

$y_1, y_2, \dots, y_\tau$

1.  $K = 10, L = 4 \rightarrow$  generate patterns  $A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 10 & 10 & 10 & 10 \end{pmatrix}$

$\omega(y_i) = \{0, 1\}$ , where 1 stands for local extremum and 0 for other parts.

2.  $\forall y_i, \omega(y_i) = 1 : \text{sample } z \text{ vectors}$

$$\forall \alpha \in A : Z^\alpha = \{z_i^\alpha \mid \omega(y_i) = 1\}$$

3. Cluster  $Z^\alpha \rightarrow c_1^\alpha, c_2^\alpha, \dots, c_m^\alpha$

$$\Xi^\alpha = \{\xi_i^\alpha \mid \xi_i^\alpha = \text{centroid}(c_i^\alpha)\}$$

Inference Stage:

$i = \tau + 1 :$

1.  $\forall \alpha : z_i^\alpha$

2.  $n = |\cup_\alpha \{\xi_i^\alpha \mid |\xi_i^\alpha - z_i^\alpha| < \varepsilon\}|$

3.  $r = \frac{n}{\cup_\alpha \Xi^\alpha}$

4. If  $r > r_{\text{crit}}$  :  $\omega(y_i) = 1$

If  $r < r_{\text{crit}}$  :  $\omega(y_i) = 0$

**How to choose  $r_{\text{crit}}$ ?**

Validation dataset:

$$\forall y_i \in \text{Validation} \wedge \omega(y_i) = 1 : r_i$$

$$\forall y_i \in \text{Validation} \wedge \omega(y_i) = 0 : r_i$$

Choose  $r_{\text{crit}}$  based on how “risky” you are. Plot the probability distributions (assumingly  $r$  value against some kind of probability?)

**Predictive clustering for forecasting**

Local normalization:

Given  $z_i^\alpha \in$  Train Data, take

$$\frac{z_i^\alpha - \min(\tilde{z}_i^\alpha)}{\max(\tilde{z}_i^\alpha) - \min(\tilde{z}_i^\alpha)}$$

And somehow use this with testing data.

## Updating the algorithm

$y_1, y_2, \dots, y_\tau$

1.  $K = 10, L = 4 \rightarrow$  generate patterns  $A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 10 & 10 & 10 & 10 \end{pmatrix}$

$\omega(y_i) = \{0, 1\}$ , where 1 stands for local extremum and 0 for other parts.

2.  $\forall y_i, \omega(y_i) = 1 : \text{sample } z \text{ vectors}$

$$\forall \alpha \in A : Z^\alpha = \{z_i^\alpha \mid \omega(y_i) = 1\}, \quad \forall z_i^\alpha : \frac{z^{\alpha^i} - \min(z_i^\alpha)}{\max(z_i^\alpha) - \min(z_i^\alpha)}$$

3. Cluster  $Z^\alpha \rightarrow c_1^\alpha, c_2^\alpha, \dots, c_m^\alpha$

$$\begin{aligned} \Xi^{\alpha,0} &= \{\xi_i^\alpha \mid \xi_i^\alpha = \text{centroid}(c_i^\alpha)\}, \\ \Xi^{\alpha,1}, \dots \end{aligned}$$

Inference Stage:

$i = \tau + 1 :$

1.  $\forall \alpha : z_i^\alpha$

2.  $n_i = |\cup_\alpha \{\xi_i^\alpha \mid |\xi_i^\alpha - z_i^\alpha| < \varepsilon, \xi_i^\alpha \in \Xi^{\alpha,i}\}|, i = \overline{0, 1, \dots}$

3.  $r_i = \frac{n_i}{\cup_\alpha \Xi^\alpha}, i = \overline{0, 1, \dots}$

4. If  $r > r_{\text{crit}} : \omega(y_i) = 1$

If  $r < r_{\text{crit}} : \omega(y_i) = 0$

Take

$$X_1 = \left[ y_1^{(1)}, y_2^{(1)}, \dots, y_{\tau_1}^{(\tau_1)^{(1)}} \right], l_1 = 0,$$

$$X_2 = \left[ y_1^{(2)}, y_2^{(2)}, \dots, y_{\tau_2}^{(2)} \right], l_2 = 1,$$

...

$$X_k = \left[ y_1^{(k)}, y_2^{(k)}, \dots, y_{\tau_k}^{((k))^{\tau_k}} \right], l_k = |0|$$

1.  $\forall X_i, l_i = 1$  : sample and cluster  $z$  vectors – extract clusters which are common for the given class.
2.  $\forall X_i, l_i = 0$  : sample and cluster  $z$  vectors, then extract clusters common for class 0.

Inference stage: obtain clusters for new time series, count neighbours among zeros and ones and choose clusters with more neighbours.

$$z_i \xrightarrow[\text{norm}]{\text{norm}} z_i^{\text{normed}} \xrightarrow[\text{fit polynomials}]{\text{fit polynomials}} [\alpha_1, \beta_1, \alpha_2, \beta_2, \dots]$$

Here the last term is a vector of coefficients.

## Neural networks for time series

### Form dataset

- Use time windows:

$$X = [y_{t-n}, y_{t-n+1}, \dots, y_{t-1}]$$

$$y = [y_t] \text{ -- one-step ahead prediction}$$

$$y = [y_t, y_{t+1}, \dots, y_{t+h-1}] \text{ -- h-step ahead prediction}$$

- You can add other features

## Neural network architectures

### 1. FNN

Take  $X$  as input of size  $n$  and make the output layer the size of number of steps ahead you want to predict.

Problem: these models do not account for time dependence.

## 2. RNN

Problem: RNNs do not have long-term memory, as earlier outputs decay and may not be accounted for in the end.

## 3. LSTM

<Import pic>

## 4. CNN

Idea: let us use 1D filters to extract local patterns (motives). Use dilation or increase lookback period.

Example:

1. kernel = 3,  $d = 1$  :  $[y_{t-2}, y_{t-1}, y_t]$
2. kernel = 3,  $d = 2$  :  $[y_{t-4}, y_{t-2}, y_t]$

But here you can sometimes use data from the future thus breaking causality. To deal with this problem TCNN was developed.

## 5. TCNN

Idea: Combine properties of CNN and LSTM to make thing good.

- retain causality (solved by causal convolutions – left padding with size  $k - 1$  is used when applying filters instead of values to the right of one that the filter is applied to);
- enable parallelization;
- long effective memory (solved by using all sorts of different dilations).

Temporal (TCNN) block.

Input → Causal dialated convolution → Normalization → Activation function → Backprop → Causal dialated convolution → Normalization → Activation function → Output

Deep TCNN: skip connections.

TCNN Block gets some data as input and summed with it. Note that 1D convolution is used to adjust the sizes.

## 6. Transformers

...

## 7. VAE

VAE (*variational autoencoder*) architecture:

1. Encoder: Input → CNN/RNN → Flatten → FC →  $\mu, \log(\sigma^2)$ , a.k.a. latent distribution of data
2. Sampling:  $z = \mu + \sigma \cdot \varepsilon, \varepsilon \sim \mathcal{N}(0, 1)$
3. Decoder:  $z \rightarrow \text{FC} \rightarrow \text{Reshape} \rightarrow \text{CNN/RNN} \rightarrow \text{output}$
4. ELBO loss (maximization task)

$$\text{ELBO} = \mathbb{E}[\log p(x \mid z)] - K \cdot L(q(z \mid x) \| p(z)) \rightarrow \max$$

## Chaotic time series analysis

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(\rho - z) - y, \\ \dot{z} = xy - \beta z. \end{cases}$$

$x(t), y(t), z(t)$  — time series. If we are able to construct and solve this kind of equations, we will be able to get their values for any point in time.