Notes on Bayesian Changepoint Detection

Philip Bretz

November 19, 2020

1 Bayesian Online Changepoint Detection

1.1 Introduction

In nearly all contexts where we have data, we encounter situations where that data rapidly changes. Consider an earthquake, where the movement of the ground is essentially negligible until the event occurs, leading to substantially different behavior. Or consider the communications between a network of people, where during a normal period, the number of communications remain roughly the same, but after some shocking event, the chatter dramatically increases.

In any scenario where we encounter different regimes of behavior, we need a method for identifying when those changes occur. There is always a balance between falsely classifying normal deviations as change and missing real change when it occurs. This becomes even more challenging in contexts where we may not even know the parameters that govern the data in each regime.

This is precisely where Bayesian methods shine. The Bayesian approach allows us to quantify our uncertainties in a natural way and update our beliefs about unknown quantities based on incoming data. This is Bayesian online changepoint detection.

1.2 Overview

The standard Bayesian approach to changepoint detection, as described in Adam and MacKay's *Bayesian Online Changepoint Detection* [1], is estimating the posterior distribution of the run length of the current regime. Essentially, we want to have an understanding, based on the observed data up to that point, of how long it has been since the last changepoint.

The algorithm's efficiency is due to the fact that only one of two things can happen in a time step: either the run length can increase by 1 (remain in current regime) or it can drop to 0 (changepoint to next regime).

The basic principle of the algorithm is 'message passing'. When a new datum comes in at time t, the algorithm first evaluates the predictive distribution at that value for each possible

run length. This gives us, for each run length, a measure of the likelihood of the new data.

Next is the 'message passing' portion of the algorithm. The posterior probability for each run length is known at time $t-1$. For each run length at time $t-1$, we calculate the probability that the run length increased by 1 by updating according the likelihood of the new datum. This gives us the desired posterior probabilities for run lengths of $1, 2, ...$ at time t.

Lastly, the algorithm calculates the probability of a run length of 0 at time t. This probability is exactly the probability of a changepoint at time t. We then have what we want; the posterior distribution of run lengths at time t.

1.3 Probabilistic Basis

The following theory is drawn directly from Adam and MacKay [1], with minor alterations.

Let the data be $x_1, x_2, ..., x_T$. We assume that the data are partitioned into different regimes $\rho = 0, 1, ...$ and the delineation between regimes are the changepoints. We further assume that the data are i.i.d. from a probability distribution $P(x_t|\eta_\rho)$, where η_ρ denotes the parameters associated with regime ρ. I.e., the data follows the same distribution for all time, but in each regime the parameters of the distribution change.[1]

Let the length of the current run at time t be denoted by r_t. Denote the r most recent data points at time t by $x_t^{(r)}$. For example, $x_{100}^{(3)} = x_{100}, x_{99}, x_{98}$. If $r = 0$, this set is empty. This is the set of past data points in the current regime for a run length equal to r.

To find the posterior distribution of run lengths

$$P(r_t|x_{1:t}) = \frac{P(r_t, x_{1:t})}{P(x_{1:t})}$$

we use the law of total probability over the run lengths at $t-1$:

$$\begin{aligned} P(r_t, x_{1:t}) &= \sum_{r_{t-1}} P(r_t, r_{t-1}, x_{1:t}) \\ &= \sum_{r_{t-1}} P(r_t, x_t|r_{t-1}, x_{1:t-1}) P(r_{t-1}, x_{1:t-1}) \\ &= \sum_{r_{t-1}} P(r_t|r_{t-1}) P\left(x_t|r_{t-1}, x_{1:t-1}\right) P(x_{1:t-1}) P(r_{t-1}|x_{1:t-1}) \end{aligned}$$

Since $P(x_{1:t})$ and $P(x_{1:t-1})$ are only normalizing constants, we have

$$P(r_t|x_{1:t}) \propto \sum_{r_{t-1}} P(r_t|r_{t-1}) P(x_t|r_{t-1}, x_{1:t-1}) P(r_{t-1}|x_{1:t-1}) \tag{1}$$

[1] The algorithm can be altered to incorporate a model where there is only a single changepoint, but the two regimes follow different distributions.

Since $P(r_{t-1}|x_{1:t-1})$ is known from the previous step, we only need to be able to (1) calculate the conditional run length probability $P(r_t|r_{t-1})$ and (2) evaluate the predictive distribution (for each r_{t-1}) at the new datum x_t.

1.4 Conditional Run Length Probability

In a given time step, there are only two things that can happen with the run length. Either the run length can increase by 1 (growth) or it can drop to 0 (changepoint). So, the conditional run length probability takes the form

$$P(r_t|r_{t-1}) = \begin{cases} 1 - H(r_{t-1}+1) & \text{if } r_t = r_{t-1}+1 \\ H(r_{t-1}+1) & \text{if } r_t = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $H(\tau)$ is the hazard function, describing how likely a changepoint is to occur at a run length of τ. The simplest choice is to make this process memoryless; set $H(\tau) = 1/\lambda$ constant. However, a more natural choice in our context may be to make $H(\tau)$ increasing to penalize longer run lengths.

1.5 Predictive Distribution

A key part of the algorithm is evaluating the predictive distribution (conditional on each possible run length) at the new datum. Conditioning on run length exactly means that we look at the predictive distribution conditional only on the most recent data (data in that run length). I.e.

$$P(x_t|r_{t-1}, x_{1:t-1}) = P(x_t|x_{t-1}^{(r_{t-1})}) \tag{3}$$

where we assume the recent data $x_{t-1}^{(r_{t-1})}$ all comes from a single regime (and is all the past data associated with that regime).

When the data all comes from a single regime, we need only apply the standard Bayesian approach to evaluating the predictive distribution:

$$P(x_t|x_{t-1}^{(r_{t-1})}) = \int_{\eta_\rho} P(x_t|\eta_\rho) P(\eta_\rho|x_{t-1}^{(r_{t-1})}) d\eta_\rho$$

and

$$P(\eta_\rho|x_{t-1}^{(r_{t-1})}) = \frac{P(\eta_\rho)P(x_{t-1}^{(r_{t-1})}|\eta_\rho)}{P(x_{t-1}^{(r_{t-1})})}$$

where ρ is the current regime and η_ρ is the (unknown) parameter governing the data.

For an arbitrary prior-likelihood combination these calculations may be intractable, requiring MCMC or other numerical techniques. However, for a conjugate model, the calculations are typically very straightforward and usually just require tracking how a couple hyperparameters (or sufficient statistics) of the prior $P(\eta_\rho)$ respond to the data $x_{t-1}^{(r_{t-1})}$.

Typically, if α and β are the original sufficient statistics of the prior distribution $P(\eta_\rho)$, with new data $y_1, ..., y_n$ they are updated in the following form:

$$\alpha' = \alpha + v(n) \text{ and } \beta' = \beta + u(y_{1:n}) \tag{4}$$

where v and u are simple functions. Moreover, the predictive distribution $P(y_{pred}|y_{1:n})$ is typically a known function of the updated hyperparameters α', β':

$$P(y_{pred}|y_{1:n}) = P(y_{pred}|\alpha', \beta') \tag{5}$$

Denote the hyperparameters associated with data $x_{t-1}^{(r_{t-1})}$ by $\alpha_{t-1}^{(r_{t-1})}, \beta_{t-1}^{(r_{t-1})}$. We can then easily calculate Equation 3 by

$$P(x_t|r_{t-1}, x_{1:t-1}) = P(x_t|\alpha_{t-1}^{(r_{t-1})}, \beta_{t-1}^{(r_{t-1})}) \tag{6}$$

For example, with our data, I decided that the data (after some transformations) was drawn from a normal distribution with mean 0 and unknown variance. By then choosing a gamma inverse prior on the variance, it is a conjugate model. The hyperparameters of the gamma inverse distribution update with new data in a very straightforward manner and the resulting predictive distribution is a t distribution with mean 0 whose degrees of freedom and variance depend on the updated hyperparameters.

1.6 Algorithm

From p. 3 of Adams and MacKay [1], with minor modifications:

1. Initialize[a]
$$P(r_0 = 0) = 1$$
2. Observe new datum x_t
3. Evaluate predictive probability for every possible value of r_{t-1}
$$\pi_t^{(r_{t-1})} = P(x_t|\alpha_{t-1}^{(r_{t-1})}, \beta_{t-1}^{(r_{t-1})})$$
4. Calculate growth probability for each possible value of r_{t-1} up to a constant C [b]
$$C \cdot P(r_t = r_{t-1} + 1|x_{1:t}) = (1 - H(r_{t-1} + 1))\pi_t^{(r_{t-1})} P(r_{t-1}|x_{1:t-1})$$
5. Calculate changepoint probability up to a constant C
$$C \cdot P(r_t = 0|x_{1:t}) = \sum_{r_{t-1}} H(r_{t-1} + 1)\pi_t^{(r_{t-1})} P(r_{t-1}|x_{1:t-1})$$
6. Normalize to find run length distribution
$$P(r_t|x_{1:t}) = \frac{C \cdot P(r_t|x_{1:t})}{\sum_{r_t} C \cdot P(r_t|x_{1:t})}$$
7. Return to step 2

[a] If we are not certain the first run begins at time 0, we can place a corresponding distribution on r_0.

[b] The constant C is exactly $P(x_{1:t-1})/P(x_{1:t})$

One thing to note is that the sum over r_t (or r_{t-1}) is always finite. Why? The run length can never be larger than the current time, so $r_t \leq t$. Alternatively, the term $P(r_t|x_{1:t}) = 0$ for $r_t > t$.

Code Snippet: Below is a code snippet of the basic algorithm for finding the run probabilities. It takes in the data **x**, and a **model** that contains the prior distribution of the parameters η and the choice of hazard function $H(\tau)$. It returns a numpy array **run_probs** of run probabilities where each row is the posterior distribution of run lengths at the given time.

```
def run_prob(x, model):
    # Each row is a time frame
    T = len(x)+1
    run_probs = np.zeros((T,T))
    # Initialize at time 0
    run_probs[0,0] = 1
    for i in range(len(x)):
        # Evaluate predictive probabilities
        pred = predictive(x[0:(i+1)], model)
        # Calculate changepoint probability
        run_probs[i+1,0] = cp_prob(i, run_probs, pred, model)
        # Calculate growth probabilities
        for s in range(1,i+2):
            prob, p = run_probs[i,s-1], pred[s-1]
            run_probs[i+1,s] = growth_prob(prob, p, model)
        # Normalize the row
        run_probs[i+1,] = run_probs[i+1,]/sum(run_probs[i+1,])
        # Truncate very small probabilities and re-normalize
        run_probs[i+1,] = truncate(run_probs[i+1,])
    return run_probs
```

1.7 Locating Changepoints

Adams and MacKay describe in depth the algorithm for determining the run probabilities. However, they omit a method for determining the location of a changepoint based on those run probabilities, other than visual clues.

The basic idea behind detecting changepoints from the run length distributions is fairly straightforward. Inside of a regime the probability mass shifts up by one at every time step, and at a particular time t, if the center of the mass is r, the beginning of the regime was at time $t - r$.

To detect when a changepoint is occurring, ideally we want to find a time when the probability mass shifts to 0. However, because the updating process can be slow to respond to change, we do not usually see a clear shift to 0 in a single time step.

A better approach is to locate a time t when the probability mass was centered at $r^{(t-1)}$ at the previous time, but instead of moving to $r^{(t-1)} + 1$ decreased significantly to a value $r^{(t)}$. We can then extrapolate back to estimate that the changepoint occurred $r^{(t)}$ frames ago, or at time $t - r^{(t)}$.

My current implementation is very crude. At each time t, I locate the maximum probability, and let that location be $r^{(t)}$. I let a 'significant decrease' be a decrease of more than 5, though the function I wrote allows that parameter to be chosen by the user. Even so, it

works decently.

The following is a more natural method for selecting the probability mass center $r^{(t)}$. If I bin the run length distribution where I let the bin size be chosen by the user and take the value at each bin to be the sum of the probabilities inside it, that will work better at determining shifts in the center of probability.

Code Snippet: Below is a code snippet of a basic algorithm for detecting regime changes given an array of run length probabilities. It takes in the array **probs** and a value **tol** that determines how much of a movement of the location of maximum probability defines a changepoint. The function **Regime(ID, location, detected)** produces an object of class 'Regime' where the first entry defines which regime it is, the second entry defines where the regime began, and the third entry is when the regime was detected. When I ran my code on the data, I set **tol** to 5.

```
def detect(probs, tol=0):
    T = probs.shape[0]
    prev_max_loc = 0
    regimes = [Regime(0, 0, 0)]
    for i in range(1,T):
        current_max_loc = np.argmax(probs[i,])
        if abs(prev_max_loc-(current_max_loc-1)) > tol:
            regimes.append(Regime(number, i-cunrrent_max_loc, i))
        prev_max_loc = current_max_loc
    return regimes
```

1.8 Notes on Efficiency

Implemented exactly as written, this algorithm is $O(n^2)$ in time and memory. Clearly, this is not optimal if we are looking at a long time series. A simple solution I implemented was placing a cap on the possible values of r_t. This cap would represent our belief in the maximum possible length of a regime time. Introducing such a cap cuts both time and memory down to $O(n)$.

We can cut down storage even more if we only keep track of the run length distribution at the current time and previous time. If we do this, whatever process we use to find changepoints will need to be done inside the algorithm.

Another issue is that for run length values that the algorithm finds to be exceedingly unrealistic, the probability could be small enough that we encounter floating point errors. A simple solution I implemented is to continually round values in the run length distribution below a set $\epsilon > 0$ down to 0 at every time step. This is the **truncate()** function in the code snippet.

2 Our Data

2.1 Context

The data I am examining is drawn from a stick-slip simulation where constant shearing force is applied to a system of granular media. The simulations are designed to mimic earthquake behavior, where pressure in the system builds up due to shearing force, eventually resulting in a slip, followed by another period of pressure build up, and so on.

Using persistent homology, my colleagues calculated a persistence diagram for each frame of the simulation. The data that I am looking at is, for each time, the Wasserstein distance between the current persistence diagram and the persistence diagram of the previous frame. Essentially, this is a measure of how the system changes as a whole at each time.

With this time series that gives a broad overview of the system, my colleagues are interested in predicting when an upcoming slip will occur. Consequently, this is an extremely natural application of changepoint detection, where different regimes in the data have a strong physical interpretations (slip, stick, just before slip, etc.).

2.2 Transformations

The raw data is not quite conducive to direct application of the changepoint algorithm. Preferably, the data in each regime would appear to be derived from the same distribution, but with different parameters for each regime. Looking at the example data below, this is clearly not the case. This is data taken from the beginning of the 7th slip until the end of the 8th slip, or from time 4798 to 5268.

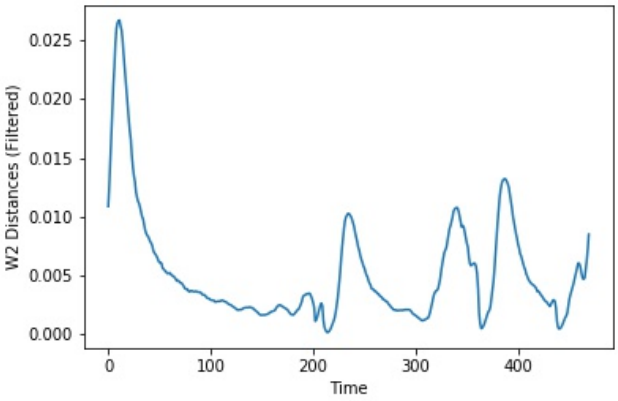

Figure 1: Raw Data

An example of what we would prefer is data that is normally distributed in each regime, where each regime has a different variance. If we take the logarithm of the data and then look at the second difference, we get a time series that appears to do exactly that. While

many features are no longer available (increasing, decreasing, or slow changes in concavity), we can see sharp changes very clearly. These correspond to 'jerk' in the system.

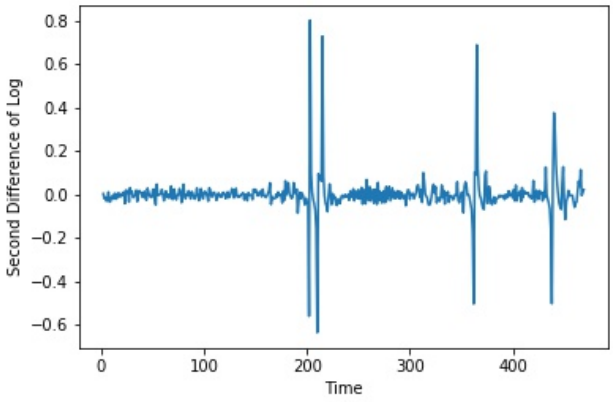

Figure 2: Second Difference

2.3 Model

Assuming the data is normally distributed with an unknown variance, it has a likelihood of

$$x \sim N(0, \sigma^2).$$

The natural choice of prior for a conjugate model is the scaled inverse chi-squared distribution with hyperparameters ν and σ_0^2. The hyperparameters update, with data $x_1, ..., x_n$ by

$$\nu' = \nu + n \text{ and } (\sigma_0^2)' = \frac{\nu\sigma_0^2 + \sum_{i=1}^{n} x_i^2}{\nu + n}.$$

The posterior predictive distribution is the T-distribution on ν degrees of freedom and σ_0^2 estimated variance, or

$$x_{pred}|x_1, ..., x_n \sim t_{\nu'}(0, (\sigma_0^2)').$$

I decided on a memory-less function for the conditional run distribution, or

$$r_t|r_{t-1} = \begin{cases} \frac{1}{\lambda} & \text{if } r_t = 0 \\ 1 - \frac{1}{\lambda} & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{otherwise} \end{cases}$$

I chose a vague prior $\nu_{prior} = 1, (\sigma_0^2)_{prior} = 1$ and $\lambda = 100$.

2.4 Preliminary Results

My algorithm needs some fine tuning, but overall, it worked fairly well. Below are the plots with detected changepoints marked.

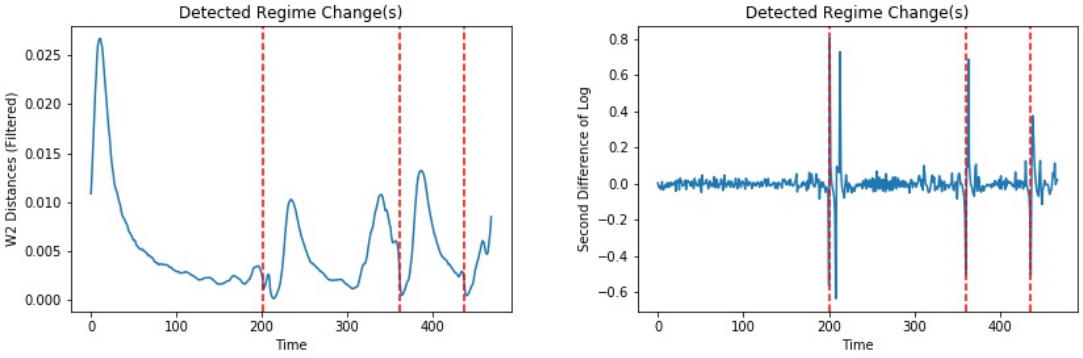

Figure 3: Regime Detection

The algorithm missed the event around the 300th frame, which is to be expected. Why? The second difference, which is what we are applying the algorithm to, does not have a sharp variance change for the corresponding mini-slip in the W_2 distances.

If we then look at where these detected changepoints lie on the plot of the horizontal wall velocity (velocities above a certain threshold are defined as slip frames), we see exactly what we hope; they precede an increase in the velocity.

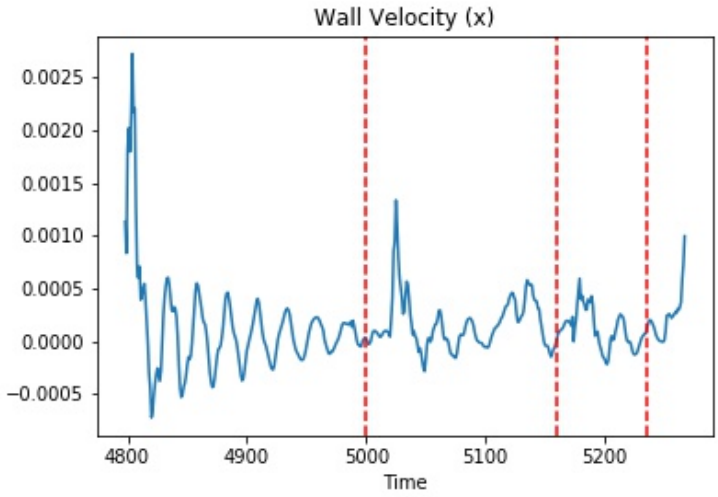

Figure 4: Wall Velocity

2.5 Future Plans

First, I need to make my model slightly more sophisticated. A natural way to do this will be by incorporating the first differences:

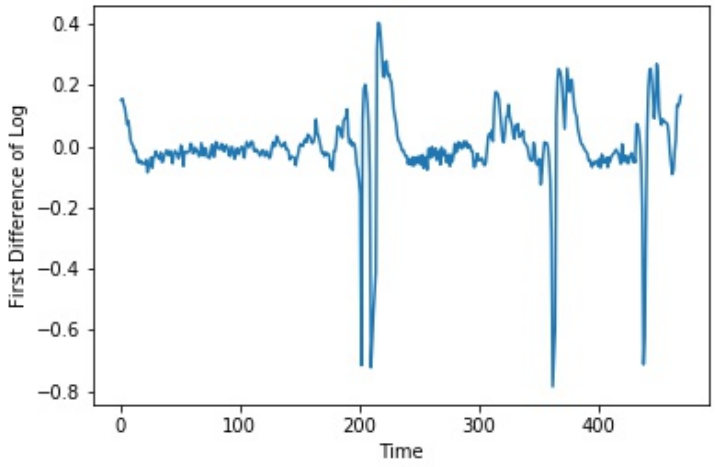

Figure 5: First Difference

I want to use the idea that changepoints occur not just at locations of large variance, but also at places like frame 300, where the first difference sharply increases to positive values.

Until I decide on the appropriate way to incorporate that into the model, I will locate changepoints from the second difference for some other time frames to see how the detection performs.

Ideally, I will be able to locate all changepoints. In each regime, I will use learning algorithms to understand how different features of the curve affect the duration of the regime. With an appropriate model, there should be a few key parameters that I can use to predict the time to the next slip event, as well as its duration and intensity.

References

[1] Ryan Prescott Adams and David J.C. MacKay. Bayesian online changepoint detection. *Arxiv*, 2007.