

# Applying Machine Learning to LTE/5G Performance Trend Analysis

Araya Eamrurksiri

Linkoping university

June 2, 2017

- Many test cases are executed for testing software packages
- Evaluate the performance of an updated software package by visualizing the graph
- Algorithm that can reduce workload of manual inspection

- Detect the state of the CPU utilization (degradation, improvement, or steady state)
- Detect whether there is any change in the test environment that affects the CPU utilization

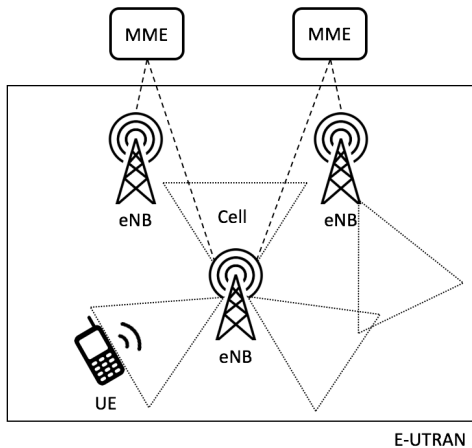
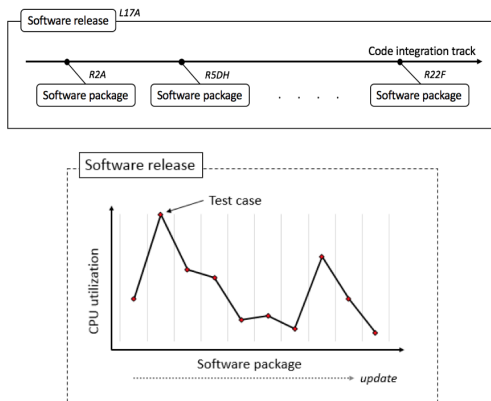


Figure: LTE architecture overview

- Software release
- Software package  
→ a *time point* in the time series data
- Test case  
→ an *observation* in the dataset



**Figure:** Top: Several software packages are launched in the timeline, Bottom: The CPU utilization value in each software package

Three datasets: Software release A, B, and C contain 64, 241, and 144 test cases, respectively

Response variable

- TotCpu%: CPU utilization

Predictor variables

- local events in 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

## Method

- Markov switching model
- E-divisive method
- Tools

## 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



## 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

A coefficient of a predictor variable can have either different values in different state,  $\beta_{S_t}$ , or a constant value in all state,  $\beta$ .

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

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

## 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

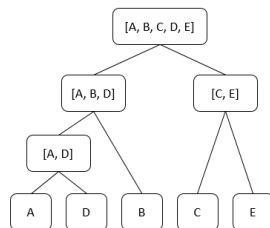


Figure: Divisive hierarchical clustering

## R programming

- Markov switching model is performed using *MSwM* package [Sanchez-Espigares, 2014]  
Various extensions and modifications were made in the package  
For example,
  - Make it more stable to use with categorical variables
  - State prediction function
  - Plot for visualizing the results
- E-divisive method is performed using *ecp* package [James, 2016]

## Results and Discussion

- Markov switching model
  - Analysis I: Number of state
  - Analysis II: Number of switching coefficients
- E-divisive method
- Comparison between both methods
  - Simulated data
  - Real data
- State inference on the results from the Markov switching model

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 (BIC) is used for model selection

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

where,  $m$  is the number of parameters and  $T$  is the number of observations  
BIC attempts to reduce an overfitting problem by penalizing on the number of parameters in the model

## Analysis I: Number of states

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

---

- Model with lower BIC value is preferable
- However, model output along with plot should also be taken into account as well

## Analysis I: Number of states

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

---

- Model with lower BIC value is preferable
- However, model output along with plot should also be taken into account as well
- Two-state model offered less details, and plots were unrealistic and difficult to make an interpretation
- **Three-state** model was chosen for further analysis as it provided more interpretable plots and better fit
- Remark: Higher number of states  $k \geq 4$  are more likely to give worse results and were not considered

## Analysis II: Number of switching coefficients in the model

Hypothesis: Test environments (*DuProdName*, *Fdd/Tdd*, and *Numcells*) is possible to have non-switching effects

---

- *Recall: The variable whose coefficient is constant in all states are said to have a non-switching effect*
- Attempt to reduce the number of parameters to be estimated in the model
- Algorithm used numerical optimization  
→ more estimated parameters will make the obtained result unstable
- Each dataset of the software release was tested with different models
- Three final models for each dataset were obtained



- Apply the E-divisive method to all three datasets of the software release A, B, and C
- Input: the value of the CPU utilization

- In the real data, the state of the CPU utilization is unknown  
→ Evaluation of the model can't be made

- In the real data, the state of the CPU utilization is unknown  
→ Evaluation of the model can't be made
- Simulated two datasets (Dataset 1 and Dataset 2) with the ground truth about the state.

The real models for each state are

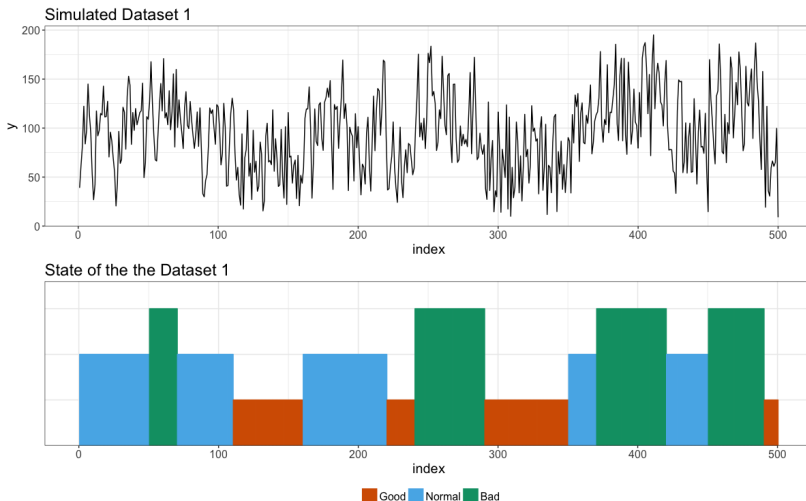
$$y_t = \begin{cases} 10 + 0.6X_{1,t} - 0.9X_{2,t} + 0.5y_{t-1} + \varepsilon_t^{(1)} & \text{Normal} \\ 2 + 0.8X_{1,t} + 0.2y_{t-1} + \varepsilon_t^{(2)} & \text{Bad} \\ -12 + 0.7X_{1,t} + 0.2X_{2,t} - 0.2y_{t-1} + \varepsilon_t^{(3)} & \text{Good} \end{cases}$$

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

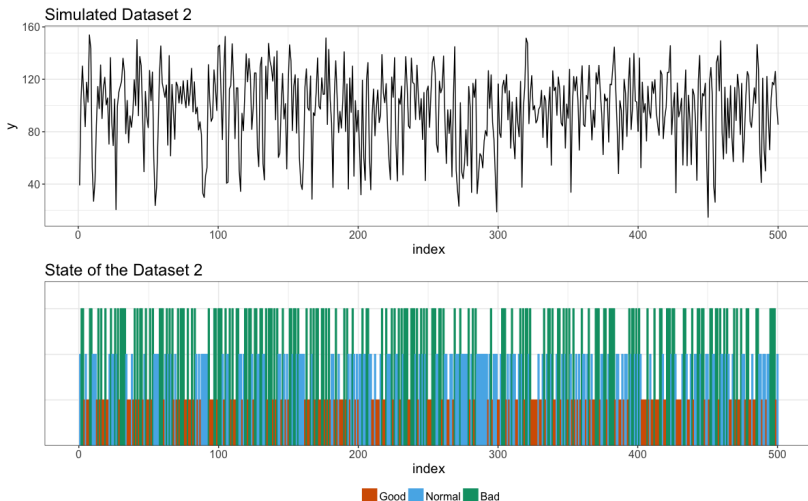
$$x_{1,t} \sim U[50, 200]$$

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

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



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



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

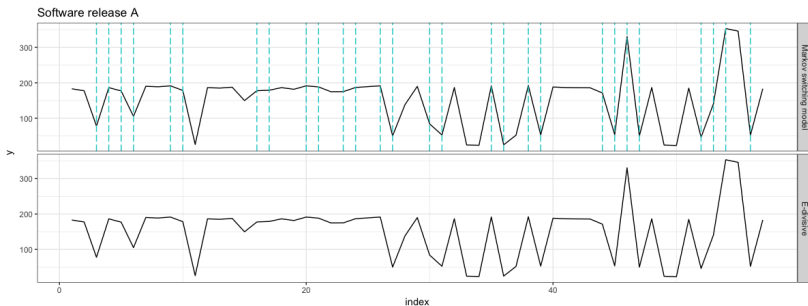
Apply the Markov switching model and the E-divisive method to the simulated Dataset 1 and Dataset 2, it is found that

Apply the Markov switching model and the E-divisive method to the simulated Dataset 1 and Dataset 2, it is found that

- Both methods detect changes at the same location  
→ high probability to be an actual change
- Both methods detect changes close to one another but not at the exact location  
→ lower chance to be a false alarm

## Real data: Software release A

- E-divisive cannot detect any changes in the time series data

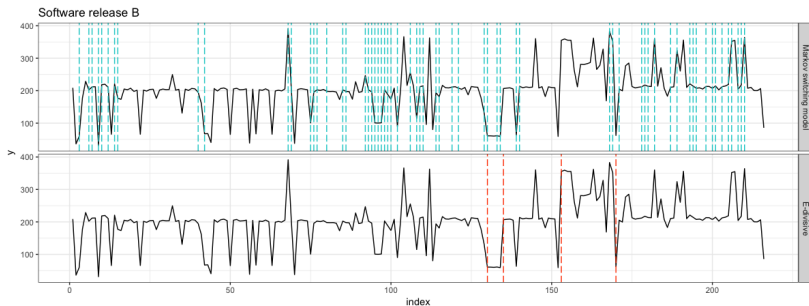


**Figure:** Top: Results from the Markov switching model, Bottom: The change point locations from the E-divisive



## Real data: Software release B

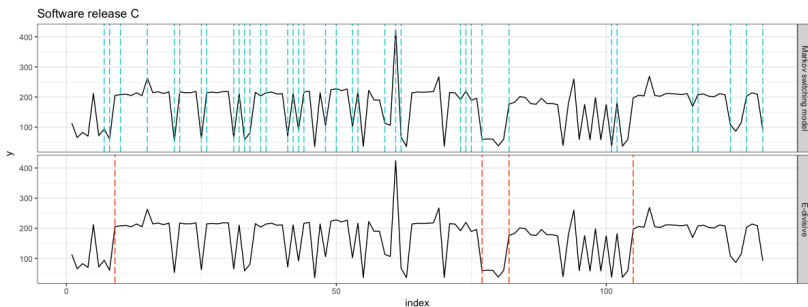
- E-divisive method detects change-points at 130, 135, 153, and 170



**Figure:** Top: Results from the Markov switching model, Bottom: The change point locations from the E-divisive

## Real data: Software release C

- E-divisive method detects change-point at 9, 77, 82, and 105

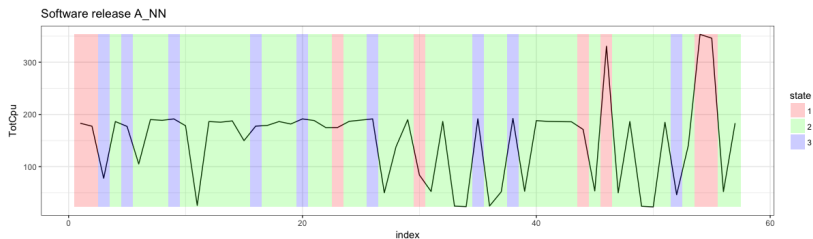


**Figure:** Top: Results from the Markov switching model, Bottom: The change point locations from the E-divisive

## Software release A

## Software release A

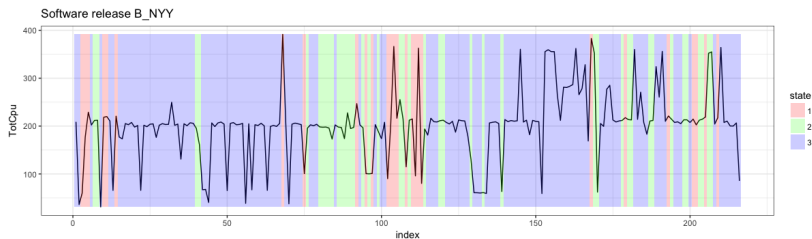
- State 1: Degradation



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

## Software release B

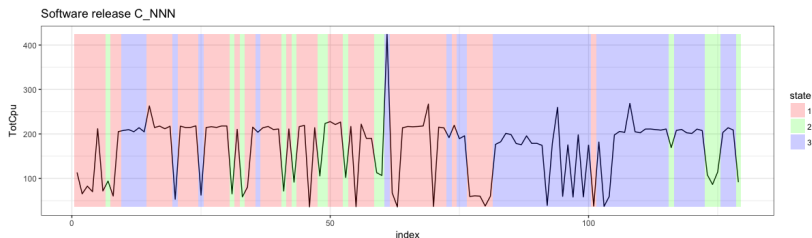
- State 1: Degradation



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

## Software release C

- State 1: Degradation



**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 A:  
*Fdd/Tdd* and *NumCells*
- Software release B:  
*DuProdName* and *NumCells*
- Software release C:  
*DuProdName*

- Markov switching model is able to identify any changes between states rather well, despite some false alarms and missed detections
- E-divisive method is less powerful as it can detect fewer changes and failed to detect many changes  
→ the method only take into account the value of the CPU utilization
- Both methods could be used together to confirm the state change



- Require more extensive data
- Consider on the other performance metrics (e.g., memory usage and latency)
- Apply the Markov switching model to each QA Capacity test case type (i.e., one model for one type of test case)
- Normalize feature set by introducing *weight* parameters
- Use semi-supervised learning algorithm if some test cases are labeled with state  
→ combining clustering-based and classification-based methods



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*.