Bayesian Online Changepoint Detection

- Ryan Prescott Adams, David J.C. MacKay

Goal and Contributions

The objective of the paper Bayesian Online Changepoint Detection (BOCD) by Ryan and David JC is **identification of abrupt changes of the generative parameters of sequential data.**

A Change Point (CP) is a specific sequential position in the time domain at which the underlying distribution parameters undergoes a change. The paper provides an algorithm to **determine the posterior distribution of the 'length of the current run', which denotes the number of time steps since the last changepoint was observed, given the data observed so far.** The paper also provides a method to determine an accurate distribution of the next unseen datum in the sequence, given only data already observed**. Highly modular implementation** so that the algorithm may be applied to a variety of types of data.

Frequentist methods have yielded online filtering and prediction techniques, most Bayesian approaches to changepoint detection have been offline and retrospective. This paper provides a novel **recursive Bayesian changepoint detection algorithm for online inference**.

Use cases

For many applications in machine intelligence, Change Point Detection is a natural requirement. Robots must navigate based on past sensor data from an environment that may have abruptly changed. Change point models have been applied in a wide scope of settings including **finance, biometrics, econometrics, EEG analysis, process control, disease demographics, DNA segmentation,** and in different zones where long sequential of data are accessible. Change point (CP) detection has also many implications in the climate modelling, speech-recognition, image analysis and human activity systems.

Algorithm to determine Run-Length Distribution

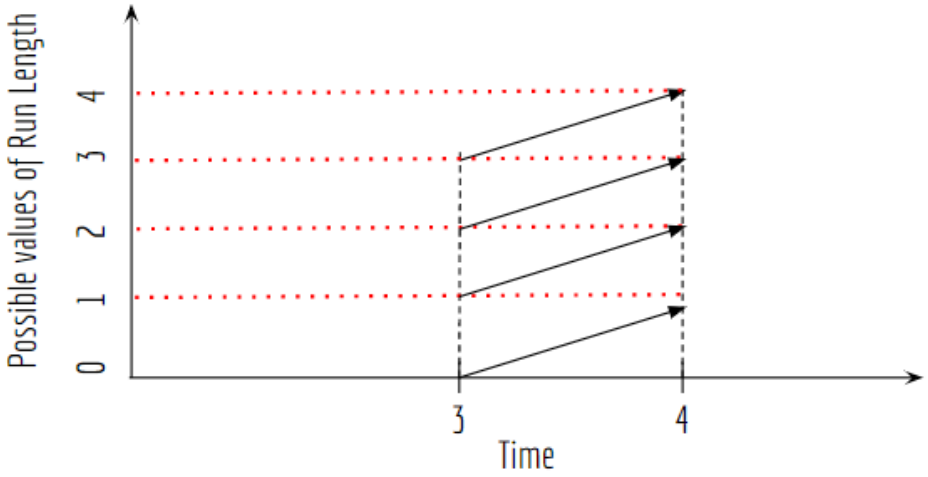
Assumptions: iid nature of datapoints within segments as well as the iid nature of the parameters of the data generating process across segments, and the family of the data generating processes across segments does not change and there is only change in parameters.

Step 1: Initialize the priors of the run-lengths (either 1 or the normalized survival function) and the hyperparameters for UPM predictive probabilities

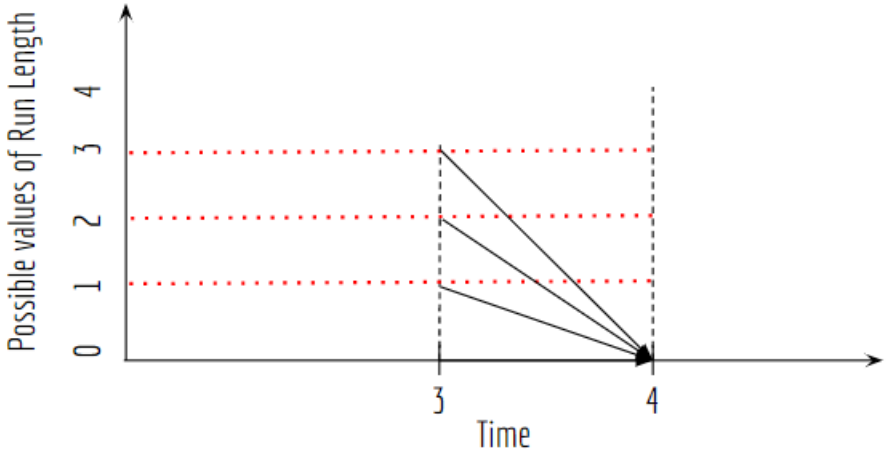
Step 2: Observe a new datum

Step 3: Compute Underlying Probabilistic Model (UPM) Predictive Probabilities - rather than computing the UPM predictive by computing the EF posterior and integrating out the

parameters, we just need to keep track of the hyperparameters by time t-1 to make a prediction at time t.



Step 4: Compute the Growth Probabilities (Run length increases by 1) – This is calculated recursively for the case when r(t) = r(t-1) + 1 using the UPM predictive probability and hazard function. For example, here r(4) could be 1,2,3,4 depending on the probability of r(3).



Step 5: Compute the Changepoint Probabilities (Run Length becomes 0): This is calculated recursively for the case when r(t)= 0. This means that r(t-1) values could be anything ranging from 0 to t-1. For example, in the for r(4) = 0, r(3) could have been 0,1,2,3 or 4. Here we sum over all the possibilities.

Step 6: Calculate the normalizing factor/evidence

Step 7: Determine Run-Length Distribution - Changepoint Probabilities/Normalizing Factor

Step 8: Update the parameters of the UPM distribution - we have an efficient way to compute the updated hyperparameters, which allow us to compute the posterior predictive without integration

Step 9: Return to Step 2

Note: This UPM predictive distribution is usually a simple function of the sufficient statistics. These sufficient statistics are additive and can therefore be computed sequentially. In case the UPM is a general function, UPM predictive has to be calculated via integration which increases the complexity.

Goodness of the Model

The **space- and time-complexity of the algorithm per time-step are linear in the number of data points so far observed**. This stands as a significant improvement to the exponential time complexity for a brute force approach. So, the this model serves its purpose of finding change point effectively and efficiently. This framework **provides convenient delineation between the implementation of the changepoint algorithm and the implementation of the model**. It leverages conjugate-exponential models to achieve modularity and efficient parameter estimation.

Shortcomings and Limitations

Assumptions include **iid nature of datapoints within segments as well as the iid nature of the parameters of the data generating process across segments, and the family of the data generating processes across segments does not change**. These assumptions often do not hold in real systems. The model is highly sensitive to **the initialization of the hyperparameters and the assumed priors**. Therefore, mis-specified model or prior may not result in reliable performance. The model could be made more robust by developing approximate inference routines for **non-conjugate models**, where it is difficult to integrate with the changepoint detection scheme described in the paper. **Memory usage** to compute the Run length distribution increases quadratically with time.