DL for change point detection

Artem Ryzhikov[1]

[1]HSE University

April 19, 2021

LAMBDA • HSE

Motivation

Motivation

How to apply deep learning for change point detection (CPD)?

KLCPD

Problem statement

Let $\mathbb{P}$ - distribution of normal time stamps, $\mathbb{Q}$ - distribution of change points.
Given kernel k, the Maximum Mean Discrepancy (MMD) distance looks following:

$$M_k(\mathbb{P}, \mathbb{Q}) = ||\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}||^2_{\mathcal{H}_k} = \mathbb{E}_{\mathbb{P}} k(x, x') - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}} k(x, y) + \mathbb{E}_{\mathbb{Q}} k(y, y')$$

Advantages of MMD w.t. KL:

- no need in pdf
- can be empirically estimated: $\hat{M}_k(X, Y) = \frac{1}{C_m^2}\Sigma_{i \neq j} k(x_i, x_j) - \frac{2}{m^2}\Sigma_{i,j} k(x_i, y_j) + \frac{1}{C_m^2}\Sigma_{i \neq j} k(y_i, y_j)$
- can be used to test null hypothesis $\mathbb{P} = \mathbb{Q}$: $\frac{\hat{M}_k(X,Y) - M_k(\mathbb{P},\mathbb{Q})}{\sqrt{V_m(\mathbb{P},\mathbb{Q})}} \xrightarrow{D} \mathcal{N}(0, 1)$.
  The test power is then $P(m\hat{M}_k(X, Y) > c_\alpha) \to \Phi(\frac{M_k(\mathbb{P},\mathbb{Q})}{\sqrt{V_m(\mathbb{P},\mathbb{Q})}} - \frac{c_\alpha}{m\sqrt{V_m(\mathbb{P},\mathbb{Q})}})$, where Φ is CDF of $\mathcal{N}(0, 1)$

KLCPD. Idea 1

$P(m\hat{M}_k(X, Y) > c_\alpha) \to \Phi(\frac{M_k(\mathbb{P},\mathbb{Q})}{\sqrt{V_m(\mathbb{P},\mathbb{Q})}} - \frac{c_\alpha}{m\sqrt{V_m(\mathbb{P},\mathbb{Q})}})$

Idea: use $\hat{M}_k(X, Y)$ as score

Problem: too small samples in $Y \sim \mathbb{Q}$

KLCPD. Idea 2

Problem: too small samples in $\mathbb{Q}$

Solution: use generative model $\mathbb{G}$ for change points (anomalies)

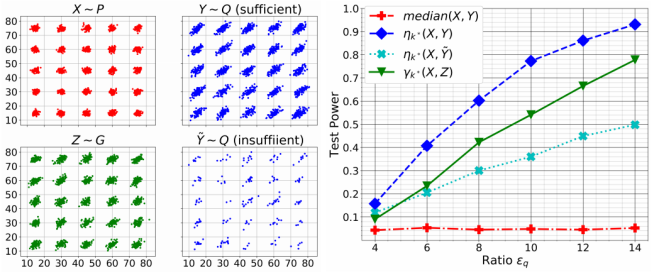


Figure 2: **Left**: 5×5 Gaussian grid, samples from $\mathbb{P}$, $\mathbb{Q}$ and $\mathbb{G}$. We discuss two cases of $\mathbb{Q}$, one of sufficient samples, the other of insufficient samples. **Right**: Test power of kernel selection versus ϵ_q. Choosing kernels by $\gamma_{k^*}(X, Z)$ using a surrogate distribution $\mathbb{G}$ is advantageous when we do not have sufficient samples from $\mathbb{Q}$, which is typically the case in time series CPD task.

Upper estimated for surrogate model $\mathbb{G}$

We will try to fit $\mathbb{G}$ s.t.

$$M_k(\mathbb{P}, \mathbb{P}) < M_k(\mathbb{P}, \mathbb{G}) < M_k(\mathbb{P}, \mathbb{Q}), \forall k \in \mathcal{K}$$

It leads to stronger test power:

$$\max_{k \in \mathcal{K}} \frac{M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} - \frac{c_\alpha / m}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} \geq \max_{k \in \mathcal{K}} \frac{M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{v_u / m}} - \frac{c_\alpha}{\sqrt{m v_l}} \geq \max_{k \in \mathcal{K}} \frac{M_k(\mathbb{P}, \mathbb{G})}{\sqrt{v_u / m}} - \frac{c_\alpha}{\sqrt{m v_l}} = \gamma_{k*}(\mathbb{P}, \mathbb{G})$$

Kernel and final algorithm

Kernel:

$$K(x, x') = exp(-||f_\phi(x) - f_\phi(x')||^2)$$

Final loss:

$$\min_\theta \max_\phi \quad M_{f_\phi}(\mathbb{P}, \mathbb{G}_\theta) - \lambda \cdot \hat{M}_{f_\phi}(X, X') - \beta \cdot \mathbb{E}_{\nu \in \mathbb{P} \cup \mathbb{G}_\theta} \|\nu - F_\psi(f_\phi(\nu))\|_2^2$$

Implementation

Unofficial PyTorch implementation of KL-CPD

License BSD 3-Clause

KL-CPD is an algorithm for change point and anomaly detection in time series.

More information can be found in the 2019 paper *Kernel Change-point Detection with Auxiliary Deep Generative Models*.

Usage

```
from klcpd import KL_CPD

dim, seq_length = 1, 100
ts = np.random.randn(seq_length,dim)
device = torch.device('cuda')
model = KL_CPD(dim).to(device)
model.fit(ts)
preds = model.predict(ts)
print(preds)
```

Installation

```
pip install git+https://github.com/HolyBayes/klcpd
```

TIRE

TIRE

Idea: use autoencoders latent representations to detect abnormal change points

Tricks

Pre processing

- train AE on time slices of size N
- use both FFT and original timeseries

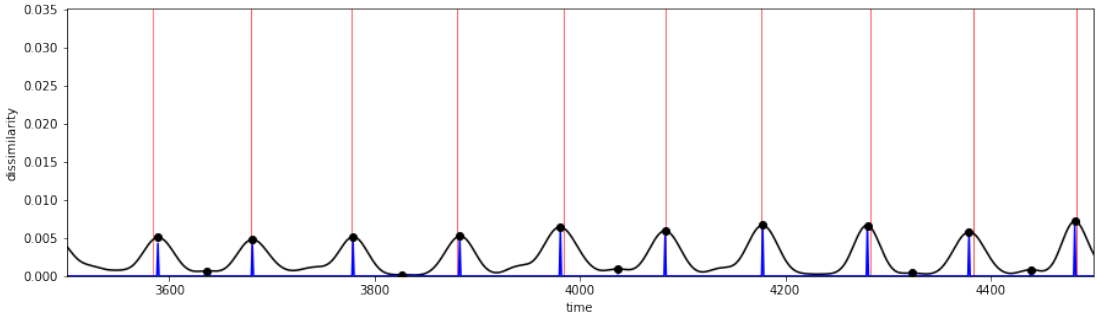
Inference

- split AE latent code z on *time-invariant* (s_t) and *instantaneous* (u_t) parts
- minimize $\left(||y_t - \tilde{y}_t||^2 + \frac{\lambda}{K}\Sigma_{k=0}^{K}||s_{t-K} - s_{t-K-1}||^2\right)$

Post processing

- use moving average smoothing $\tilde{s}_t$ of latent codes s_t
- use $D_t = ||\tilde{s}_t - \tilde{s}_{t+N}||^2$ as change point score
- exclude local peaks by $\hat{D}_t = D_t - max\{\min_{t_L<t^*<t} D_{t^*}, \min_{t_R>t^*>t} D_{t^*}\}$

Example


0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
dissimilarity
3600
3800
4000
4200
4400
time

Implementation

PyTorch implementation of TIRE (UNOFFICIAL)

License BSD 3-Clause

TIRE is an autoencoder-based change point detection algorithm for time series data that uses a TIme-Invariant Representation (TIRE). More information can be found in the 2020 preprint *Change Point Detection in Time Series Data using Autoencoders with a Time-Invariant Representation*.

Usage

```
from TIRE import DenseTIRE as TIRE
import torch
import numpy as np

seq_length = 4500
dim = 1
ts = np.random.randn(seq_length, dim)

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = TIRE(dim).to(device)

model.fit(ts, epoches=1)

dissimilarities, change_point_scores = model.predict(ts)

# plt.plot(dissimilarities)
```

Results

Results

TABLE II
COMPARISON OF THE AUC OF THE PROPOSED TIME-INVARIANT REPRESENTATION CPD METHODS (TIRE) WITH BASELINE METHODS.

	Mean	**Variance**	**Coefficient**	**Gaussian**	**Bee dance**	**HASC-2011**	**Well log**	**Average**
GLR [14], [15]	0.73 ± 0.02	0.81 ± 0.02	$\mathbf{1.00} \pm 0.01$	$\mathbf{0.989} \pm 0.004$	0.55 ± 0.06	0.6431	0.2109	0.71 ± 0.01
RuLSIF [22]	0.708 ± 0.008	0.65 ± 0.02	0.36 ± 0.02	0.874 ± 0.007	0.47 ± 0.06	0.3162	0.798	0.597 ± 0.009
KL-CPD [20]	0.872 ± 0.007	0.23 ± 0.02	0.11 ± 0.01	0.84 ± 0.07	0.56 ± 0.07	0.343	0.4247	0.48 ± 0.01
ABD [27]	0.22 ± 0.02	0.17 ± 0.02	0.08 ± 0.02	0.18 ± 0.02	0.20 ± 0.04	0.2487	0.477	0.224 ± 0.008
TIRE-TD-*a*	0.86 ± 0.01	0.25 ± 0.01	0.26 ± 0.01	0.958 ± 0.009	0.36 ± 0.05	0.4166	0.8002	0.558 ± 0.007
TIRE-FD-*a*	0.86 ± 0.01	$\mathbf{0.85} \pm 0.01$	0.96 ± 0.01	0.83 ± 0.04	$\mathbf{0.70} \pm 0.10$	**0.6504**	0.6278	$\mathbf{0.78} \pm 0.02$
TIRE-*a*	0.86 ± 0.01	$\mathbf{0.85} \pm 0.01$	0.74 ± 0.05	0.92 ± 0.02	0.65 ± 0.09	0.6172	0.7656	0.77 ± 0.02
TIRE-TD-*b*	$\mathbf{0.882} \pm 0.009$	0.26 ± 0.02	0.26 ± 0.02	0.965 ± 0.006	0.42 ± 0.06	0.4284	**0.8151**	0.58 ± 0.01
TIRE-FD-*b*	0.86 ± 0.01	0.84 ± 0.02	0.95 ± 0.02	0.74 ± 0.03	0.69 ± 0.10	0.6261	0.200	0.70 ± 0.02
TIRE-*b*	0.877 ± 0.009	0.83 ± 0.02	0.76 ± 0.05	0.89 ± 0.02	0.60 ± 0.09	0.6258	0.8134	0.77 ± 0.01

Results of *"Change Point Detection in Time Series Data using Autoencoders with a Time-Invariant Representation"* paper

Bibliography

Links

- KLCPD - Chang, Wei-Cheng et.al., Kernel Change-point Detection with Auxiliary Deep Generative Models, *ICLR 2019*
- KLCPD implementation - Artem Ryzhikov, https://github.com/HolyBayes/klcpd
- TIRE - Tim, De Ryck et.al., Change Point Detection in Time Series Data using Autoencoders with a Time-Invariant Representation
- TIRE implementation - Artem Ryzhikov, https://github.com/HolyBayes/TIRE_pytorch

Thank you for your attention!

Artem Ryzhikov aryzhikov@hse.ru