

# Apply machine learning to Performance trend analysis

Araya Eamrurksiri

Linkoping university

May 19, 2017

- Many test cases are executed for testing software packages
- Evaluate how each software package performs for an updated software package
- Tool or algorithm that can reduce workload of manual inspection

- Detect the state of the CPU utilization (degrading, improving or steady state)
- Detect whether there is any change in the test environment that effects the CPU utilization

- Software release
- Software package - treated as a time point in the time series

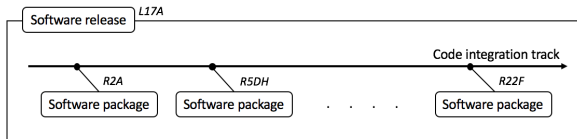


Figure: Several software packages that are launched in the timeline

- Test cases in QA capacity area on signaling capacity - treated as an observation in the dataset

Data is collected on January 20, 2017

Three datasets: Software release L16A, L16B, and L17B

- Sorted by software package version
- Filtered out test cases which are not executed properly
- Selected test case which has *lowest* value of the CPU utilization to represent a performance of a specific software package

In total, each dataset contains 64, 241, and 144 test cases, respectively

## EventsPerSec: Event intensity

- Contains several *local events*
- Stores multiple values separated by a tab character
- Some local events are used as predictor variables
- Implement a function to split each element to columns

## Response variable

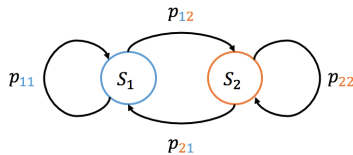
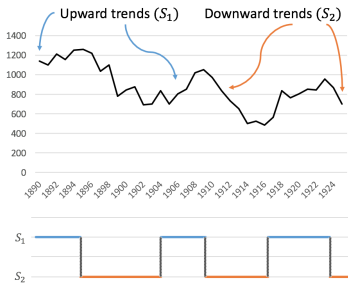
- TotCpu%: CPU utilization

## Predictor variables

- EventsPerSec
  - RrcConnectionSetupComplete
  - Paging
  - X2HandoverRequest
- Test environments
  - DuProdName: Product hardware name
  - Fdd/Tdd: Different standard of LTE 4G Technology
  - NumCells: Number of cells in the base station

## Markov switching model [Hamilton, 1989]

- Describe evolution of the process at different period of time
- Involve multiple structures that can characterize the time series behaviors in different states
- The switching mechanism between the states is assumed to be an unobserved Markov chain





## Markov switching autoregressive model

$$y_t = X_t \beta_{S_t} + \phi_{1,S_t} y_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N(0, \sigma_{S_t}^2)$$

Assuming that  $S_t$  denote an unobservable state variable

$y_t$  is the observed value of time series at time  $t$

$X_t$  are the predictor variables of time series at time  $t$

$\beta_{S_t}$  are the coefficients in state  $S_t$ , where  $S_t = 1, 2, \dots, k$

$\phi_{1,S_t}$  is an autoregression coefficient at time  $t - 1$  in state  $S_t$

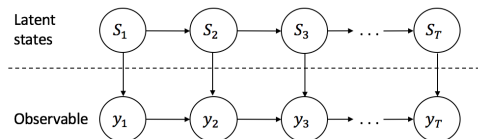


Figure: Model with additional dependencies at observation level

A coefficient of a predictor variable can have either different values in different state or a constant value in all state.

The variable that can take on different values is said to have a *switching effect*.

The variable which have the same coefficient in all states is the variable that does not have a switching effect, or said to have a *non-switching effect*.

When applying the Markov switching model, we need to decide on

- Number of states,  $k$
- Number of switching coefficients in the model

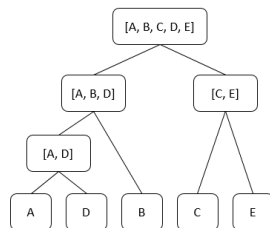
Based on the applied literature, the information criteria called the Bayesian Information Criterion is used to select these numbers

$$\text{BIC} = -2 \ln(L(\hat{\theta})) + m \cdot \ln(T)$$

Reduce overfitting problem by penalizing on the number of parameters in the model

## E-divisive [James, 2016]

- 1 Non-parametric approach: more flexible as no assumption about the distribution is made
- 2 Detects multiple change point locations based on a divisive hierarchical estimation algorithm
- 3 Algorithm: Recursively partition a time series, and perform a permutation test to find the statistical significance of an estimated change point.
- 4 Remark: Obtain a rough idea of the change point location



- State of the CPU utilization is unknown

- State of the CPU utilization is unknown
- Simulated two datasets - Dataset 1 and Dataset 2 - with different switching between states

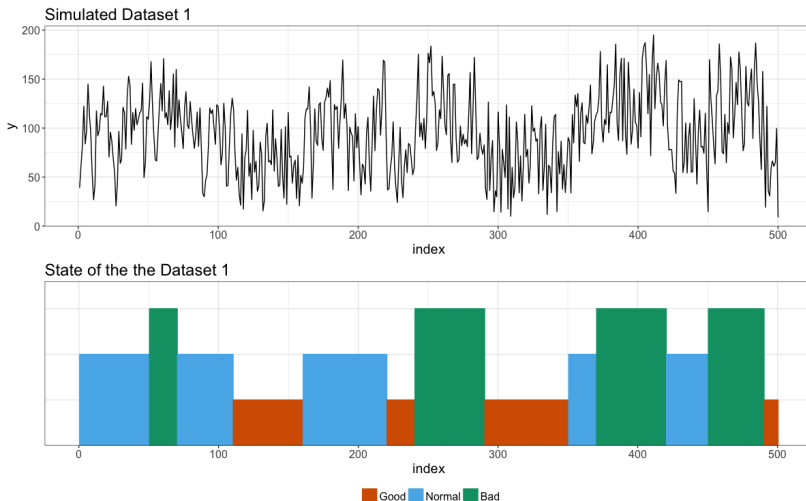
$$y_t = \begin{cases} 10 + 0.6X_{1,t} - 0.9X_{2,t} + 0.5y_{t-1} + \varepsilon_t^{(1)} & \text{Normal} \\ 2 + 0.8X_{1,t} + 0.2y_{t-1} + \varepsilon_t^{(2)} & \text{Bad} \\ -12 + 0.7X_{1,t} + 0.2X_{2,t} - 0.2y_{t-1} + \varepsilon_t^{(3)} & \text{Good} \end{cases}$$

$y_t$  is assumed to be value of a CPU usage

$$x_{1,t} \sim U[50, 200]$$

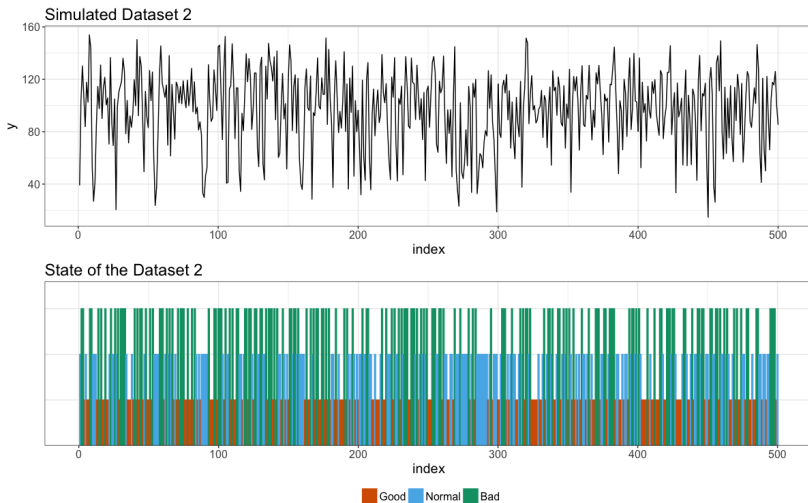
$$x_{2,t} \sim U[0, 50]$$

$$\varepsilon_t^{(1)} \sim N(0, 1), \quad \varepsilon_t^{(2)} \sim N(2, 0.5), \quad \text{and} \quad \varepsilon_t^{(3)} \sim N(1, 1)$$



**Figure:** A simulated data of Dataset 1 and the period in the time series when observation is in each state.

## Simulation study for model evaluation



**Figure:** A simulated data of Dataset 2 and the period in the time series when observation is in each state.



Decide: Number of states

Hypothesis: Markov switching model with *two* or *three* states

---

- BIC is one criteria to select the appropriate model but model output and plot should also be taken into account
- *Three-state* model are chosen for further analysis
- Remark: Higher number of states  $k \geq 4$  are more likely to give worse results and were not considered

Decide: Number of switching coefficients in the model

Hypothesis: test environments is possible to have non-switching effects

---

- Software release L16A:  
*Fdd/Tdd* and *Numcells* are non-switching coefficients
- Software release L16B:  
*DuProdName* is non-switching coefficient
- Software release L17A:  
*DuProdName*, *Fdd/Tdd*, and *NumCells* are non-switching coefficients

## Simulated Dataset 1 and Dataset 2

- E-divisive is less powerful in detecting changes  
→ the method only look at the value of CPU utilization
- Both method detect change at the same location  
→ high probability to be an actual change
- Both method detect change close to one another but not exact location  
→ lower chance be a false alarm

Real data: Software release L16A

- E-divisive cannot detect any changes in the time series data
- No comparison is made

## Real data: Software release L16B

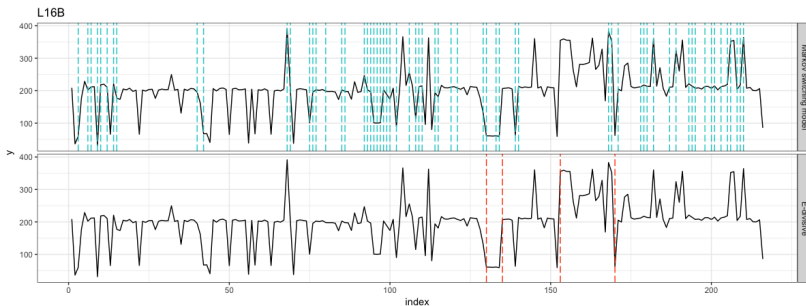


Figure: The estimated change point locations

## Real data: Software release L17A

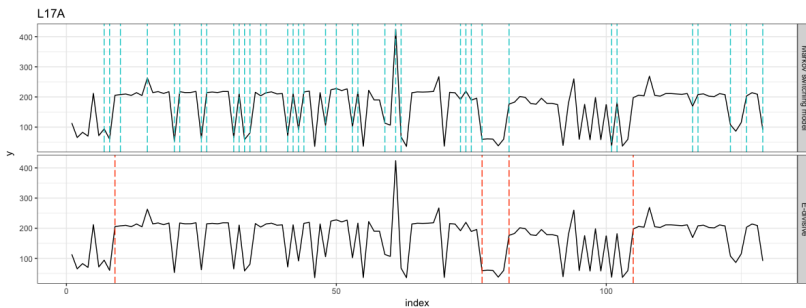
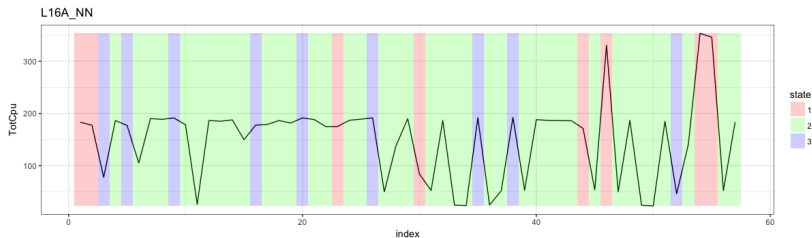


Figure: The estimated change point locations

## Software release L16A

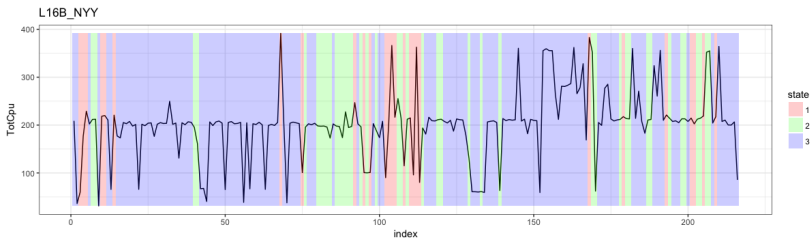
- State 1: Degradation
- State 2: Improvement
- State 3: Steady



**Figure:** The CPU utilization showing the periods where the observation is in the specific state.

## Software release L16B

- State 1: Degradation
- State 2: Improvement
- State 3: Steady

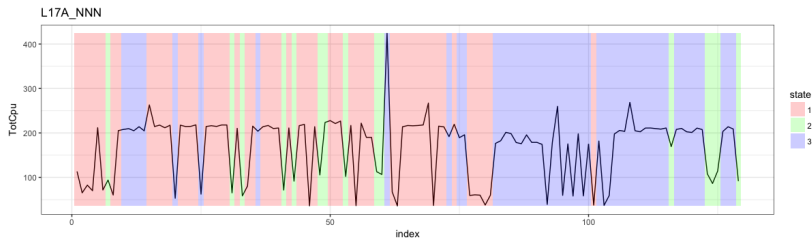


**Figure:** The CPU utilization showing the periods where the observation is in the specific state.



## Software release L17A

- State 1: Degradation
- State 2: Improvement
- State 3: Steady



**Figure:** The CPU utilization showing the periods where the observation is in the specific state.

Effects of test environments -*DuProdName*, *Fdd/Tdd*, and *NumCells* - on the CPU utilization

- Software release L16A:  
*Fdd/Tdd* and *NumCells*
- Software release L16B:  
*DuProdName* and *NumCells*
- Software release L17A:  
*DuProdName*



- Larger dataset
- Effects of interaction terms or other local events
- Consider on the other performance metrics (e.g., memory usage and latency)
- Use semi-supervised learning algorithm (e.g., Self-Organizing Maps) if some test cases are labeled with state



James D Hamilton (1989)

A new approach to the economic analysis of nonstationary time series and the business cycle

*Econometrica: Journal of the Econometric Society*, pages 357-384.



Josep A. Sanchez-Espigares and Alberto Lopez-Moreno (2014)

MSwM: Fitting Markov Switching Models

*CRAN R*.



Nicholas A. James and David S. Matteson (2016)

ecp: Nonparametric Multiple Change Point Analysis of Multivariate Data

*CRAN R*.