. Sequential Bayesian prediction in the presence of changepoints. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 345–352, Montreal, QC, June 2009.

Golub, G.H. and Van-Loan, C. F. *Matrix Computations*. The Johns Hopkins University Press, 3 edition, 1996.

Grigg, O. A. and Spiegelhalter, D. J. An empirical approximation to the null unbounded steady-state distribution of the cumulative sum statistic. *Technometrics*, 4:501–511, November 2008.

Jones, S. The formula that felled Wall Street. In *Financial Times*. April 24 2009.

Li, D. X. On default correlation: A copula function approach. Working paper 99-07, The RiskMetrics Group, 44 Wall St. New York, NY 10005, April 2000.

Murray-Smith, R. and Girard, A. Gaussian process priors with ARMA noise models. In *Irish Signals and Systems Conference*, pp. 147–152, Maynooth, Ireland, 2001.

Neal, R. M. Bayesian training of backpropagation networks by the hybrid Monte Carlo method. Working paper CRG-TR-92-1, University of Toronto, Connectionist Research Group, ON, Canada, April 1992.

Quiñonero-Candela, J., Girard, A., Larsen, J., and Rasmussen, C. E. Propagation of uncertainty in Bayesian kernel models—application to multiple-step ahead forecasting. In *ICASSP 2003*, volume 2, pp. 701–704, Hong Kong, April 2003.

Rasmussen, C. E. and Ghahramani, Z. Bayesian Monte Carlo. In *NIPS 15*, Vancouver, BC, Canada, 2002.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

Scholkopf, B. and Smola, A. J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Cambridge, MA, 2001.

Xuan, X. and Murphy, K. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, USA, 2007.