Partially Hidden Markov Chain Linear AutoRegressive model (PHMC-LAR)

PHMC-LAR is a novel regime switching model dedicated to time series data.
Let $\{X_t\}$ a time series subject to switches in regime. Let $\{Z_t\}$ the corresponding state process which is observed at some random timesteps and hidden at other timesteps. In PHMC-LAR model, $\{Z_t\}$ is a PHMC process and within each regime, X_t is a LAR process:

$$X_t \,|\, X_{t-p}^{t-1}, Z_t = k \;:=\; \phi_{0,k} + \sum_{i=1}^{p} \phi_{i,k} X_{t-i} + h_k\, \epsilon_t \quad \text{for} \quad t = 1, \ldots, T. \tag{1}$$

with p the number of past values of X_t to be used in modelling, k the state at time-step t, $(\phi_{0,k},\ \phi_{1,k}, ..., \phi_{p,k})$ the intercept and autoregressive parameters associated with k^{th} state, h_k the standard deviation associated with k^{th} state and $\{\epsilon_t\}$ the error terms.

Model parameters are estimated by Expectation-Maximization algorithm (EM).

We present three experiments within which PHMC-LAR model was applied to synthetic simulated data and realistic machine condition data.

Related Papers

- Prediction and Inference in a Partially Hidden Markov-switching Framework with Autoregression. Application to Machinery Health Diagnosis. Fatoumata Dama, Christine Sinoquet. University of Nantes, LS2N UMR CNRS 6004. Accepted at the 33rd IEEE International Conference on Tools with Artificial Intelligence(ICTAI): 8 pages, 1-3 november 2021, virtual.
- F. Dama and C. Sinoquet. Partially Hidden Markov Chain Linear Autoregressive model : inference and forecasting. Under revision at Machine Learning Journal (arXiv version).

Required Tools

- numpy
- scipy
- pickle
- concurrent.futures

Learn PHMC-LAR model parameters by EM algortihm

hmc_lar_parameter_learning(X_order, nb_regimes, data, initial_values, states, innovation, init_method="rand", nb_iters=500, epsilon=1e-6, nb_init=10, nb_iters_init=5)

Parameters

- **X_order:** ***positive int value***
  Autoregressive order.

- **nb_regimes:** ***strictly positive int***
  Number of switching regimes.

- **data:** ***list of array_like***
  List of S observed time series where data[s] is a column vector of dimension $T_s \times 1$. And T_s denotes the size of the s^{th} time series starting at timestep t = X_order + 1.
- **initial_values:** ***list of array_like***
  List of S initial values where initial_values[s] is a column vector of dimension X_order×1. initial_values[s] are initial values associated with the s^{th} time series.

- **states:** ***list of array_like***
  List of S state sequences where states[s] is a line vector of dimension $1 \times T_s$:
  - if states[s][0,t] = -1 then $Z_t^{(s)}$ is latent.
  - otherwise states[s][0,t] is in $\{0, 1, 2, ..., \text{nb\_regimes} - 1\}$ and $Z_t^{(s)}$ is observed to be equal to states[s][0,t].

- **innovation:** ***string***
  Law of error terms, only *'gaussian'* noises are supported.

- **nb_iters:** ***int value***
  Maximum number of EM iterations.

- **epsilon:** ***real value***
  Convergence precision. EM will stops when the shift in parameters' estimate between two consecutive iterations is less than epsilon. L1-norm was used.

Returns (log_ll, A, Pi, list_Gamma, list_Alpha, ar_coefficients, sigma, intercept, psi)

- **log_ll:** r***eal value***
  loglikelihood
- **A:** ***array_like***
  Transition matrix.
- **Pi:** ***array_like***
  Initial state probabilities.
- **list_Gamma:** ***list of array_like***
  Smoothed marginal probabilities.
- **list_Alpha:** ***list array_like***
  Filtered marginal probabilities.
- **ar_coefficients:** ***array_like***
  Autoregressive coefficients of dimension X_order×nb_regimes. ar_coefficients[i,k] is the coefficient associated with X_{t-i} within regime k.
- **sigma:** ***array_like***
  Standard deviation, nb_regimes length array.
- **intercept:** ***array_like***
  Intercept, nb_regimes length array.
- **psi:** ***dict***
  Initial law parameters having the following entries:
  - mean: X_order length array.
  - covar: X_order x X_order matrix.
    The initial values follow a gaussian multivariate law.

Example

```
import os
import sys
sys.path.append(os.path.abspath(os.path.join(os.path.dirname(__file__), "src")))
import numpy as np
from EM_learning import hmc_lar_parameter_learning

#-----hyperparameters
X_order = 2
nb_regimes = 4
innovation = "gaussian"

#-----training data
data = []
initial_values = []
states = []

d = np.genfromtxt('data/recessions_RGNP.csv', delimiter=',')
#initial values
initial_values.append(np.zeros(shape=(X_order,1), dtype=np.float64))
initial_values[0][:, 0] = d[0:X_order]
#observed time series
data.append(np.ones(shape=(d.shape[0]-X_order, 1), dtype=np.float64))
data[0][:, 0] = d[X_order:]
#state sequences, completely latent
states.append(-1 * np.ones(shape=(1, data[0].shape[0]), dtype=np.int64))

#-----model learning
(total_log_ll, A, Pi, list_Gamma, list_Alpha, ar_coefficients, sigma, intercept, psi) = \
        hmc_lar_parameter_learning (X_order, nb_regimes, data, initial_values, states, innovation)
```

Experiment on synthetetic data

Data description

Binary files data-set_txxx_T=yyyy_N=zzz contain data simulated from PHMC-LAR(p=2, K=4) defines as follows:

$$A = \begin{pmatrix} 0.5 & 0.2 & 0.1 & 0.2 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.5 & 0.2 \\ 0.2 & 0.1 & 0.2 & 0.5 \end{pmatrix}, \quad \pi = (0.25, 0.25, 0.25, 0.25).$$

$\theta^{(X,k)} = (\phi_{0,k}, \phi_{1,k}, \phi_{2,k}, h_k)$:
$\theta^{(X,1)} = (2,\, 0.5,\, 0.75, 0.2), \quad \theta^{(X,2)} = (-2, -0.5,\, 0.75,\, 0.5), \quad \theta^{(X,3)} = (4,\, 0.5, -0.75,\, 0.7), \quad \theta^{(X,4)} = (-4, -0.5, -0.75,\, 0.9)$. $\mathbf{x}_0 \sim \mathcal{N}_2 \left((3,5), \begin{pmatrix} 1 & 0.1 \\ 0.1 & 1 \end{pmatrix} \right), \quad \epsilon_t \sim \mathcal{N}(0,1).$

Theses files were created by the *dump* function of pickle module. Data can be unserialized using the "load" function from the same module.
After unserialization operation, we get the following data regarding the file under consideration:

1) For files having suffix “test” we get variables (data_S, data_X, init_values).

2) For files having suffix “train” we get variables (data_S, data_X, init_values, data_S_for, data_X_for).

The previous variables are defined as bellow:

- data_S list of N state sequences of length T.
- data_X list of N time series of length T.
- init_values list of N initial values of length 2, associated to the previous time series.
- data_S_for list of N state sequences of length 30.
- data_X_for list of N time series of length 30.

Note that, data_S_for and data_X_for are respectively 30-steps ahead out-of-sample forecast states and values. They are used in forecating experiments.

Run experiments

```
python3 -O synthetic_data_experiment_1.py train_data_file output_dir P
python3 -O synthetic_data_experiment_2.py train_data_file output_dir rho
```

with

- **data_file** Training data file, equals data-set_train_T=100_N=X for X=1, 10, 100.
- **output_dir** The name of the output directory.
- **P** $\in [0, 100]$ Percentage of labelled observation within training set.
- **rho** $\in [0, 100[$ The level of unreliability upon labels.

The estimated model is serialized within a binary file and saved in output_dir.
File *synthetic_data_inference.py* and *synthetic_data_forecasting.py* contains functions that allow to perform inference and forecasting tasks.

Experiment on realistic machine condition data

Data description

Directories train_FD00X for X=1,2,3,4 contain the health indicators [1,2] and the ground truth segementations [3] computed from the corresponding original CMAPSS datasets [4].
Files train_FD00X_LHI_state are binary files in which health indicators and ground truth segementations was serialized by pickle module.
Script cmapss_experiment.py takes one of these files as its first argument.

[1] E. Ramasso. Investigating computational geometry for failure prognostics. Int. J. Progn. Health Manag., vol. 5, no. 1, p. 005, 2014.
[2] https://www.mathworks.com/matlabcentral/fileexchange/54808-segmentation-of-cmapss-trajectories-into-states.
[3] E. Ramasso, Segmentation of CMAPSS health indicators into discrete states for sequence-based classification and prediction purposes,” tech.rep., 2016.
[4] https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan.

Run experiment

```
python3 -O cmapss-experiment.py data_file output_dir P D
```

with

- **data_file** Equals data/cmapss-data/train_FD00X_HI_state for X=1,2,3,4.
- **output_dir** The name of the output directory.
- **P** Equals 0 or 1. 0 means unsupervised scheme (MSAR model) and 1 means semi-supervised schem (PHMC-LAR model).
- **D** The autoregressive order.

The estimated model is serialized within a binary file and saved in *output_dir*.
File *cmapss_inference.py* contains functions that allow to perform inference task.