Topological Attention for Time Series Forecasting

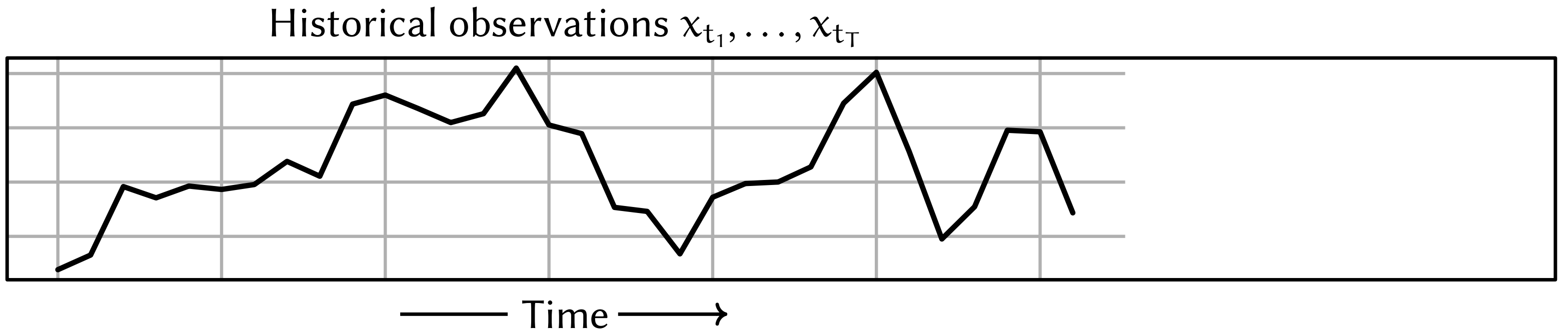
Sebastian Zeng[†] Florian Graf[†] Christoph Hofer[†] Roland Kwitt[†]

[†]University of Salzburg

NeurIPS 2021

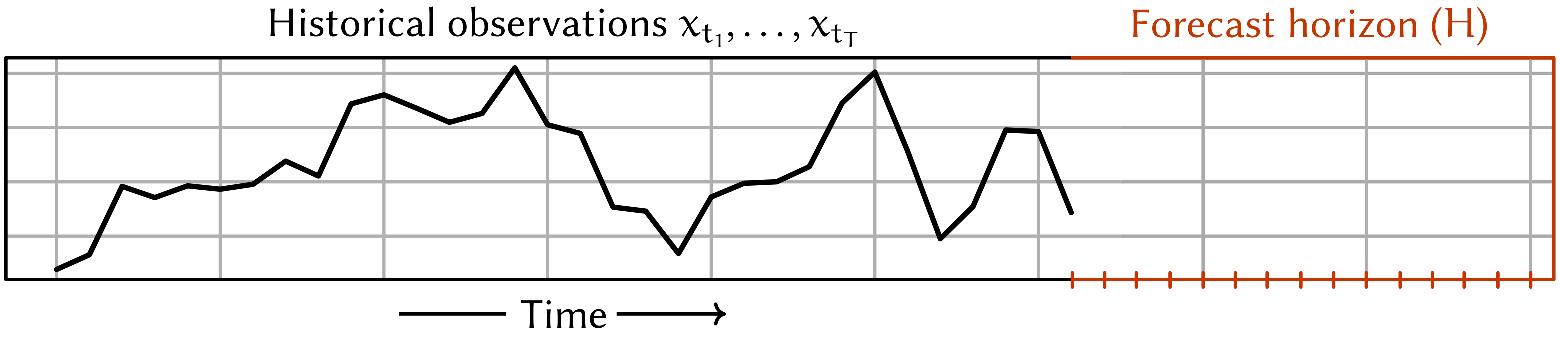
Problem statement

We consider the problem of point-forecasting of **univariate** time series.



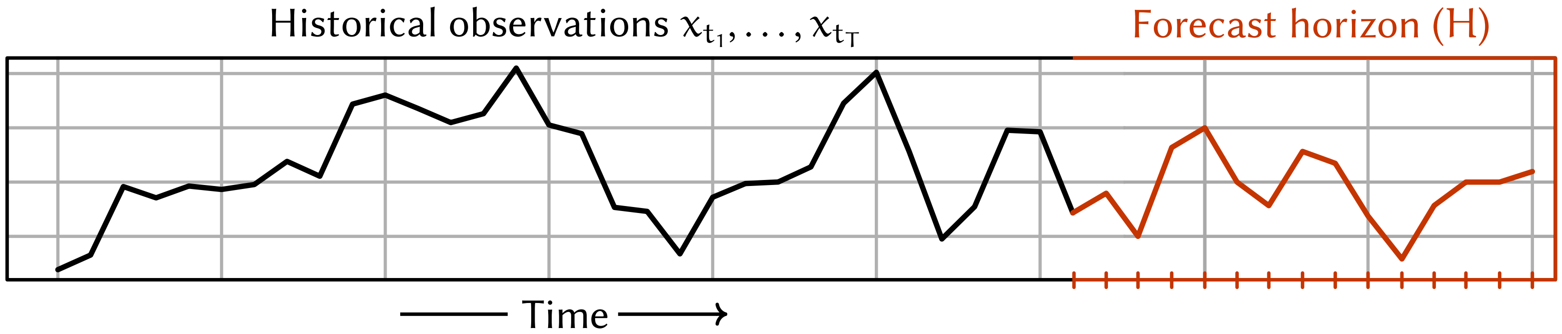
Problem statement

We consider the problem of point-forecasting of **univariate** time series.



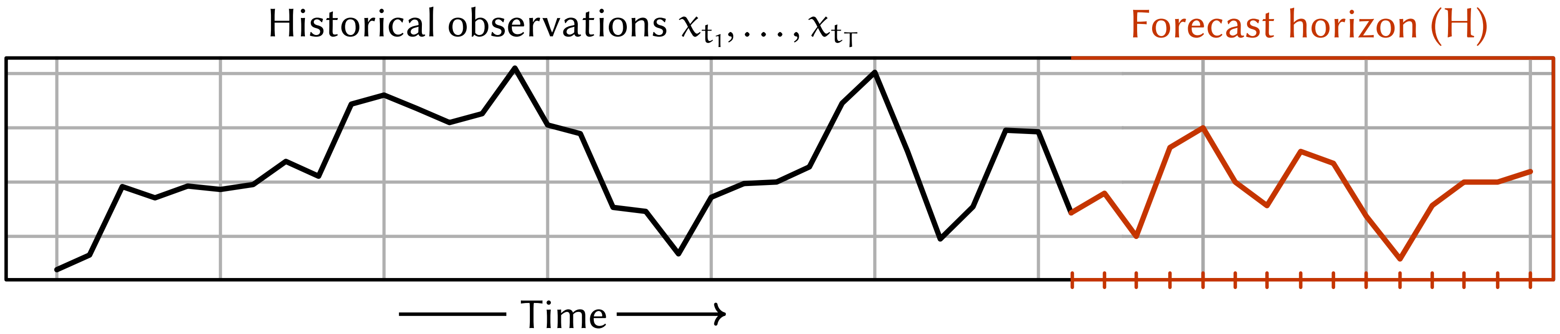
Problem statement

We consider the problem of point-forecasting of **univariate** time series.



Problem statement

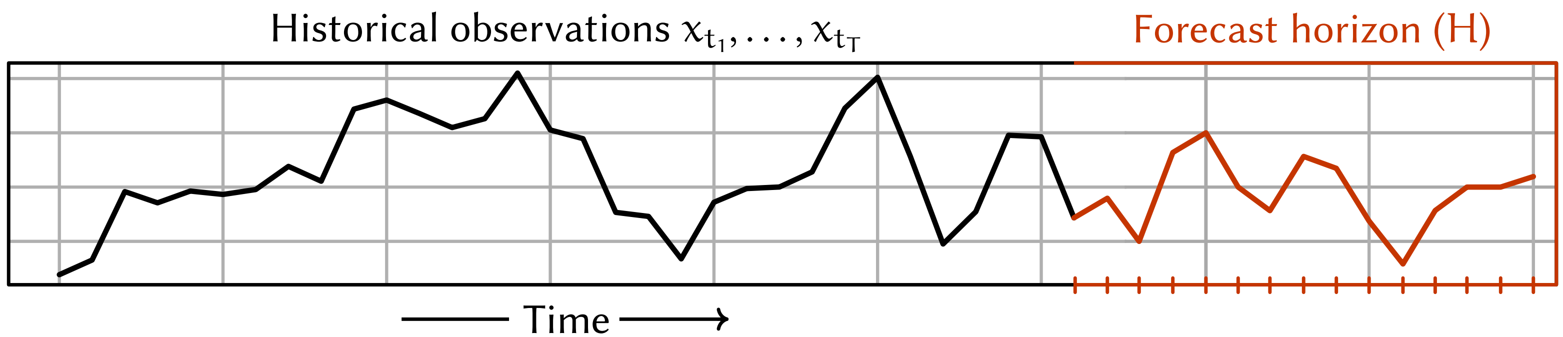
We consider the problem of point-forecasting of **univariate** time series.



We assume that a corpus of N time series is available.

Problem statement

We consider the problem of point-forecasting of **univariate** time series.



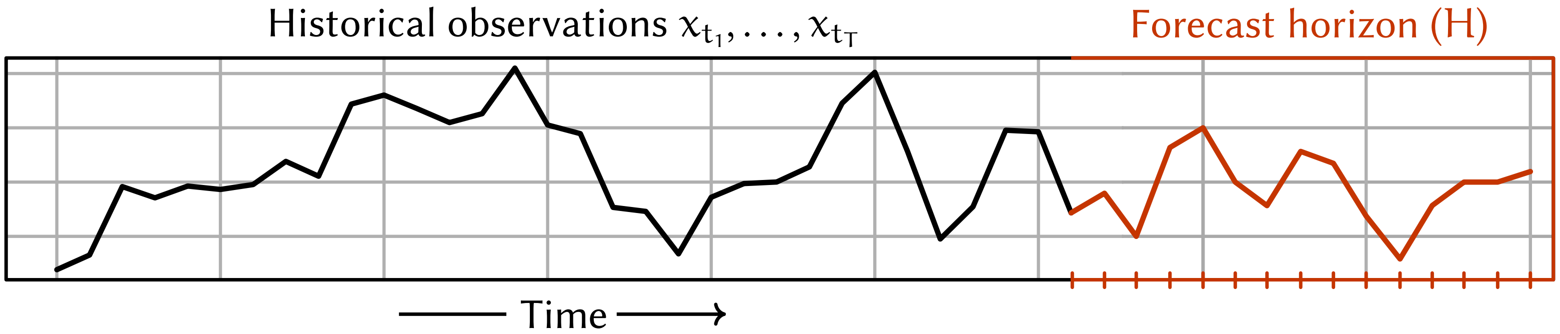
We assume that a corpus of N time series is available.

Typically, statistical/learning methods operate on **raw observations**, i.e.,

$$x_{t_1}, \ldots, x_{t_T} \longrightarrow \boxed{\text{Model}_{\theta}} \longrightarrow \hat{x}_{t_{T+1}}, \ldots, \hat{x}_{t_{T+H}}$$

Problem statement

We consider the problem of point-forecasting of **univariate** time series.



We assume that a corpus of N time series is available.

Typically, statistical/learning methods operate on **raw observations**, i.e.,

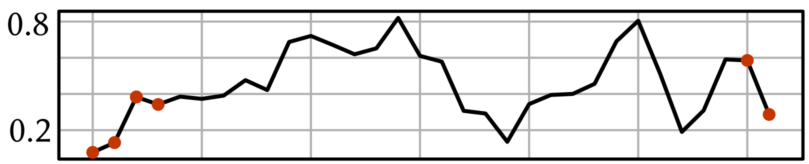
$$x_{t_1}, \ldots, x_{t_T} \longrightarrow \boxed{\text{Model}_{\theta}} \longrightarrow \hat{x}_{t_{T+1}}, \ldots, \hat{x}_{t_{T+H}}$$

Q: Can we leverage **topological** information?

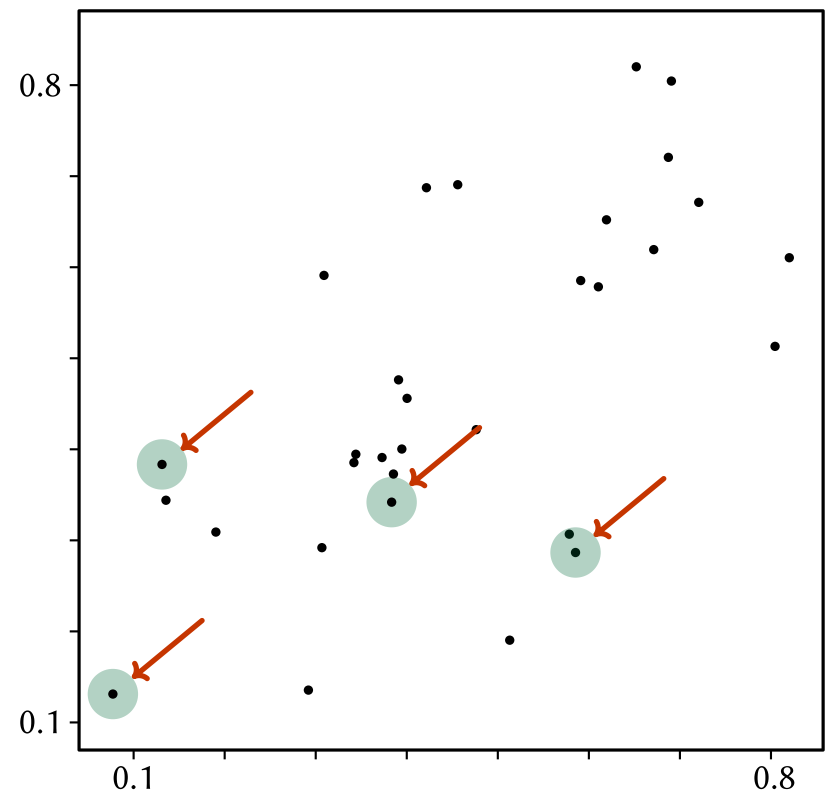
Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ ...

[de Silva et al., 2012; Perea & Harer, 2015]



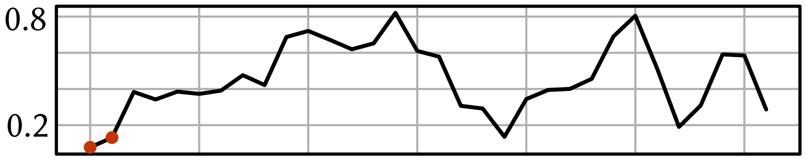
Example of an embedding in $\mathbb{R}^2$



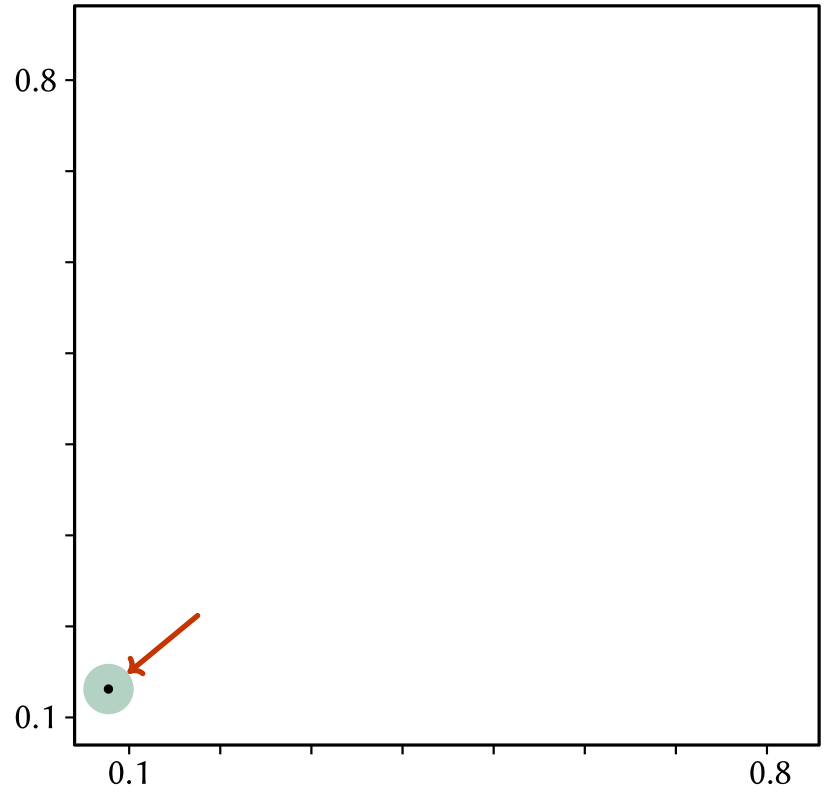
Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]



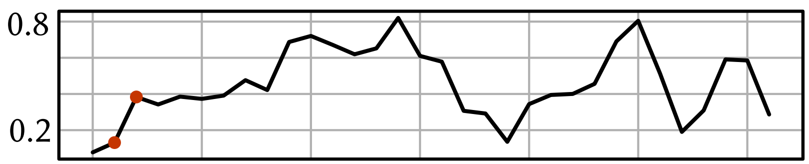
Example of an embedding in $\mathbb{R}^2$



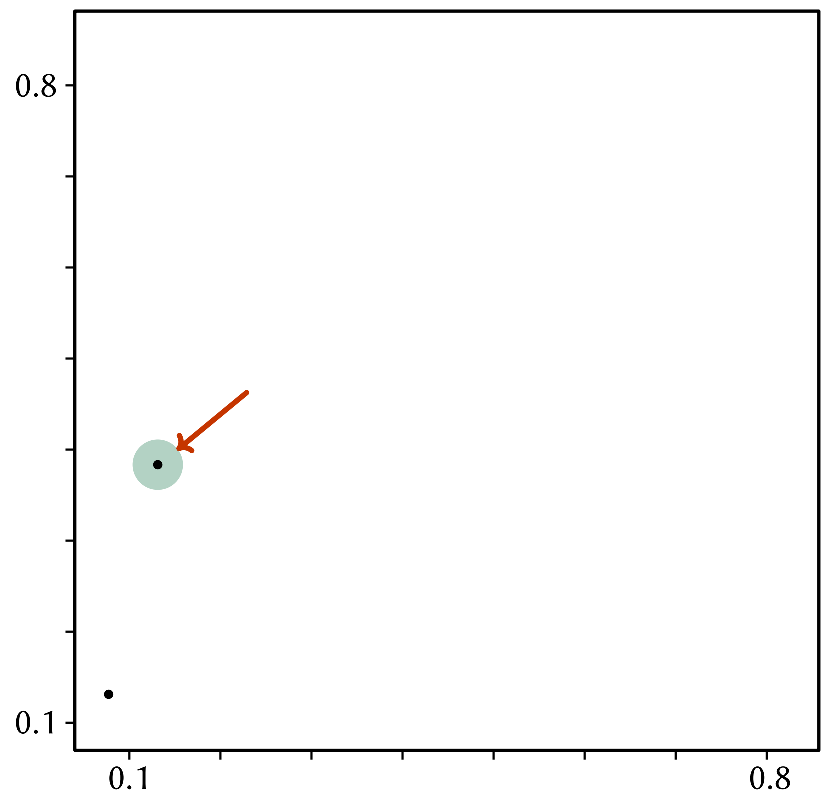
Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]



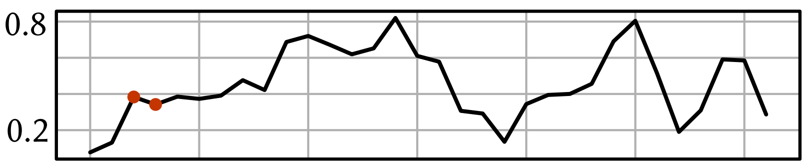
Example of an embedding in $\mathbb{R}^2$



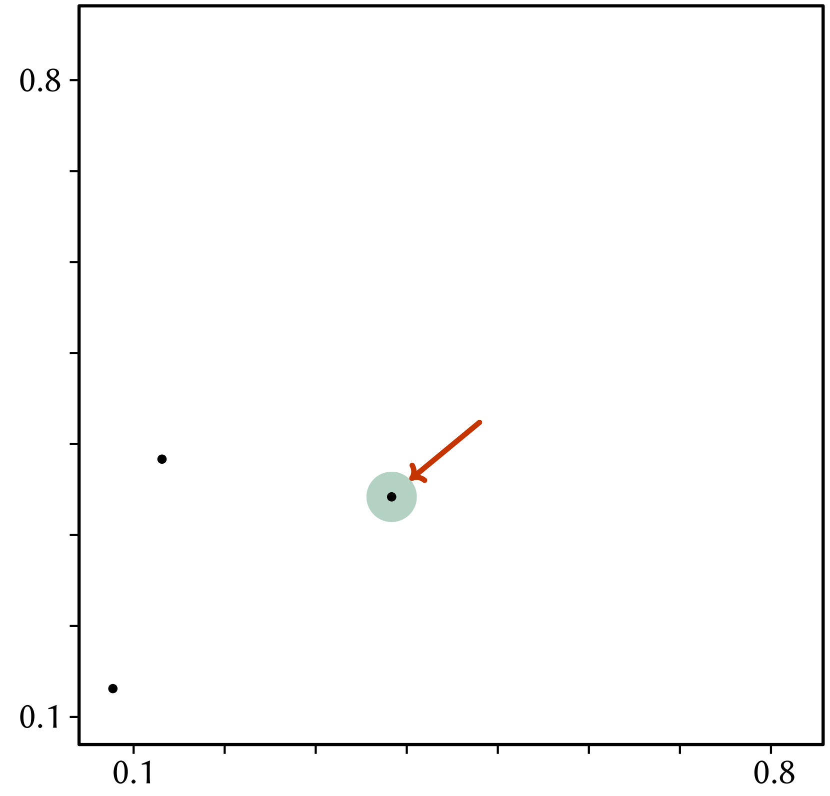
Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]



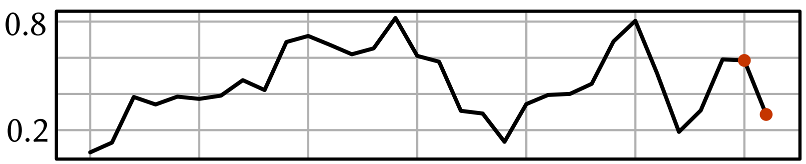
Example of an embedding in $\mathbb{R}^2$



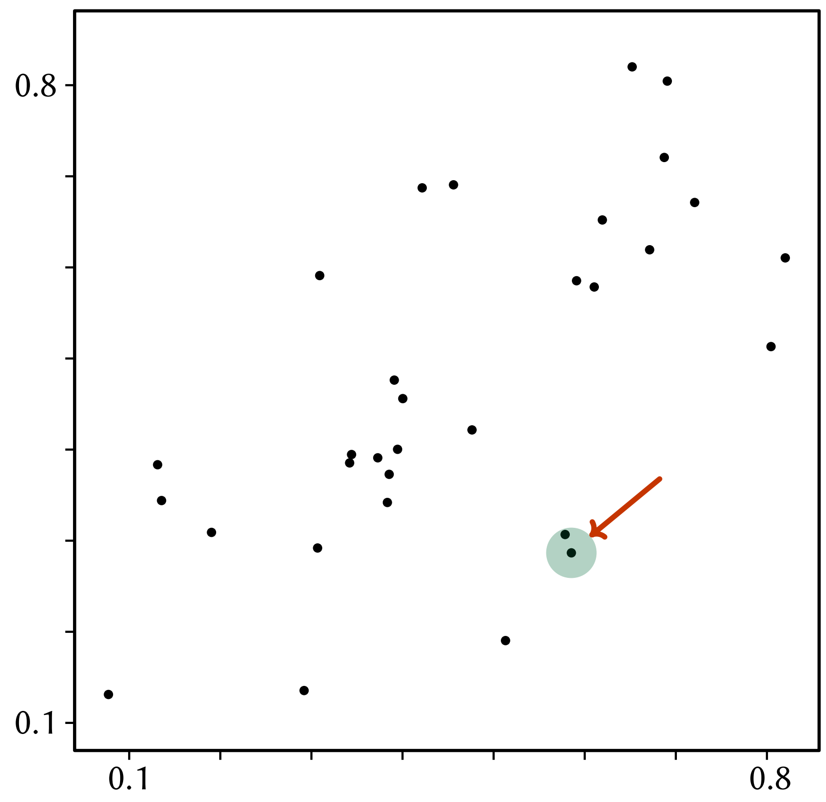
Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ ...

[de Silva et al., 2012; Perea & Harer, 2015]



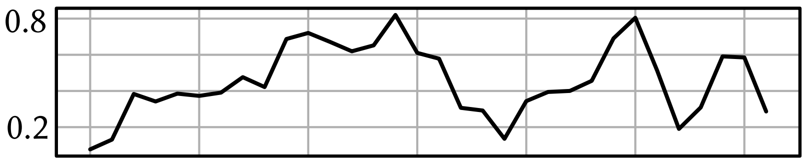
Example of an embedding in $\mathbb{R}^2$



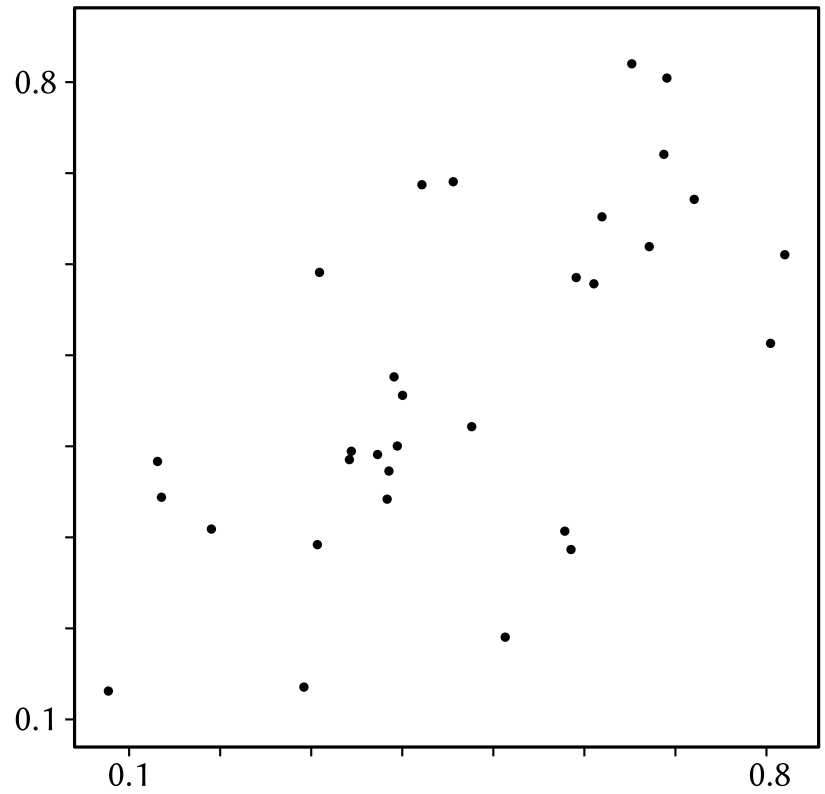
Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ ...

[de Silva et al., 2012; Perea & Harer, 2015]



Example of an embedding in $\mathbb{R}^2$

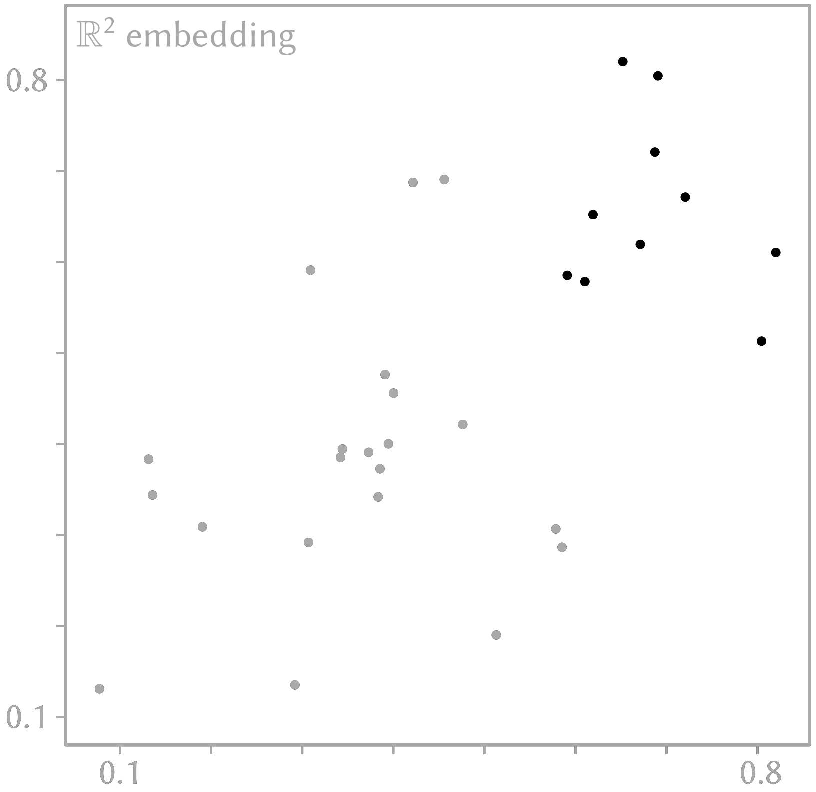


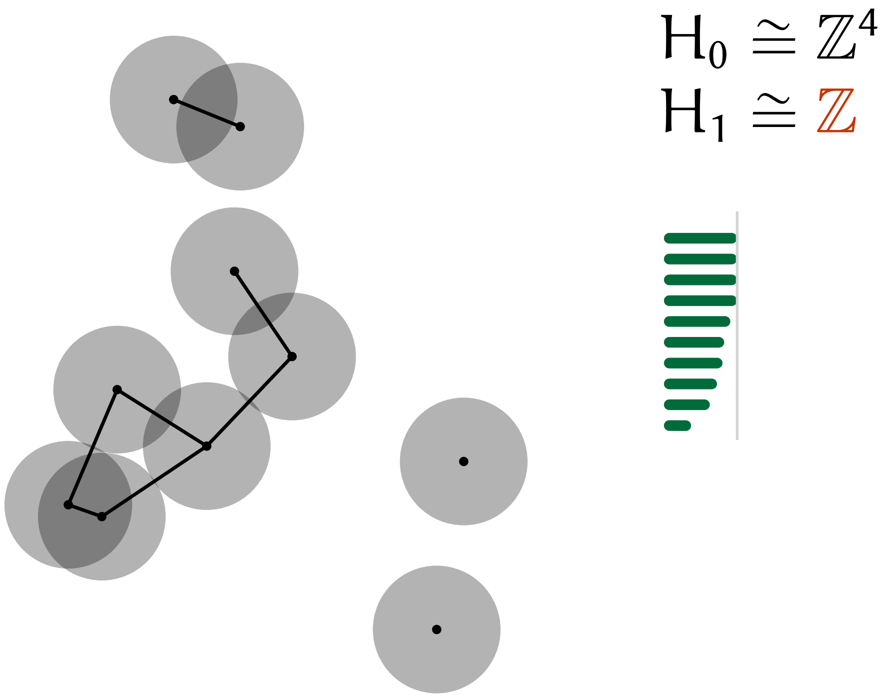
... and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]



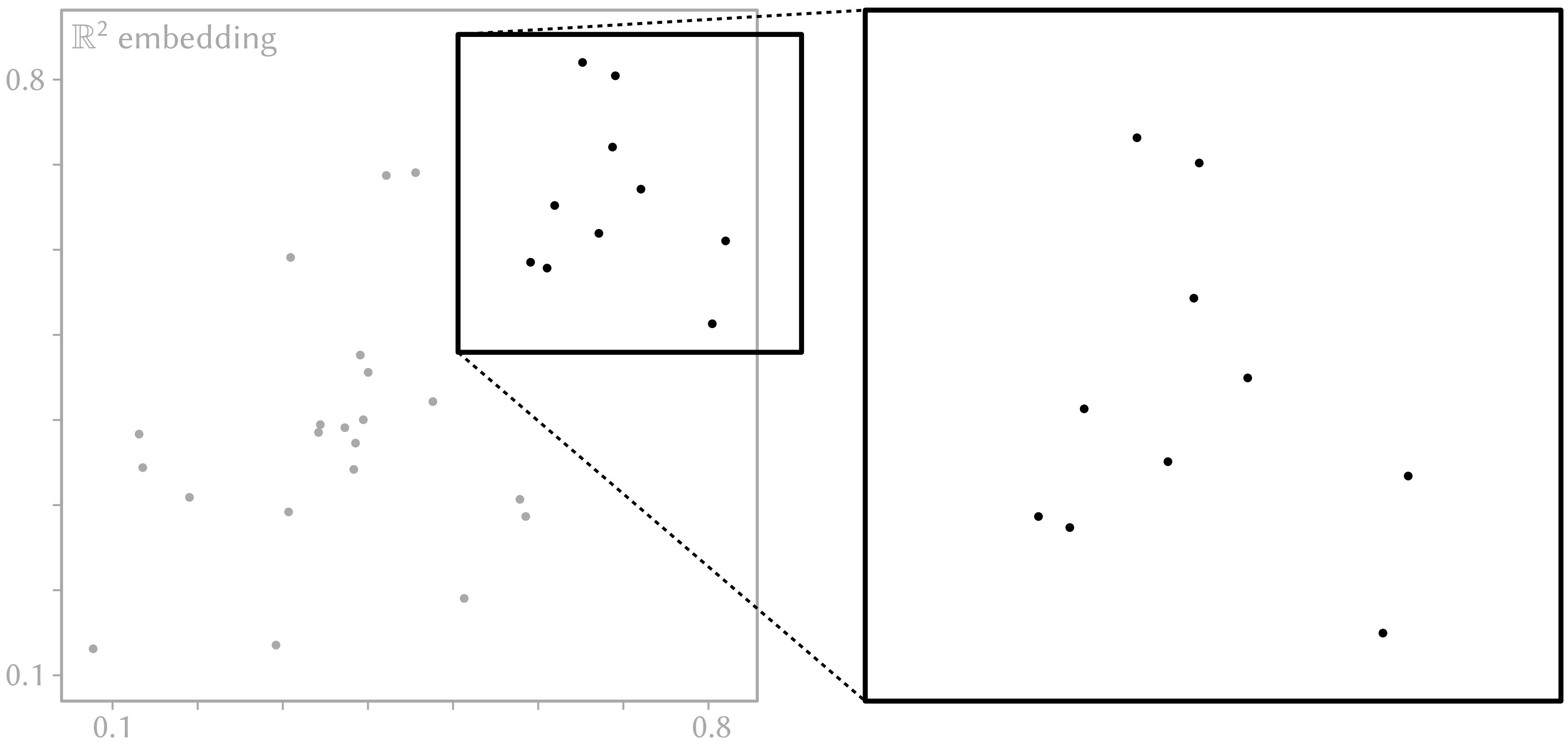


… and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]



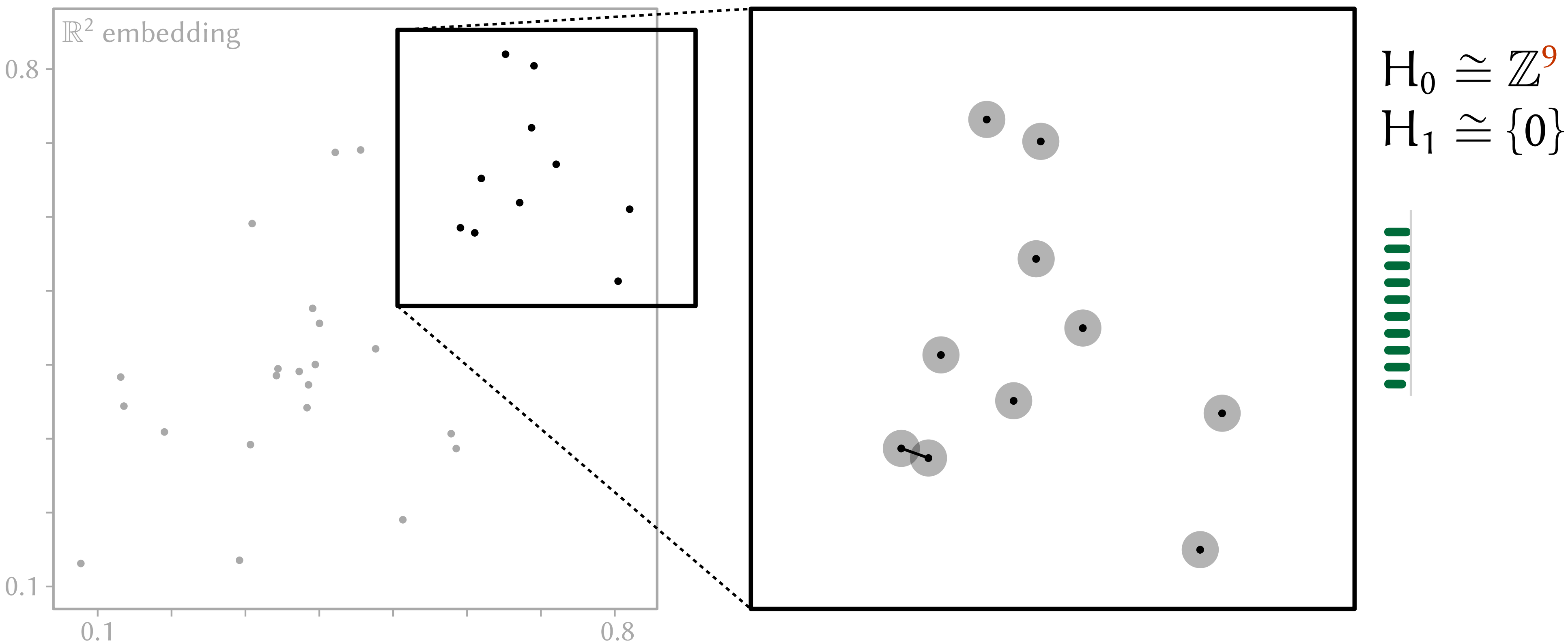
$H_0 \cong \mathbb{Z}^{10}$
$H_1 \cong \{0\}$

… and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]

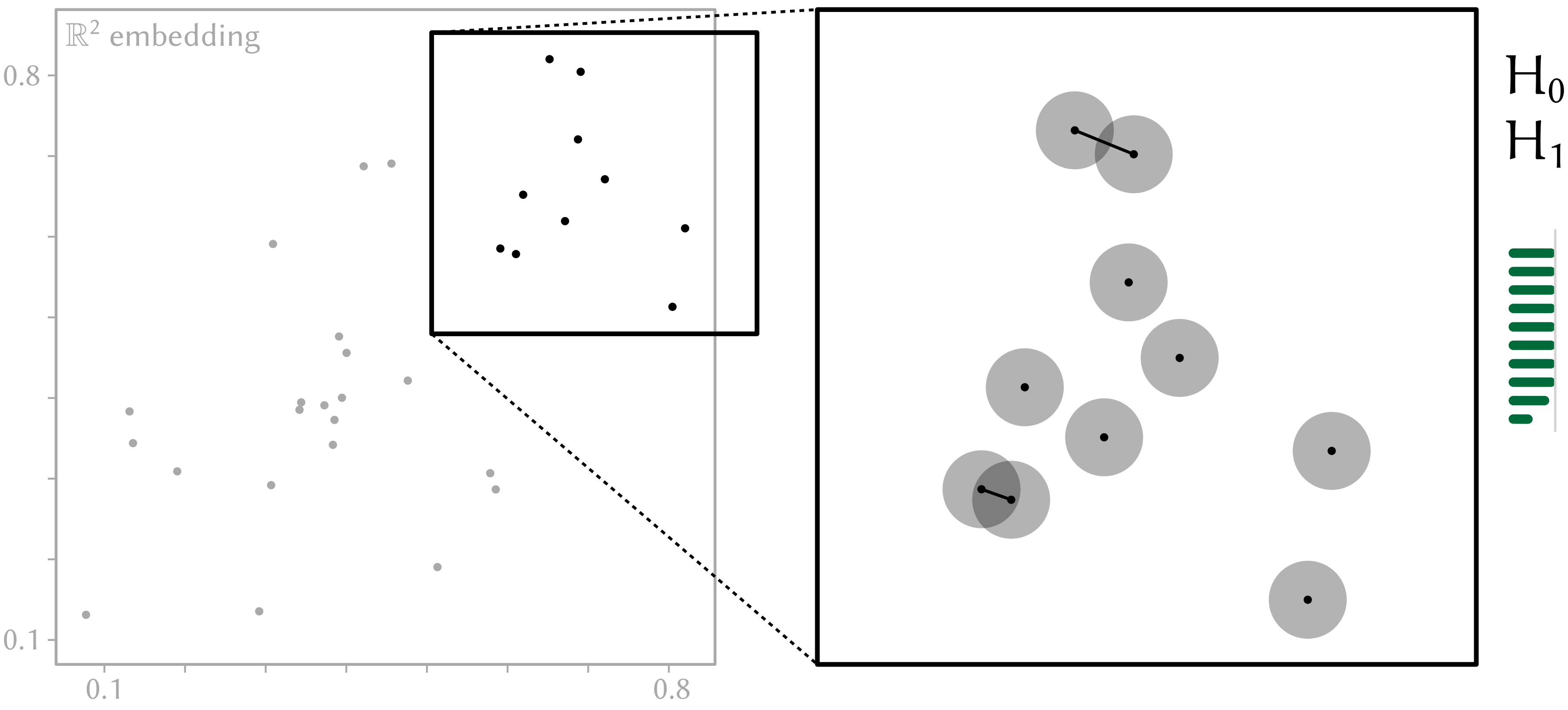


… and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]

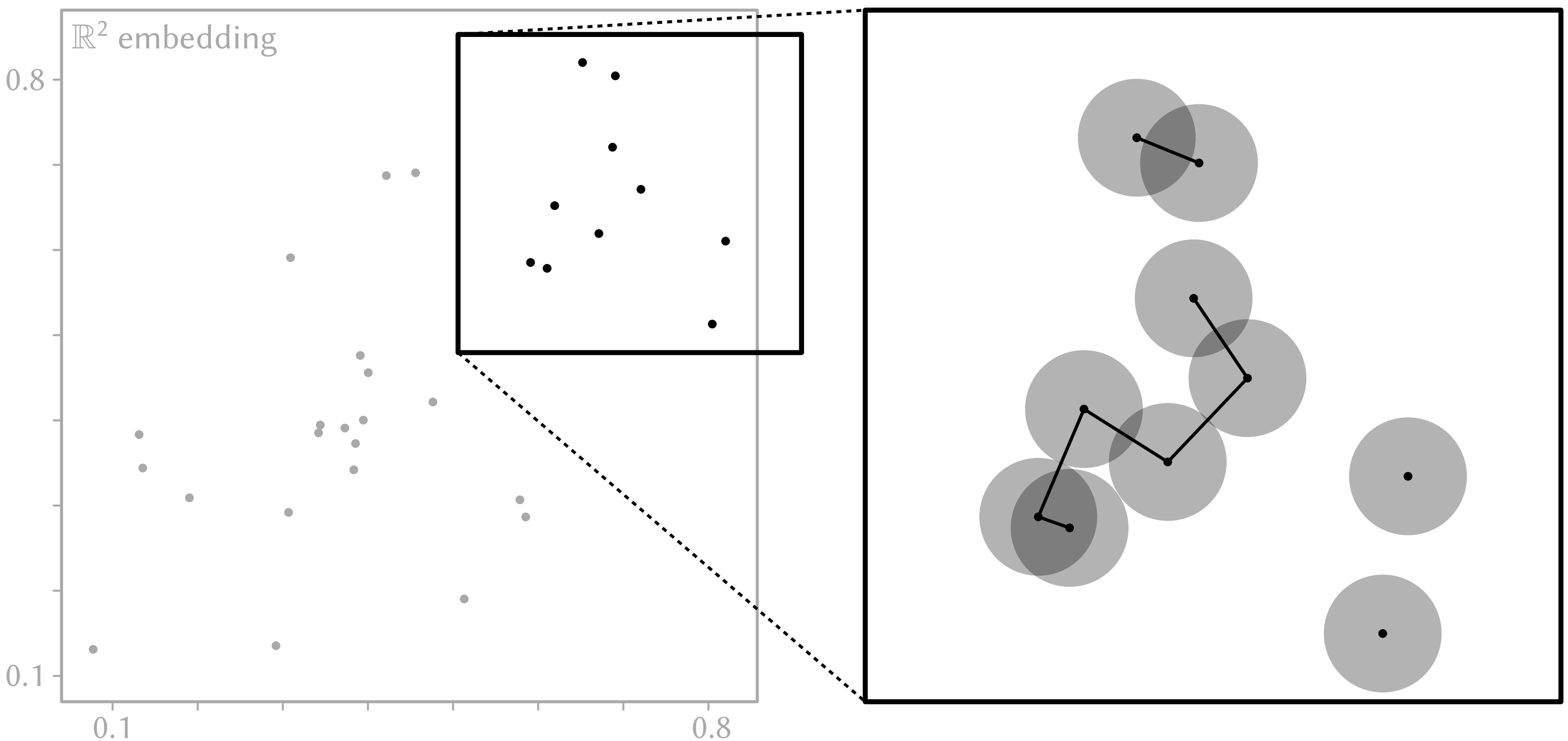


… and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]



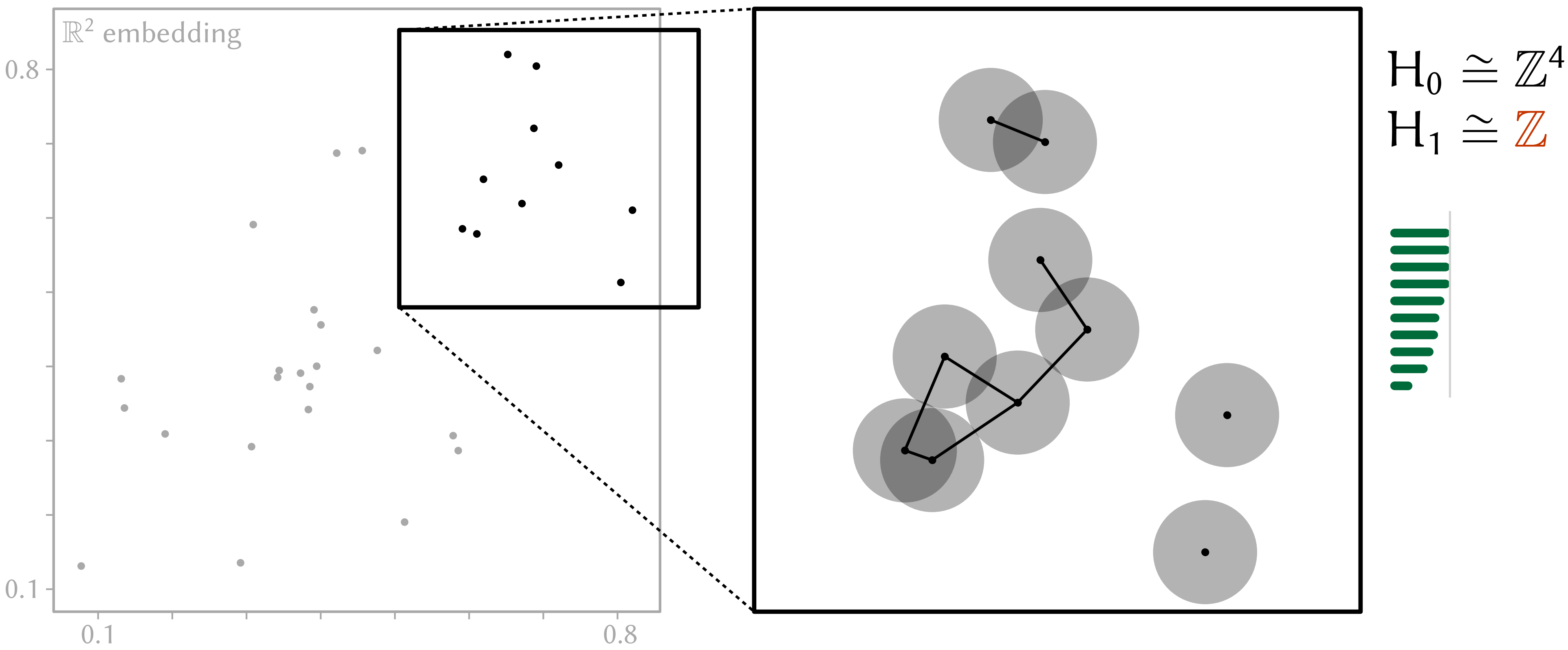
$H_0 \cong \mathbb{Z}^4$
$H_1 \cong \{0\}$

… and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]

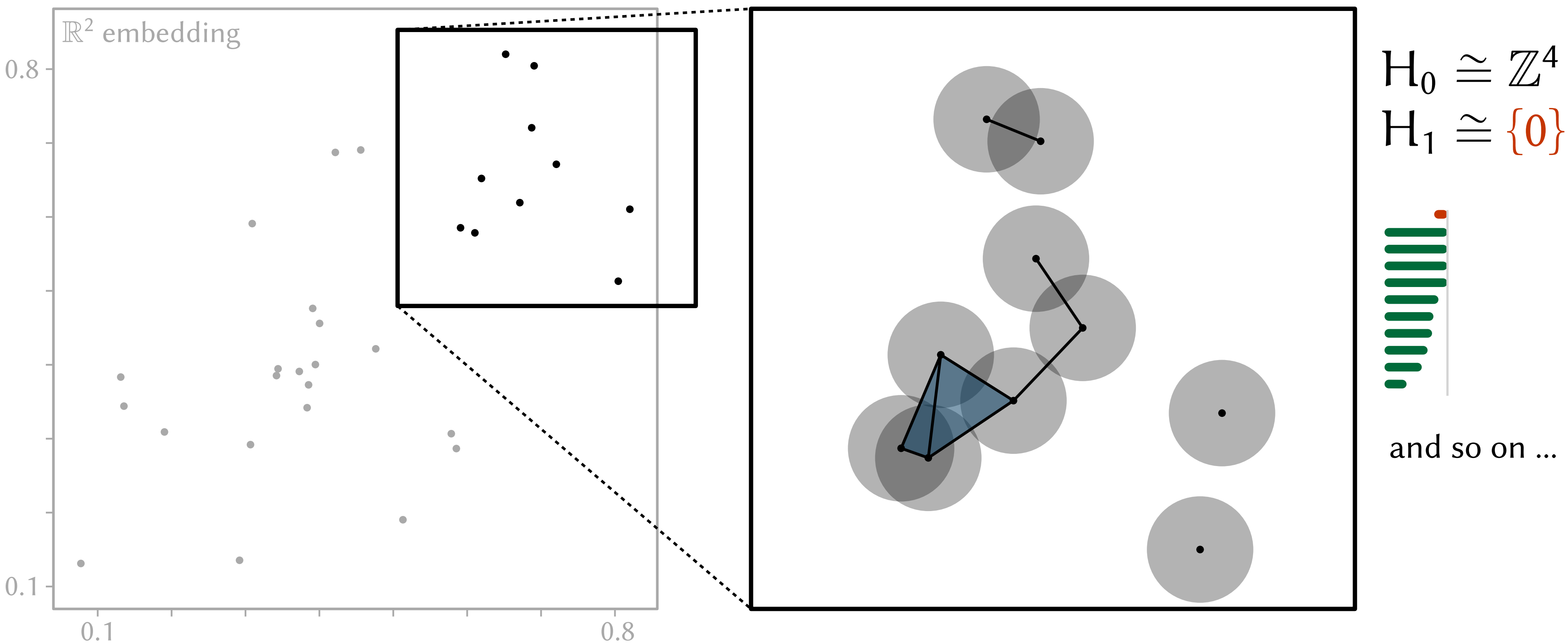


… and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ ...

[de Silva et al., 2012; Perea & Harer, 2015]

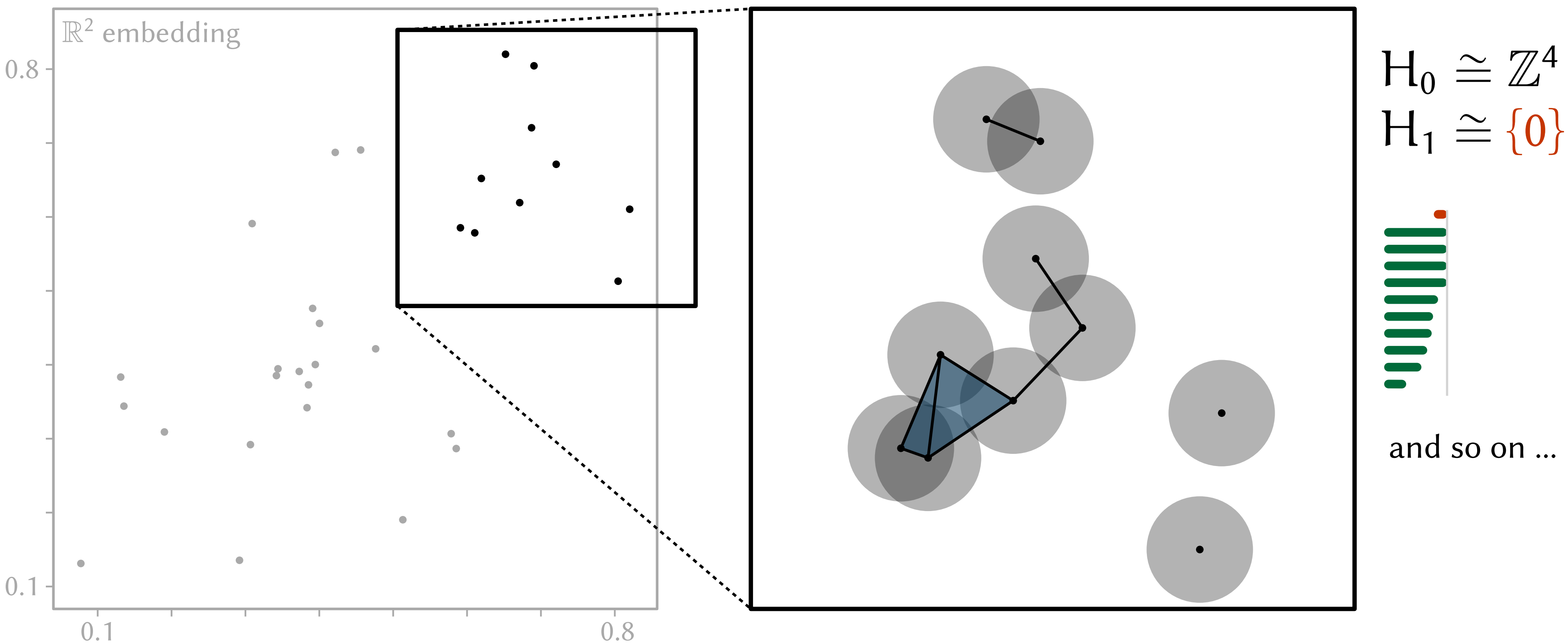


... and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ ...

[de Silva et al., 2012; Perea & Harer, 2015]



... and then compute **Vietoris-Rips persistent homology**!

Global topological features of time series

Classicially, one would first embed the time series into $\mathbb{R}^n$ …

[de Silva et al., 2012; Perea & Harer, 2015]

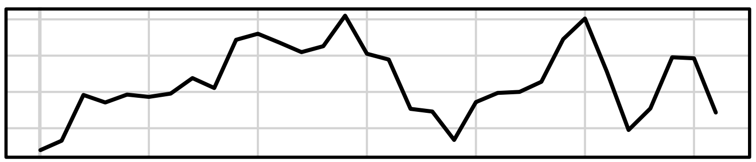
This tracks **global** topological changes!

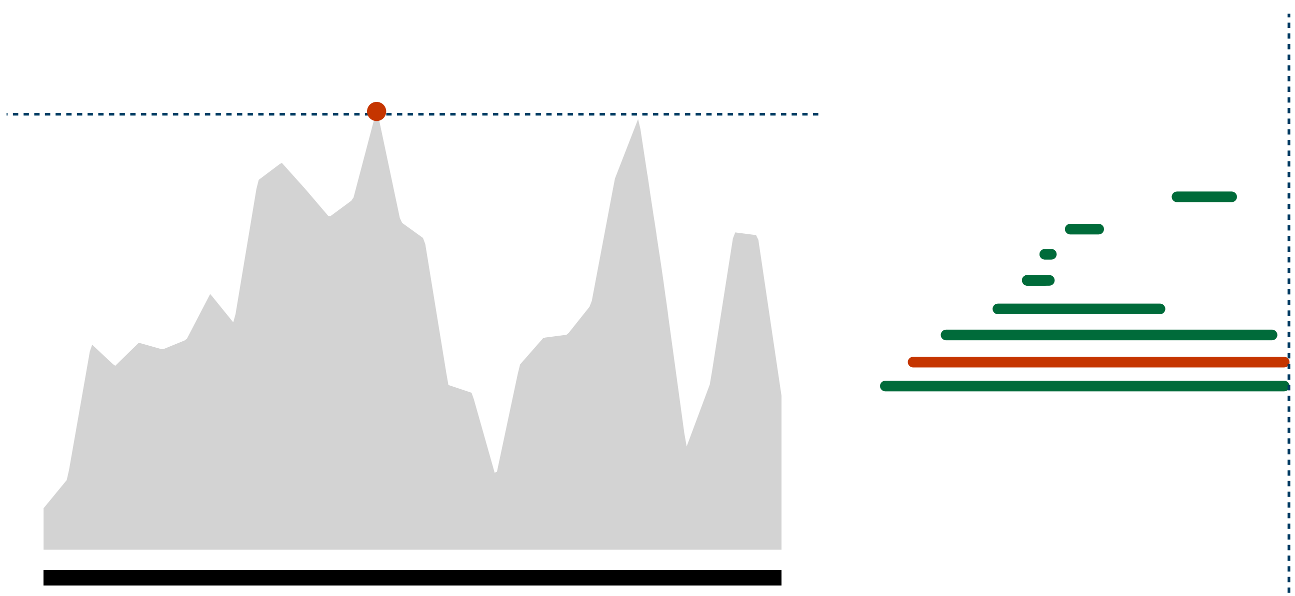
(and *locality* is lost)

… and then compute **Vietoris-Rips persistent homology**!

Local topological features for time series

Why not compute such topological features across **sliding windows**?

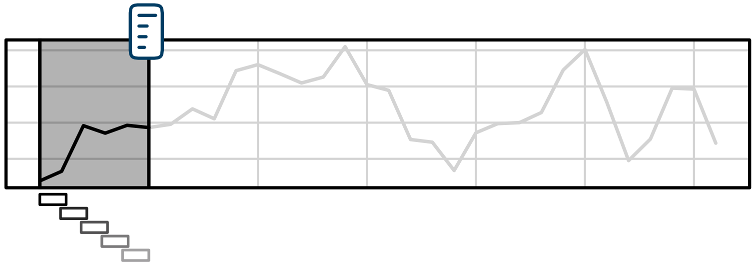




PH

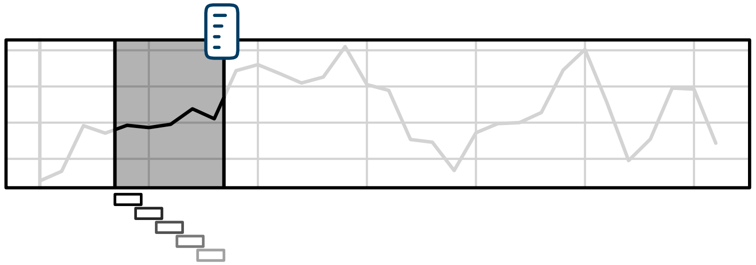
Local topological features for time series

Why not compute such topological features across **sliding windows**?



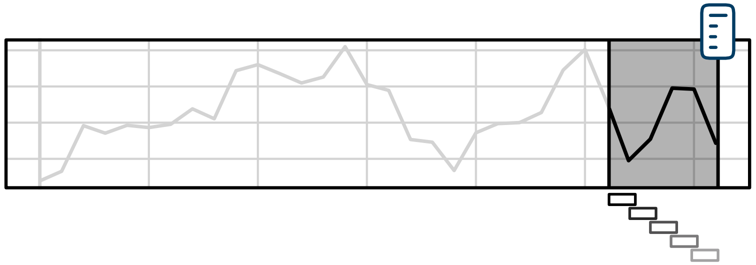
Local topological features for time series

Why not compute such topological features across **sliding windows**?



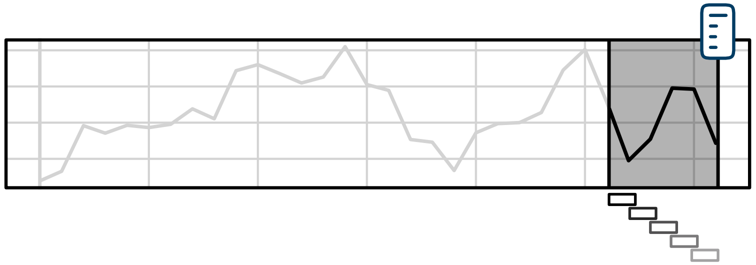
Local topological features for time series

Why not compute such topological features across **sliding windows**?



Local topological features for time series

Why not compute such topological features across **sliding windows**?



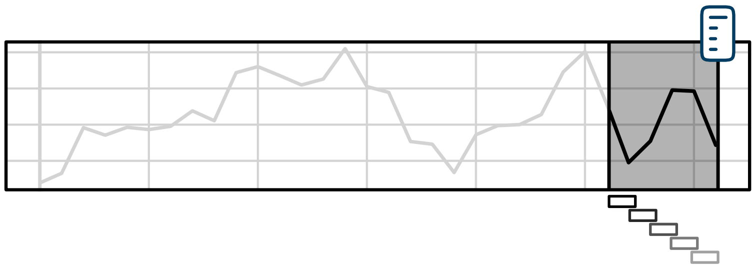
✓ Locality

✗ Little information per window

(as we need to chunk up the windows)

Local topological features for time series

Why not compute such topological features across **sliding windows**?

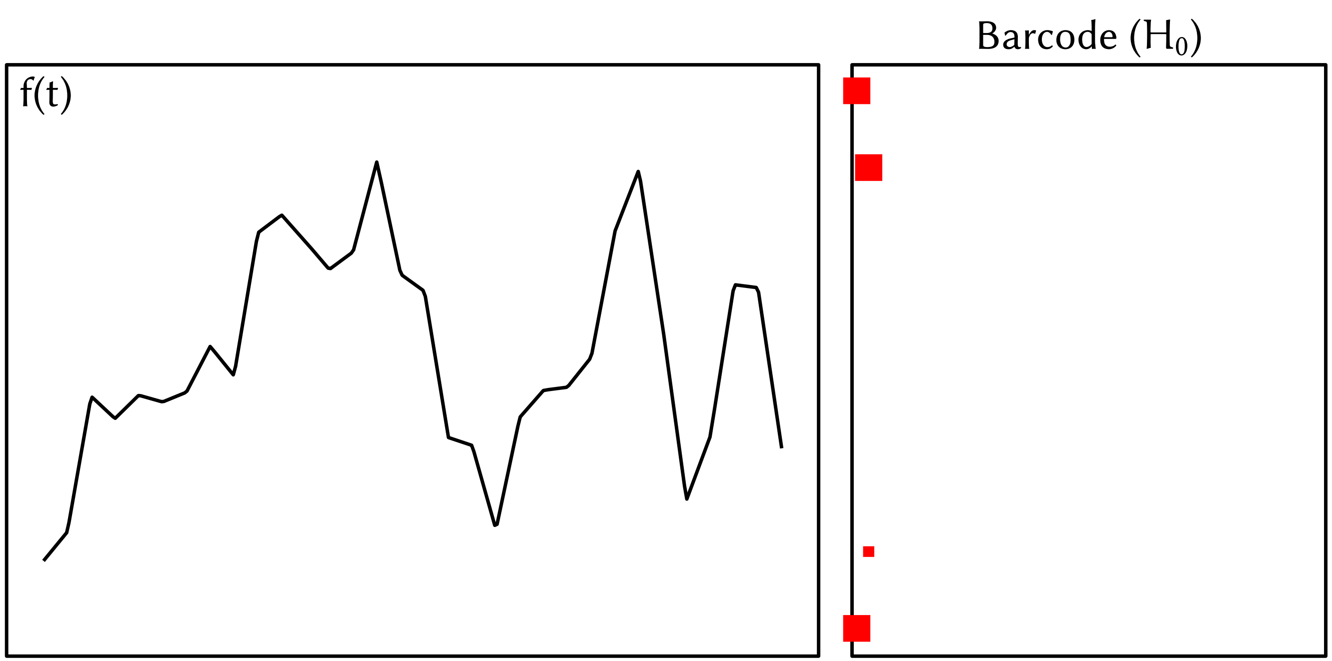


✓ Locality

✗ Little information per window

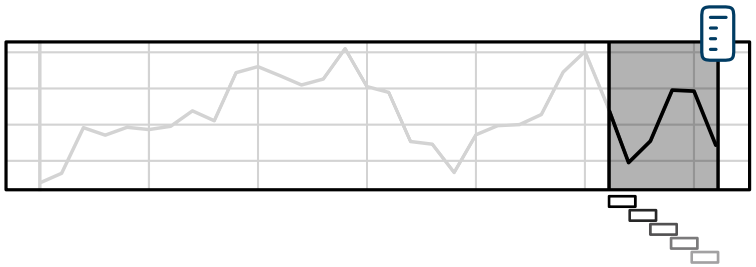
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

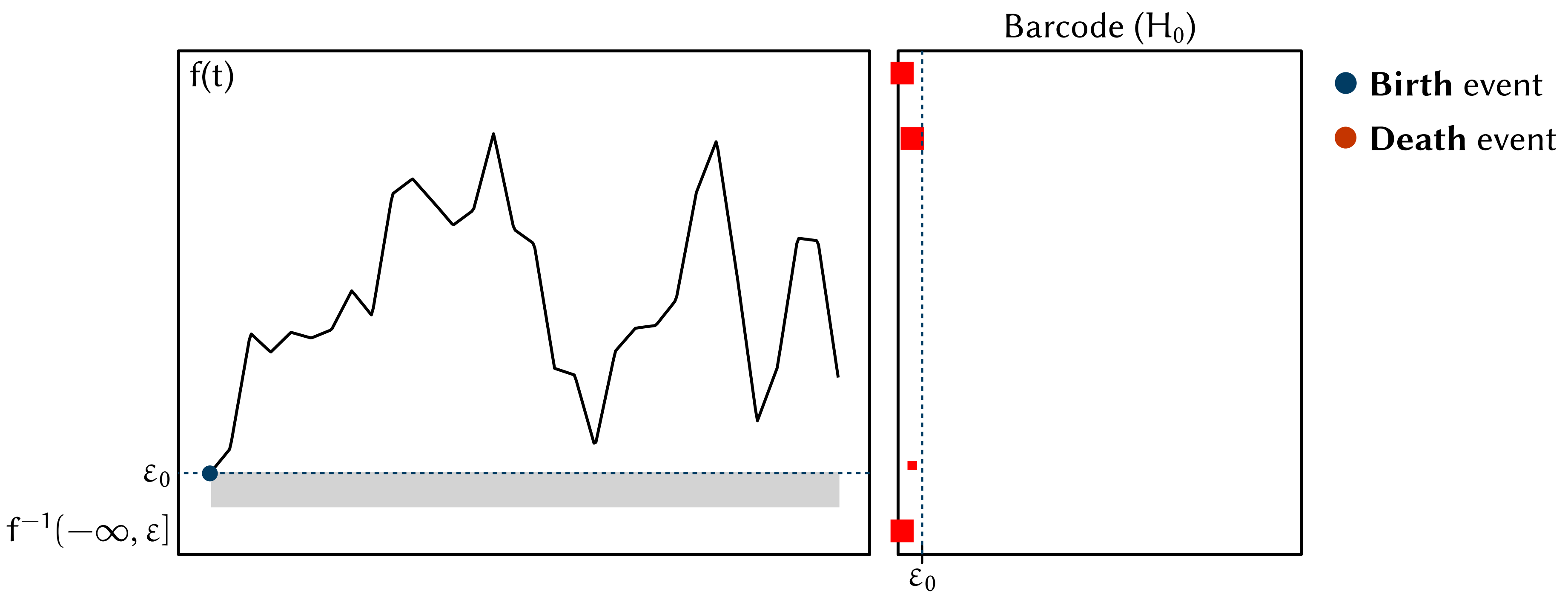


✓ Locality

✗ Little information per window

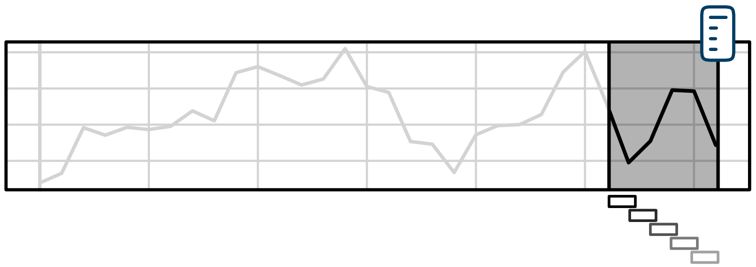
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

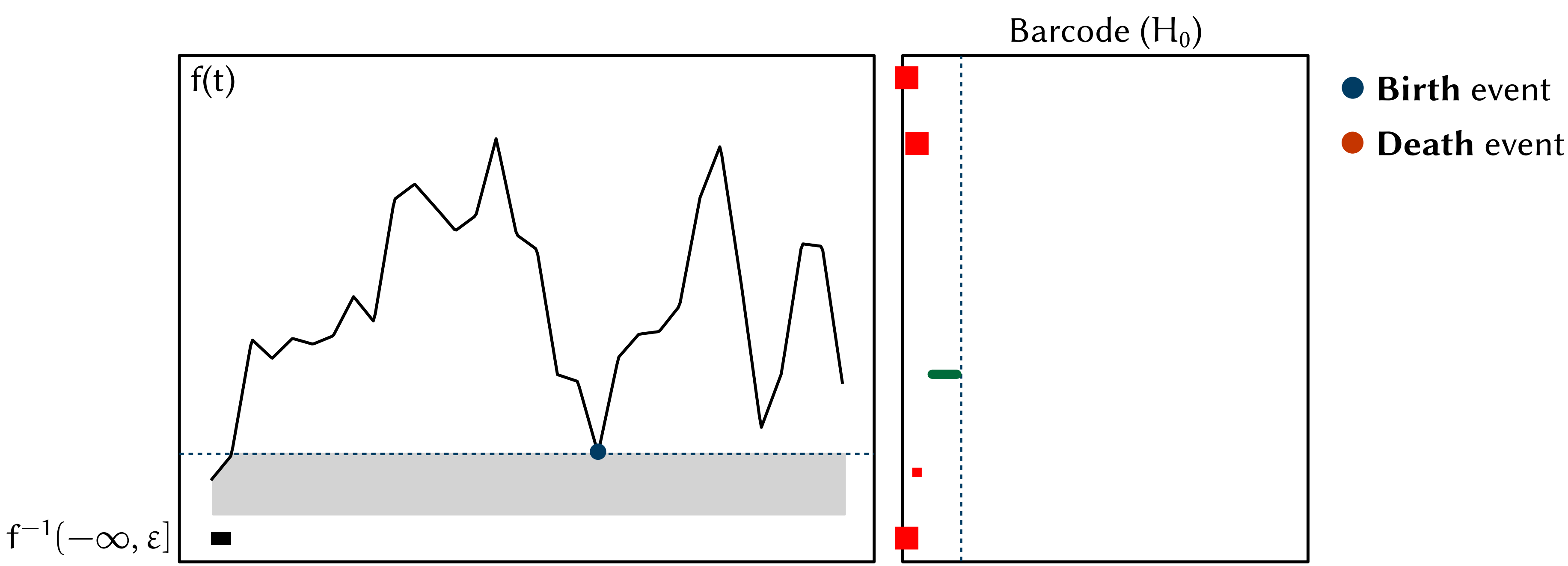


✓ Locality

✗ Little information per window

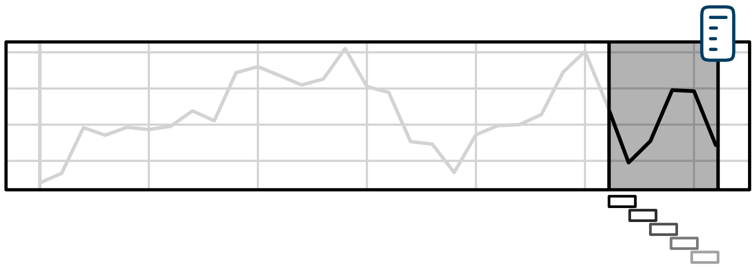
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

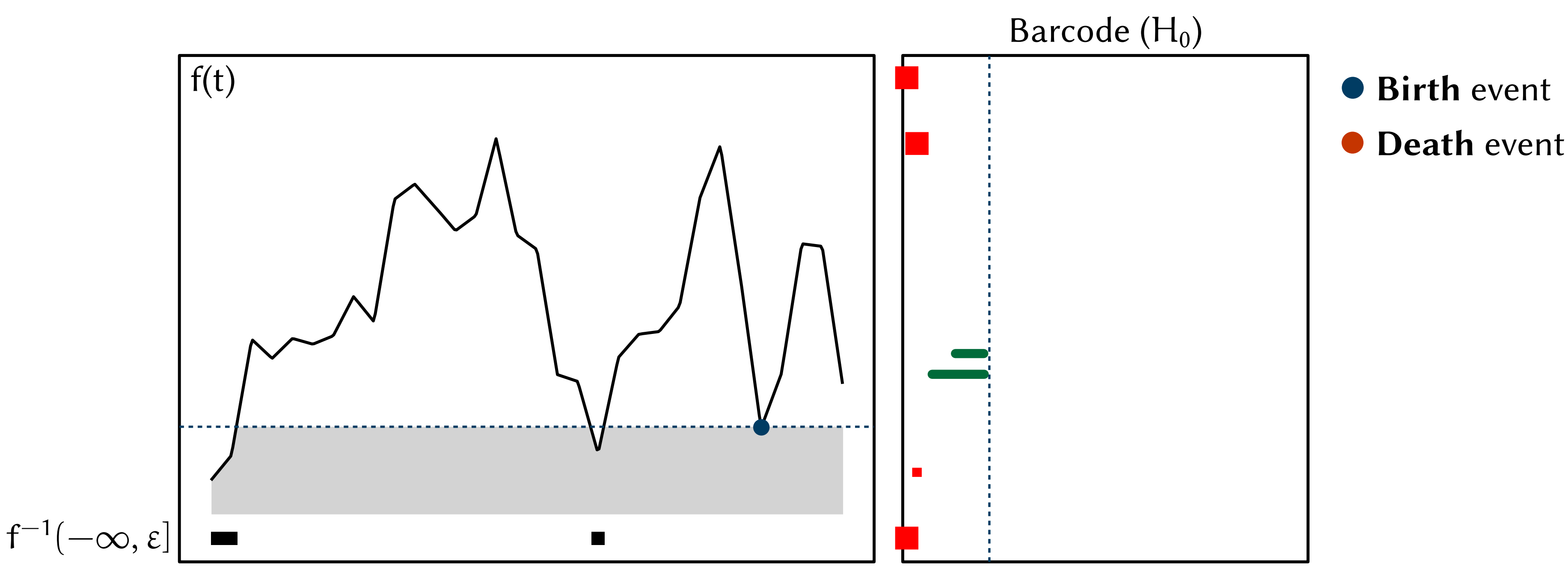


✓ Locality

✗ Little information per window

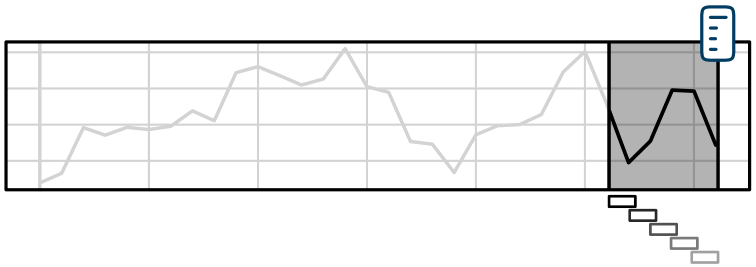
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

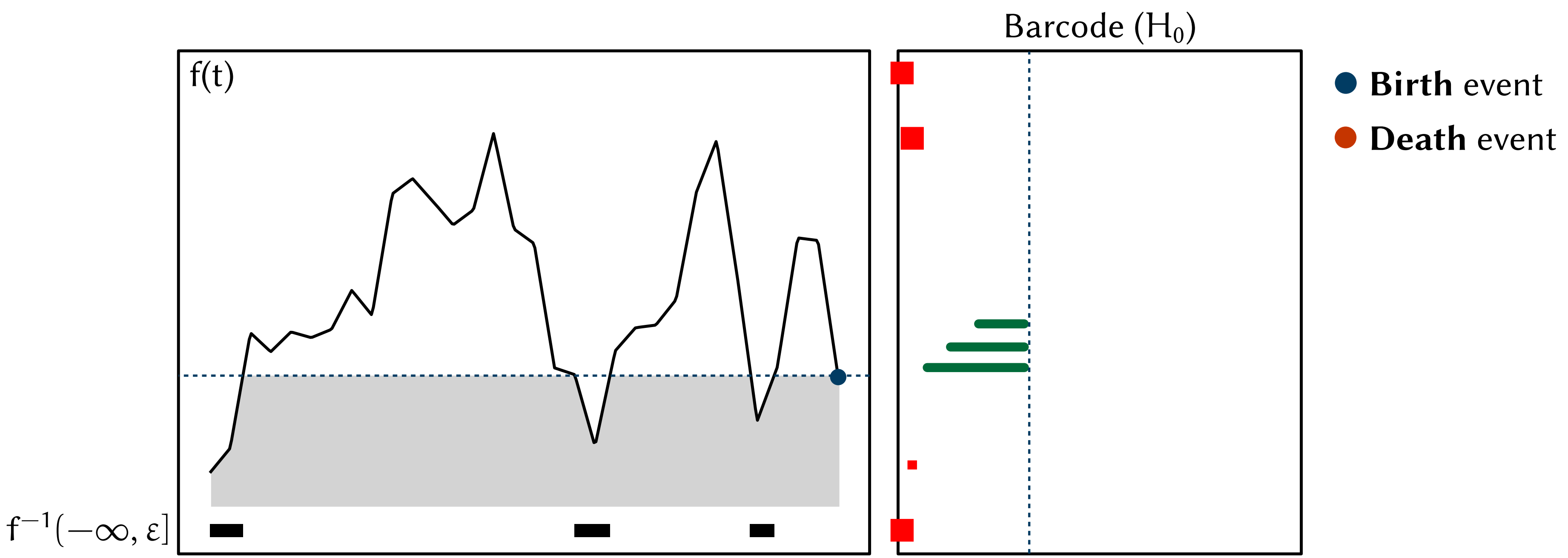


✓ Locality

✗ Little information per window

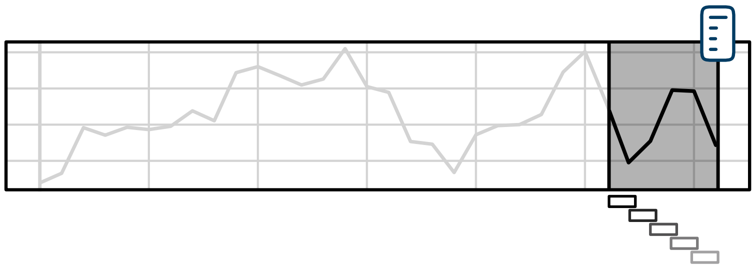
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

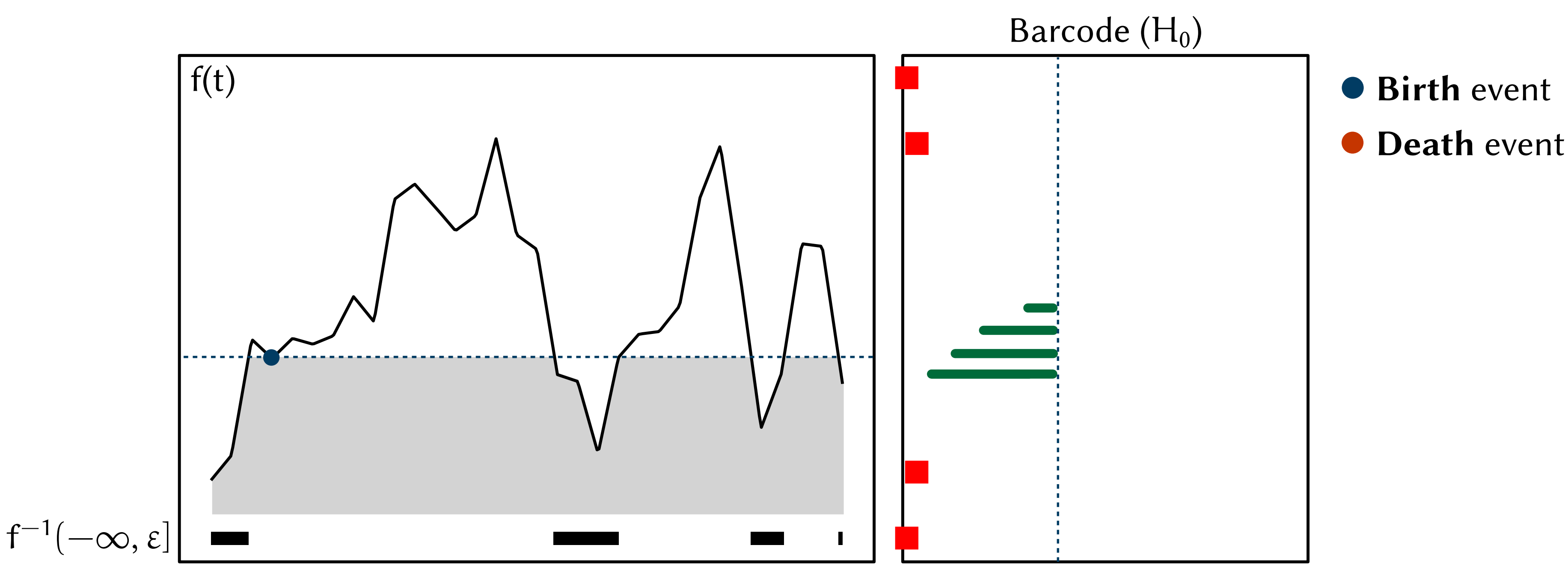


✓ Locality

✗ Little information per window

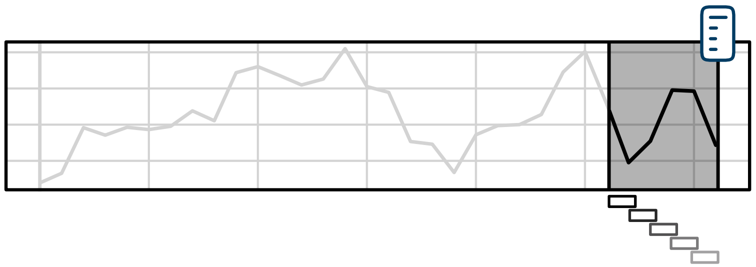
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

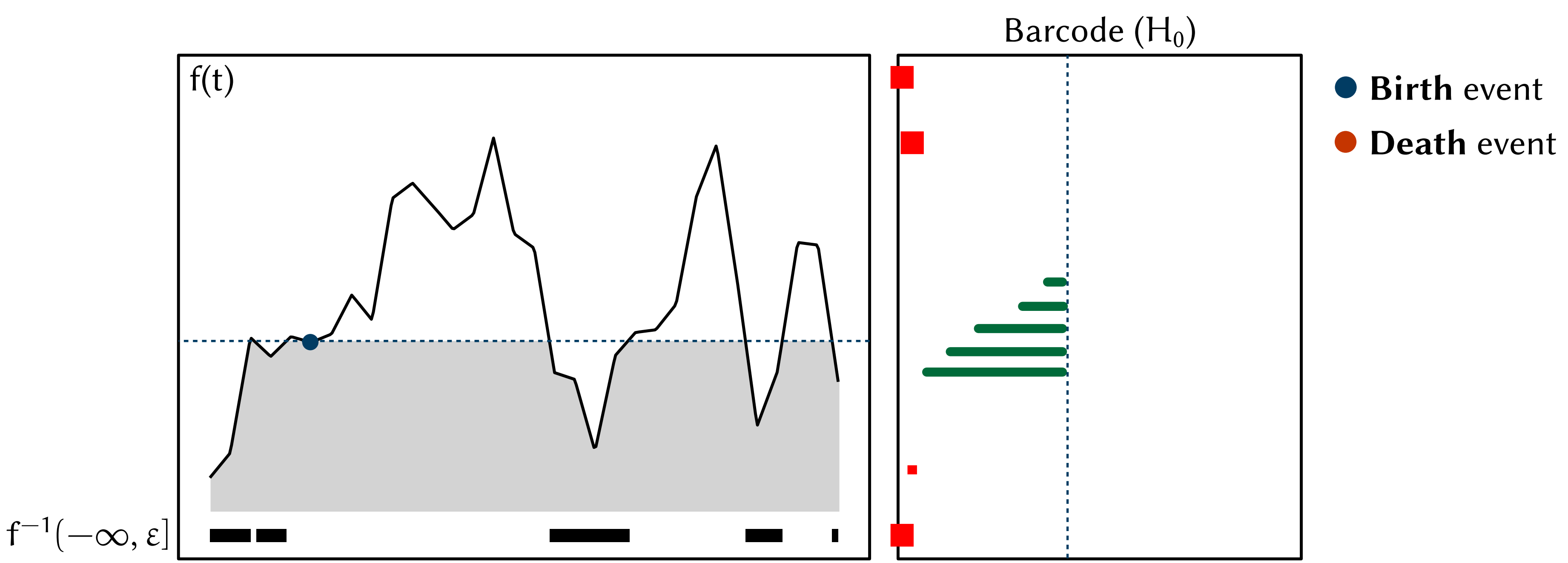


✓ Locality

✗ Little information per window

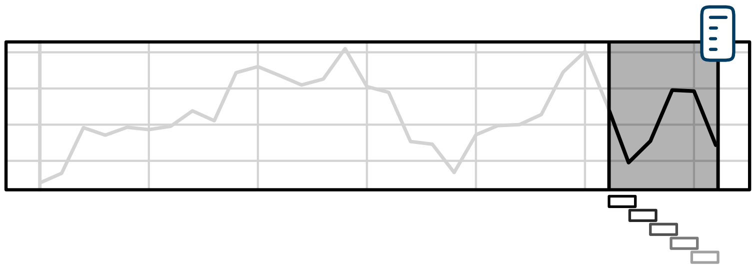
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

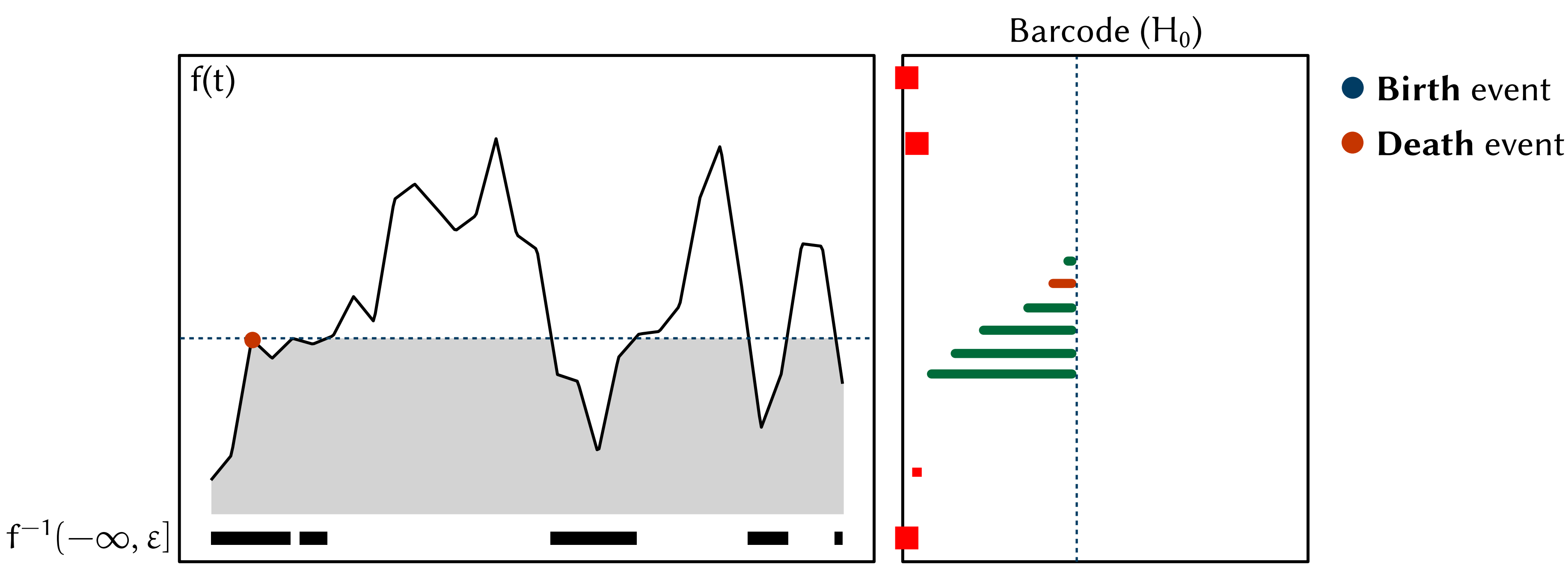


✓ Locality

✗ Little information per window

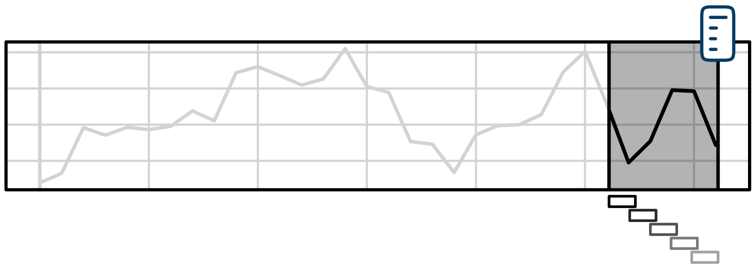
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

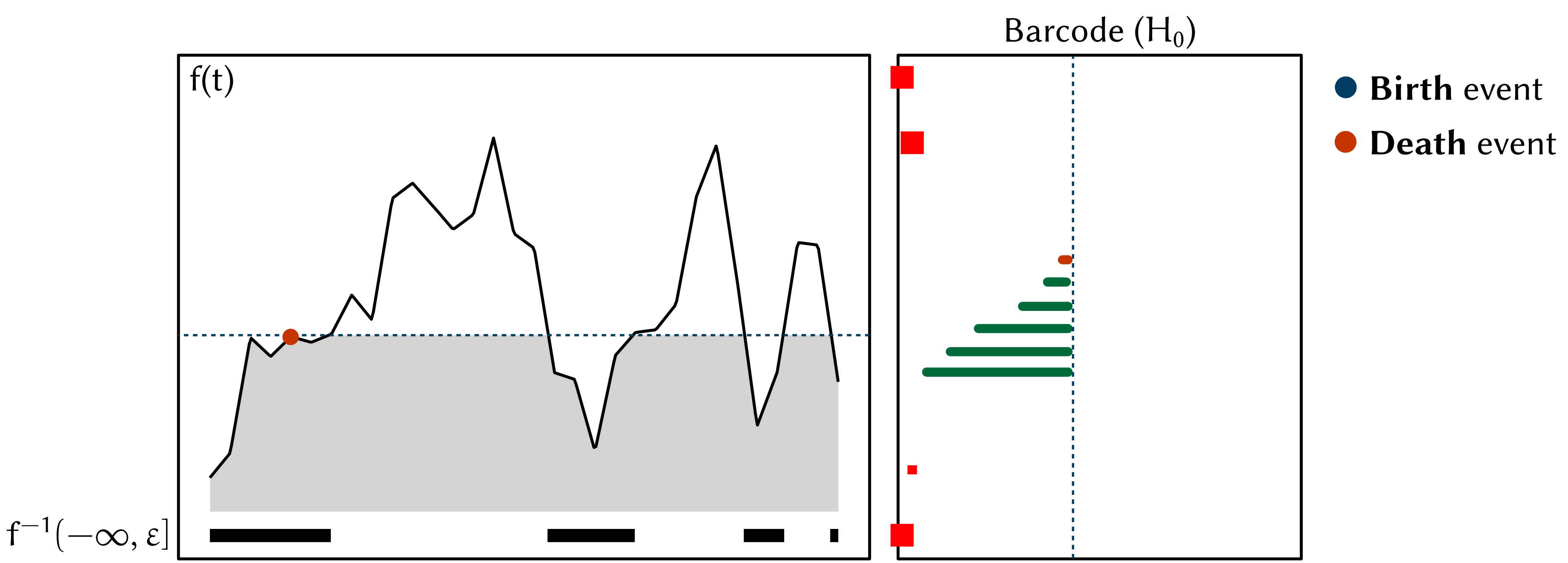


✓ Locality

✗ Little information per window

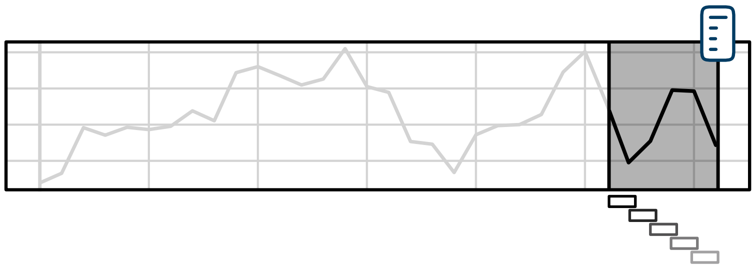
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

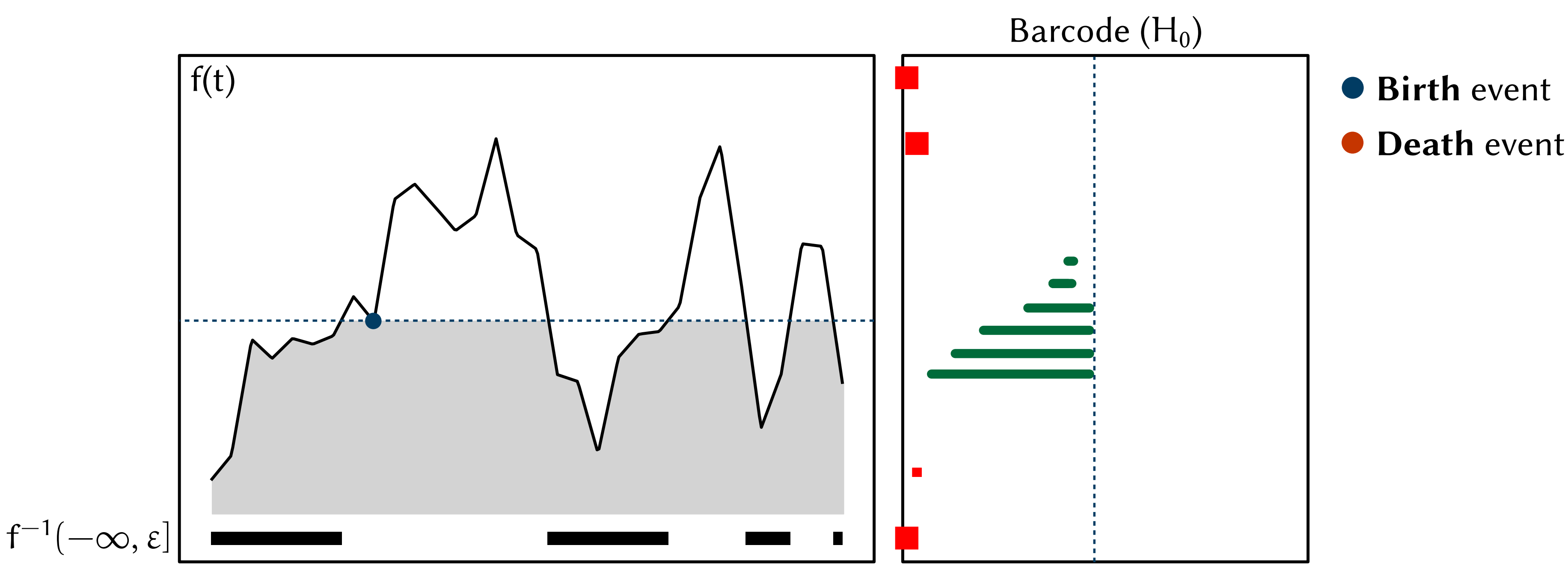


✓ Locality

✗ Little information per window

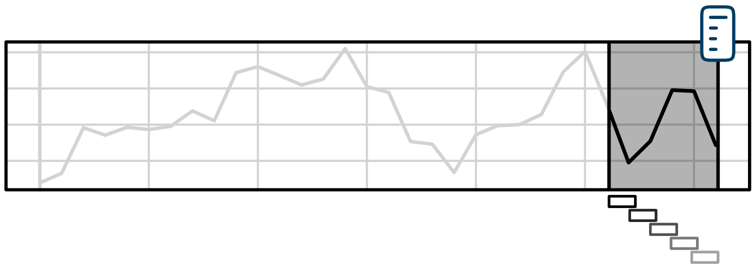
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

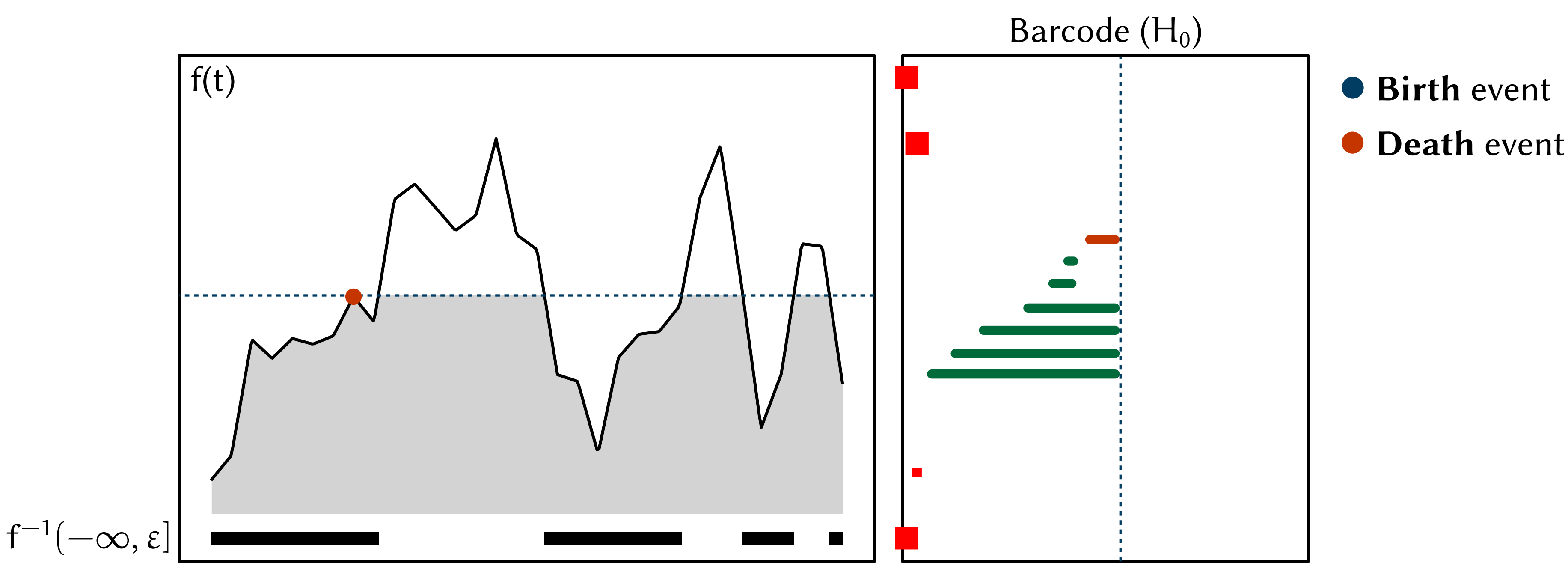


✓ Locality

✗ Little information per window

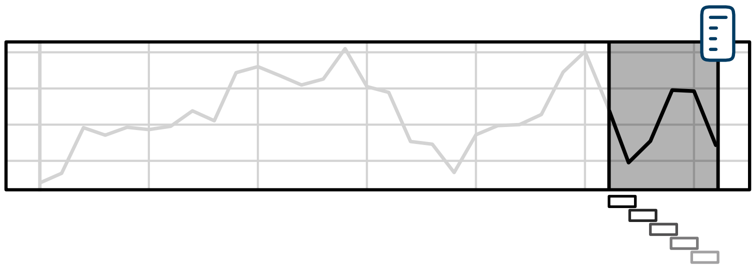
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

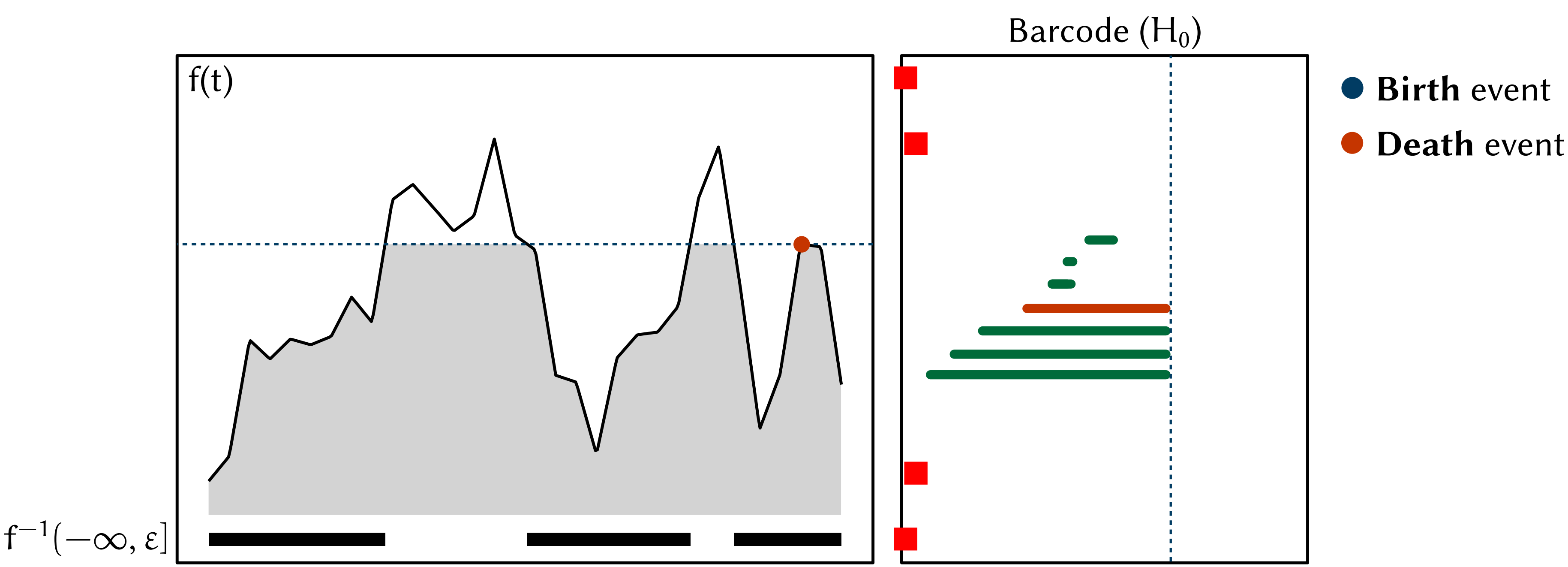


✓ Locality

✗ Little information per window

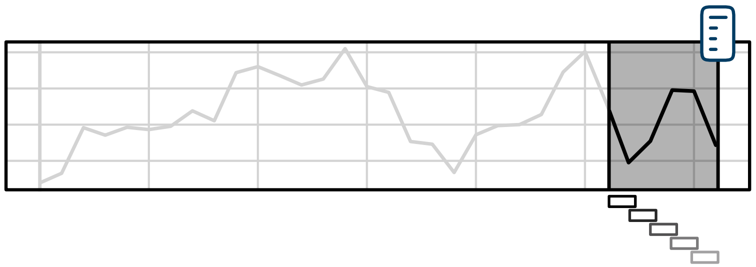
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

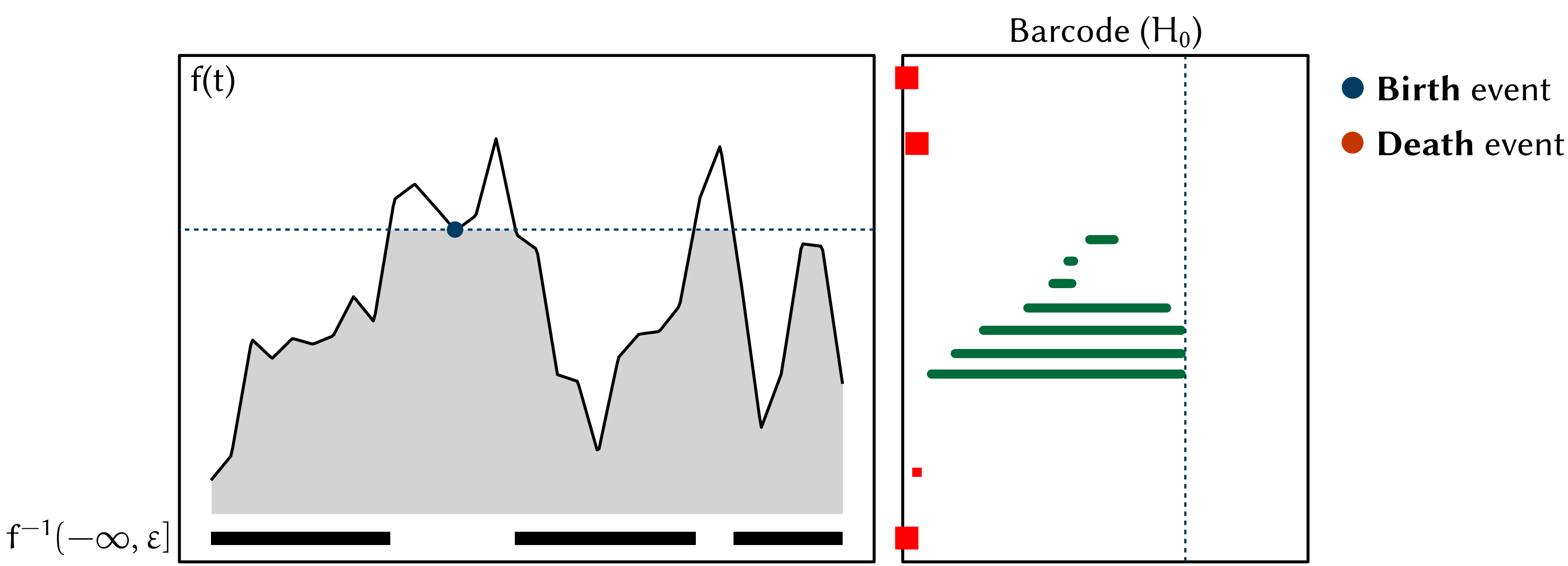


✓ Locality

✗ Little information per window

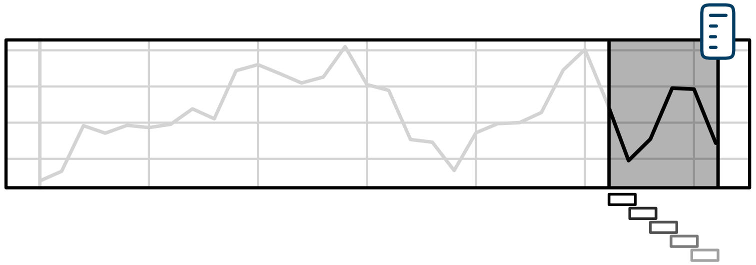
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

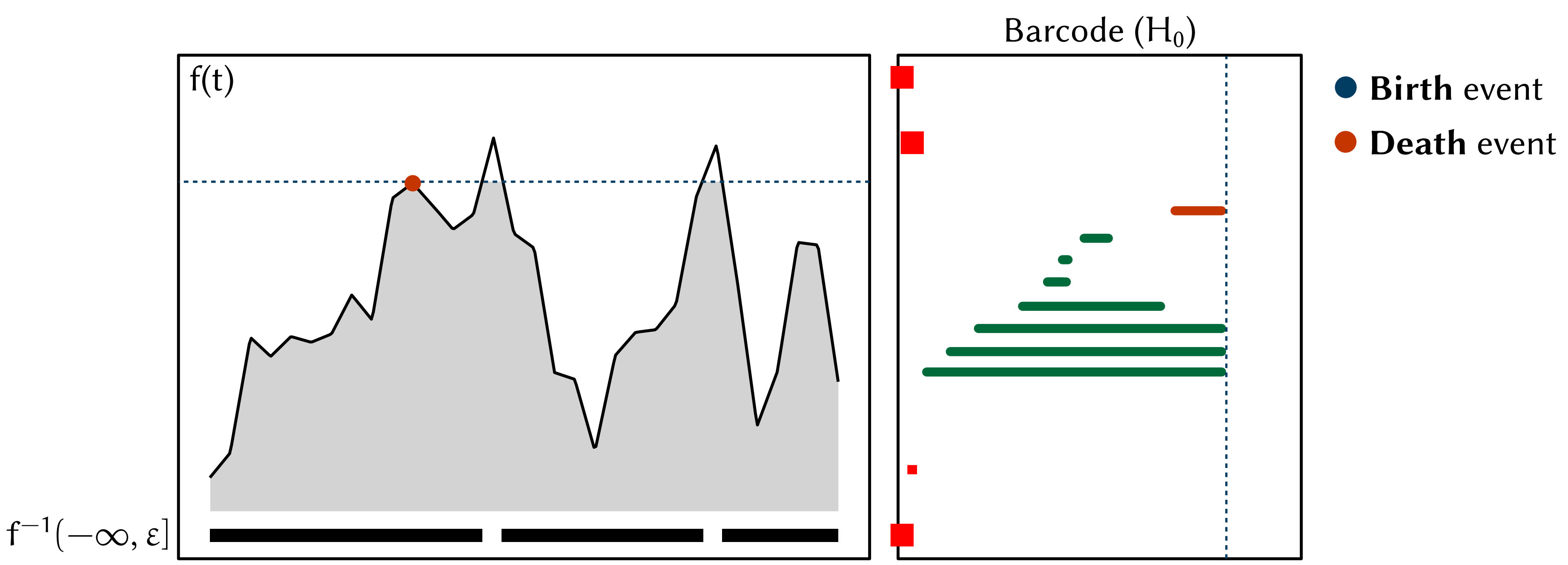


✓ Locality

✗ Little information per window

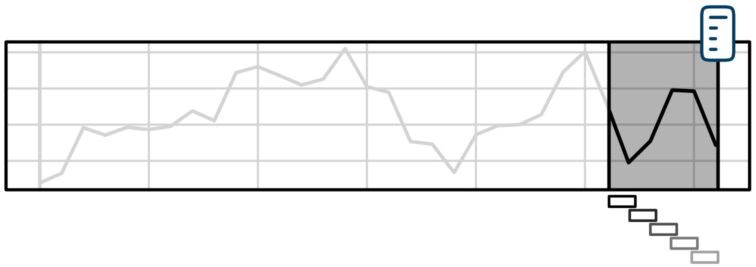
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

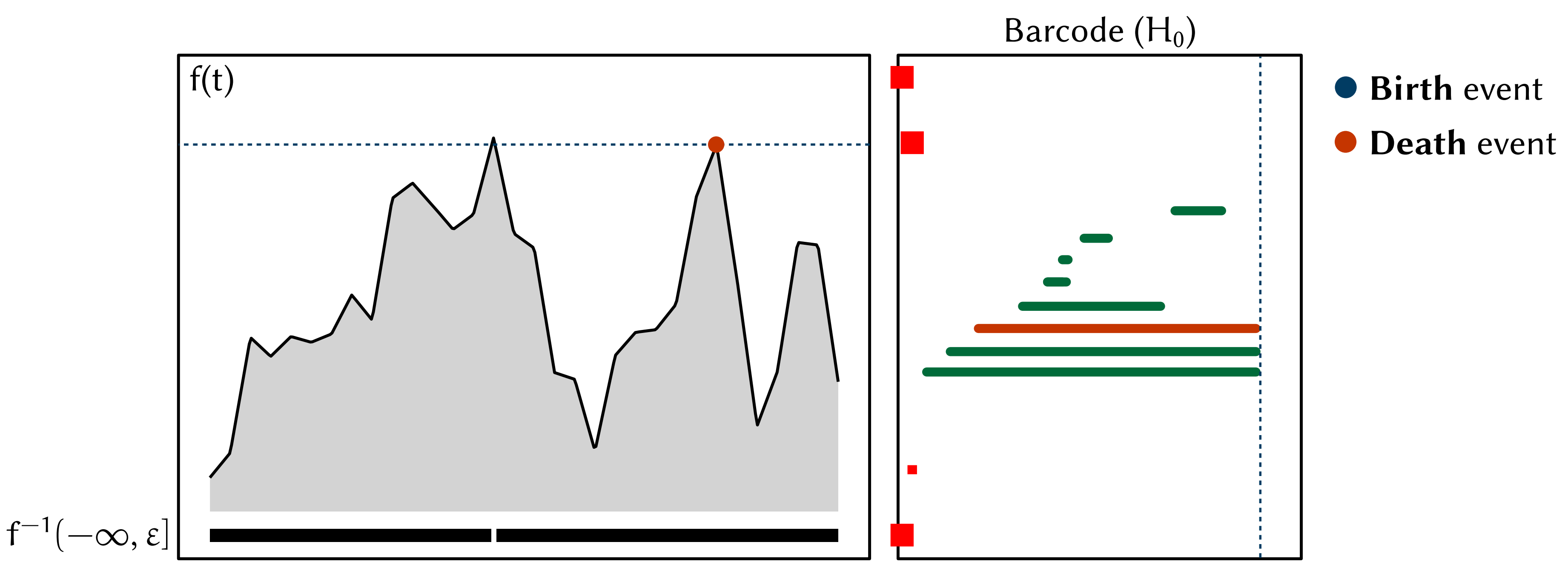


✓ Locality

✗ Little information per window

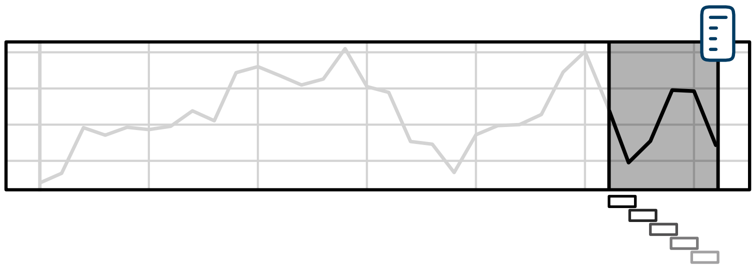
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

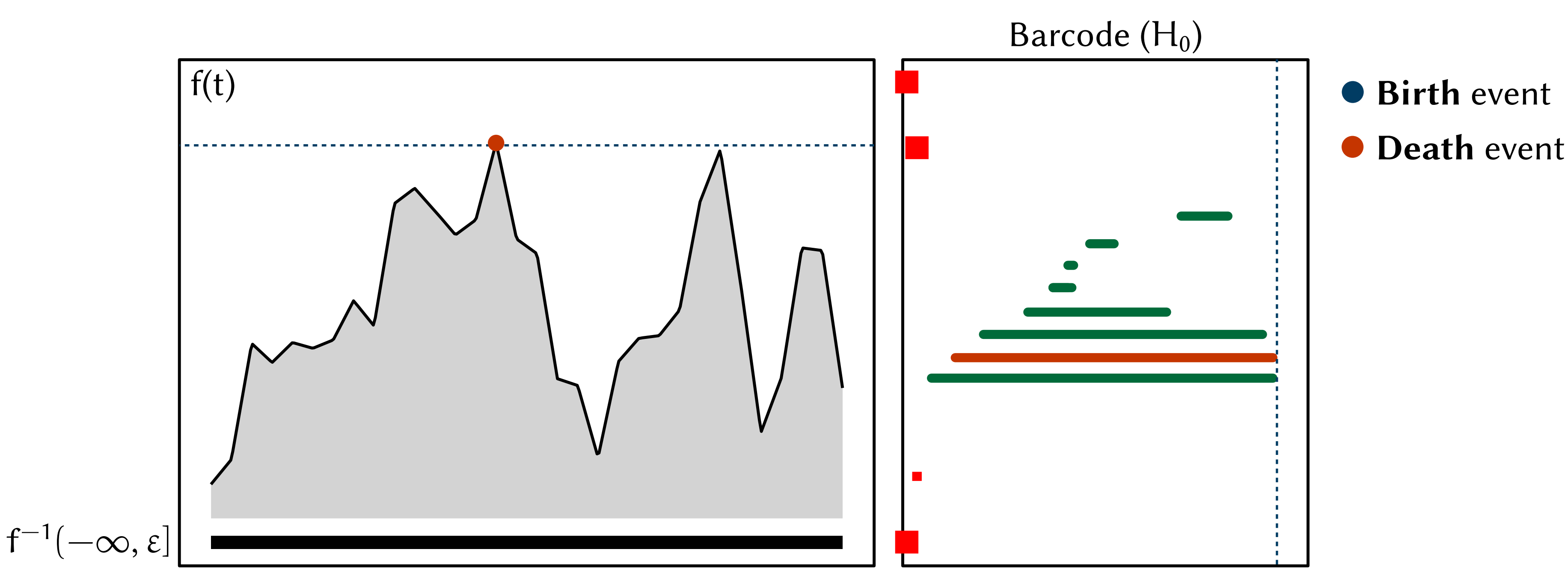


✓ Locality

✗ Little information per window

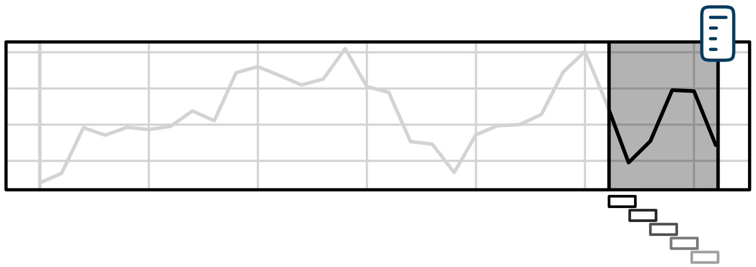
(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.



Local topological features for time series

Why not compute such topological features across **sliding windows**?

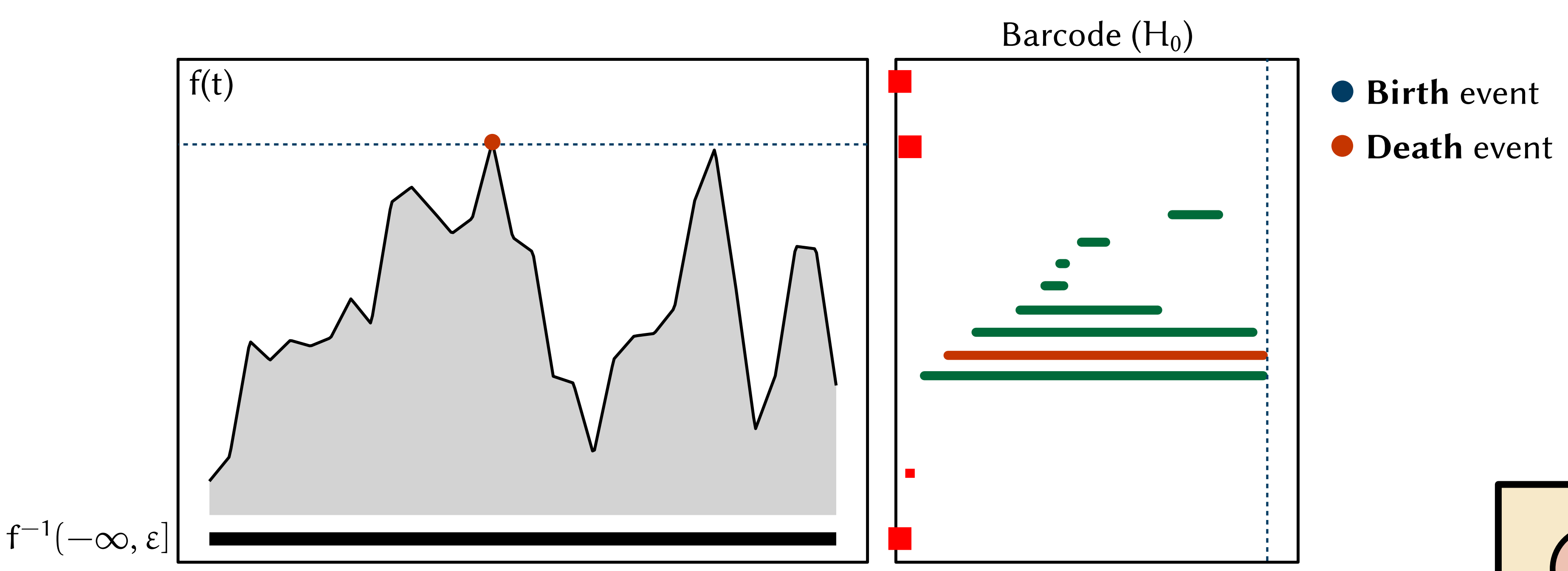


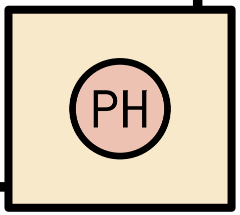
✓ Locality

✗ Little information per window

(as we need to chunk up the windows)

Instead, we extract topological features **directly** from the function graph.





Local topological features for time series

Why not compute such topological features across **sliding windows**?

✓ Locality

✗ Little information per window

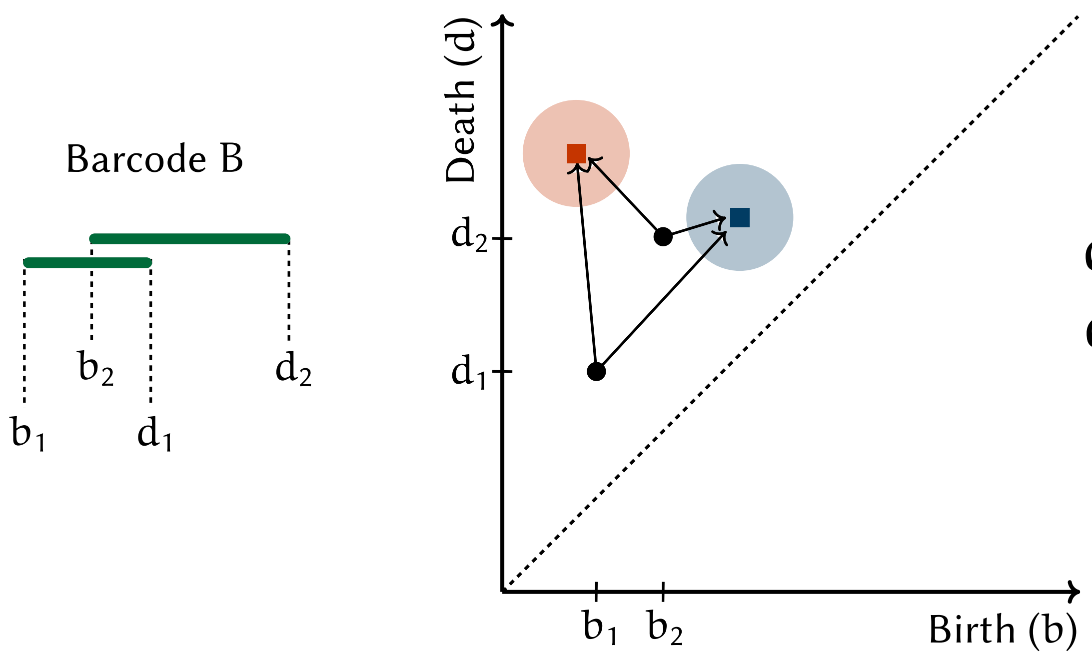
(as we need to chunk up the windows)

We will only consider 0-dimensional (**connectivity**) features in this work!

Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



$$\left.\begin{aligned} c_1 &= \textstyle\sum_{p \in B} s_{\theta_1}(p) \\ c_2 &= \textstyle\sum_{p \in B} s_{\theta_2}(p) \end{aligned}\right\} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

In general, we have differentiable map $B \mapsto \mathrm{vec}_\Theta(B)$; here: $\Theta = (\theta_1, \theta_2)$

e.g., $s_\theta(b, d) = \exp(-(\sigma_b(\mu_b - b)^2 + \sigma_d(\mu_d - d)^2))$, $\theta = (\mu, \sigma)$

vec

Vectorizing persistence barcodes

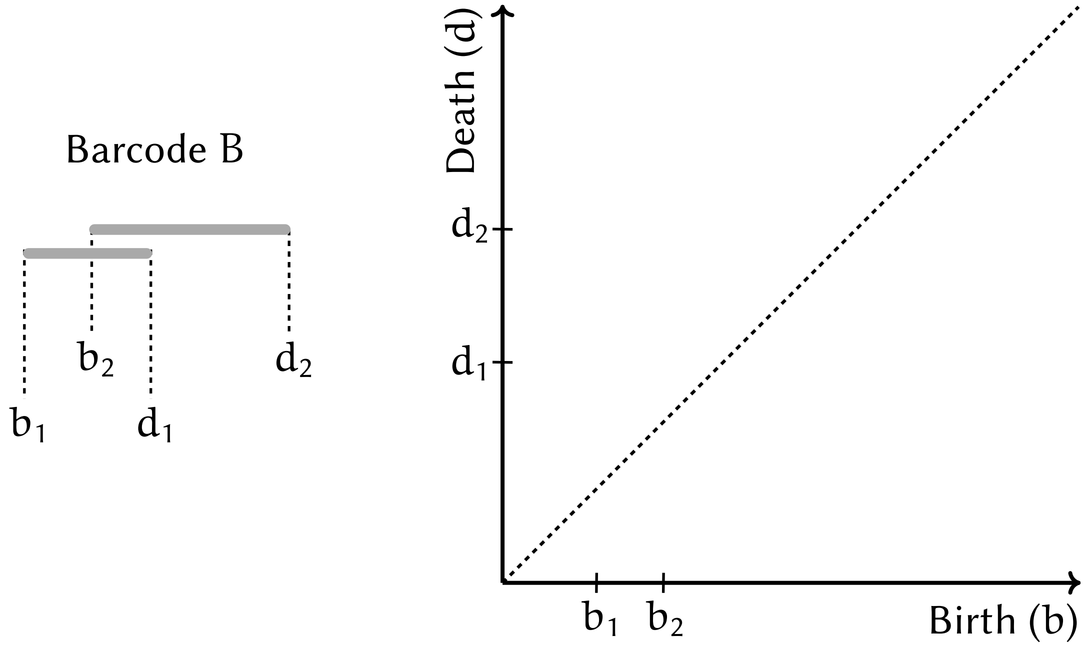
Barcodes are **multi-sets** of (birth, death) tuples.

To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]

Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

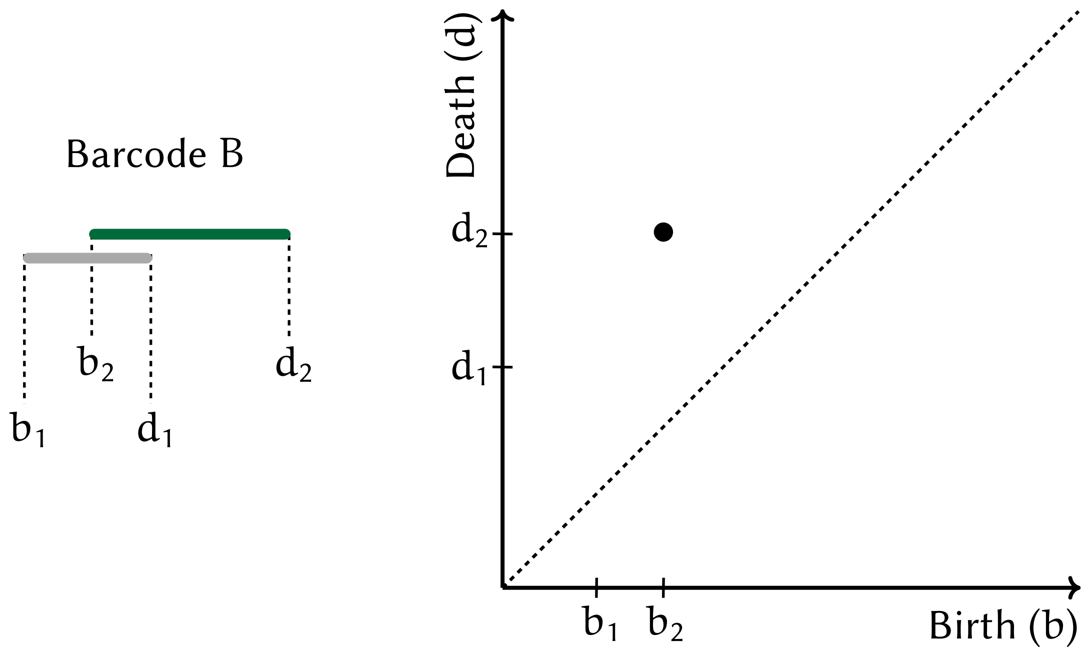
To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

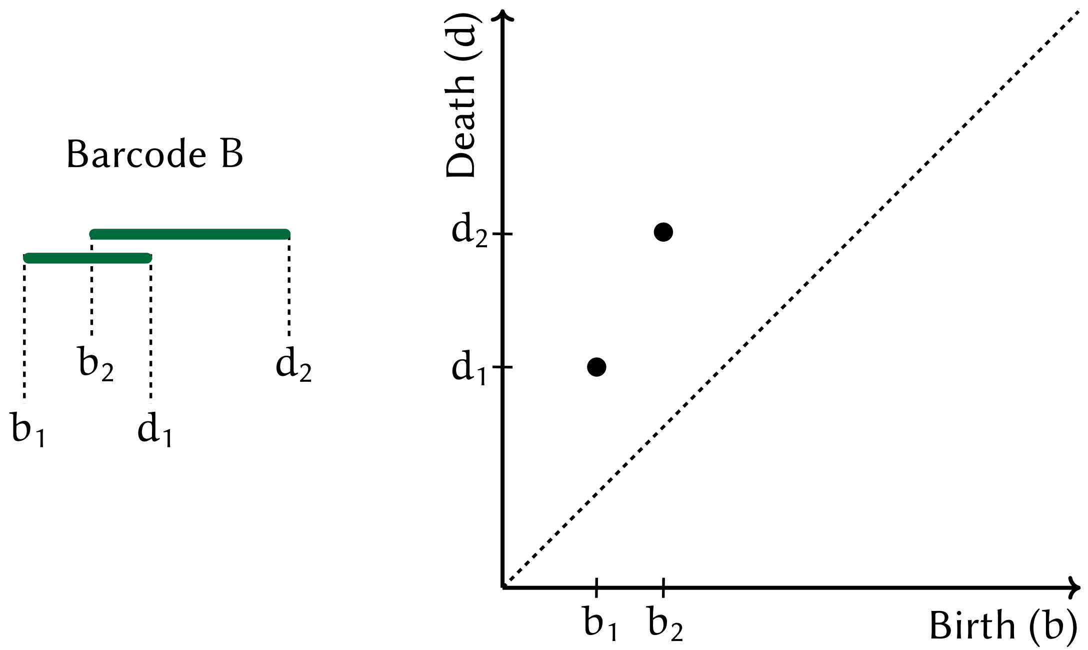
To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

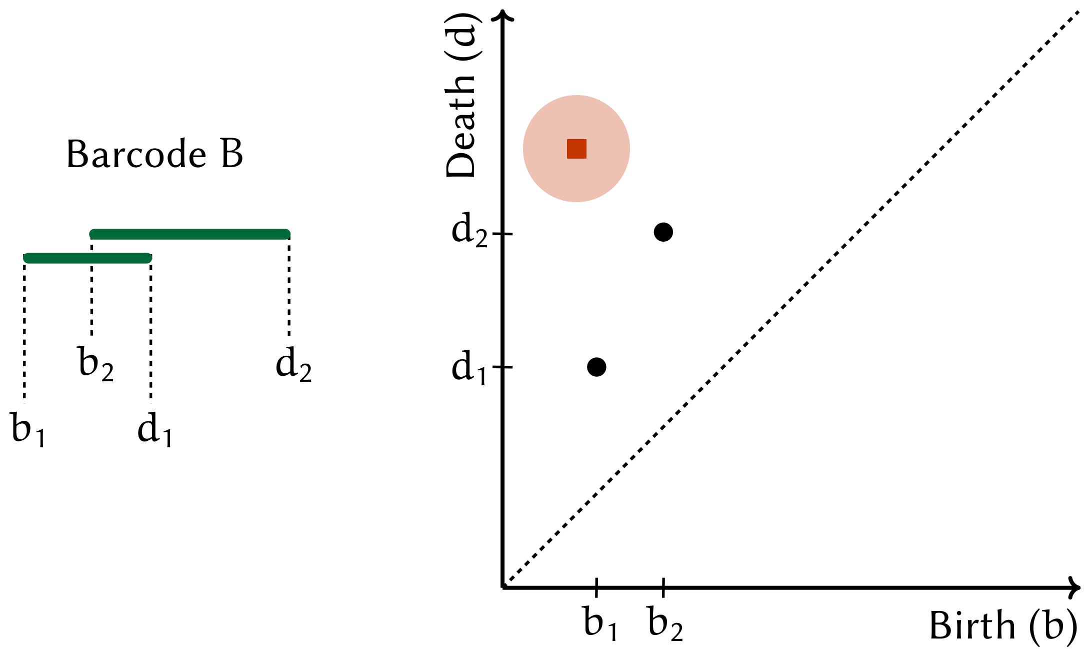
To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

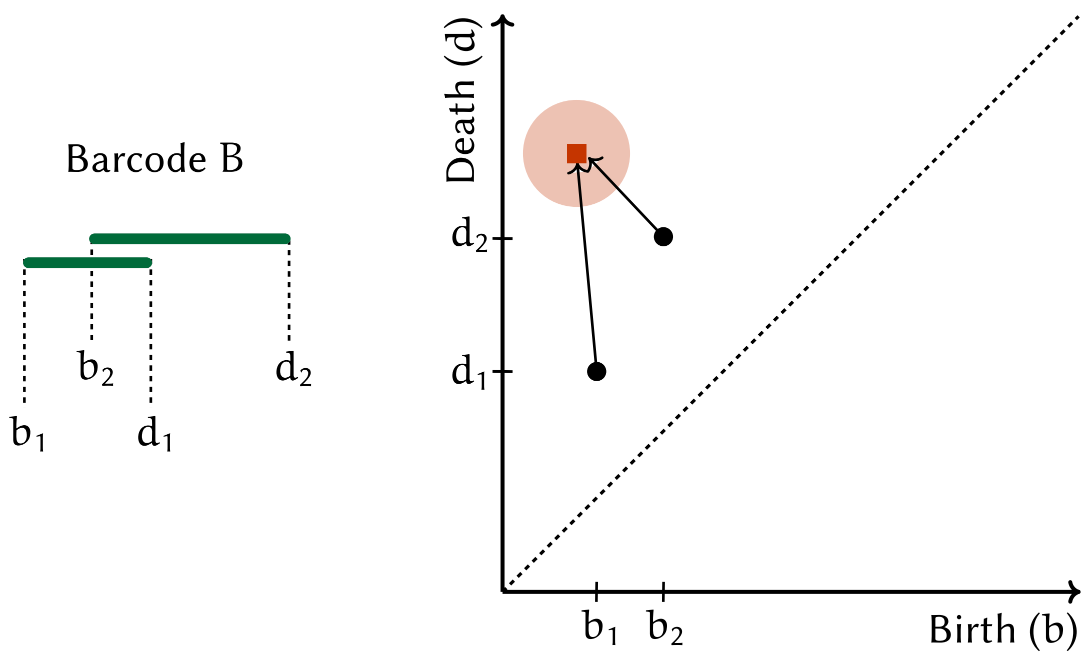
To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



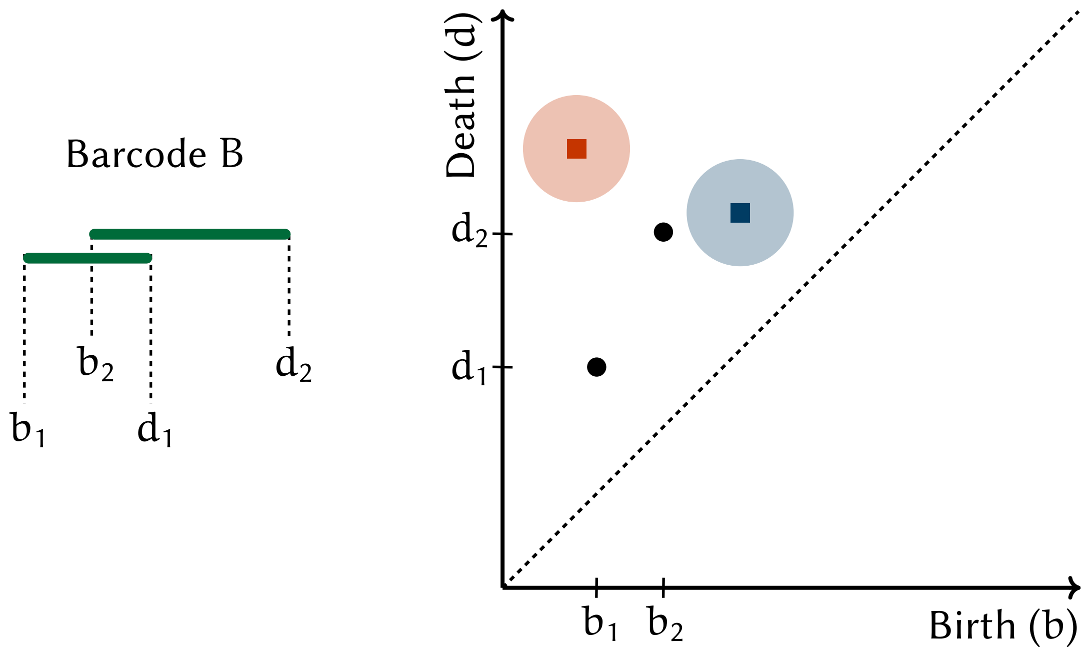
$$c_1 = \sum_{p \in B} s_{\theta_1}(p)$$

e.g., $s_\theta(b, d) = \exp(-(\sigma_b(\mu_b - b)^2 + \sigma_d(\mu_d - d)^2)), \theta = (\mu, \sigma)$

Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



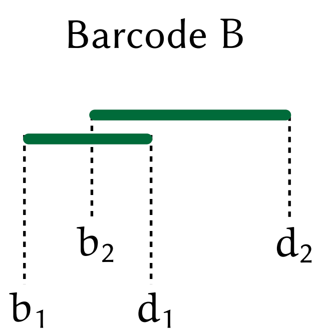
$$c_1 = \sum_{p \in B} s_{\theta_1}(p)$$

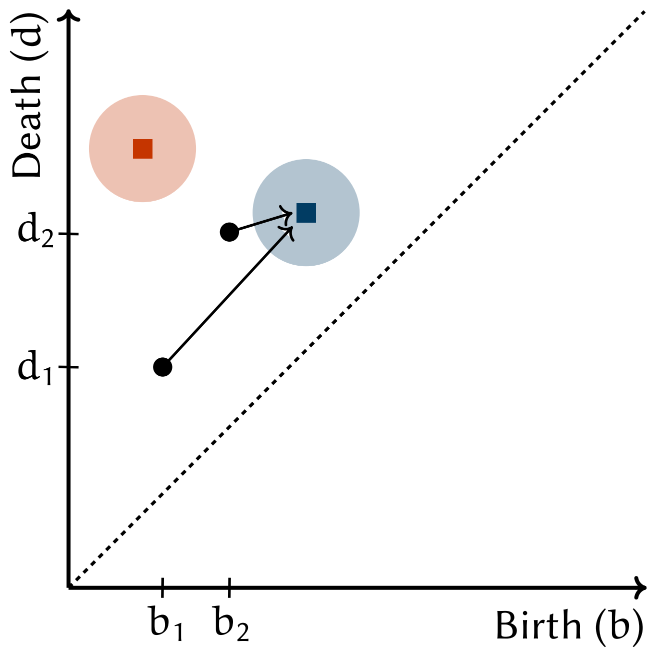
e.g., $s_\theta(b, d) = \exp(-(\sigma_b(\mu_b - b)^2 + \sigma_d(\mu_d - d)^2))$, $\theta = (\mu, \sigma)$

Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]





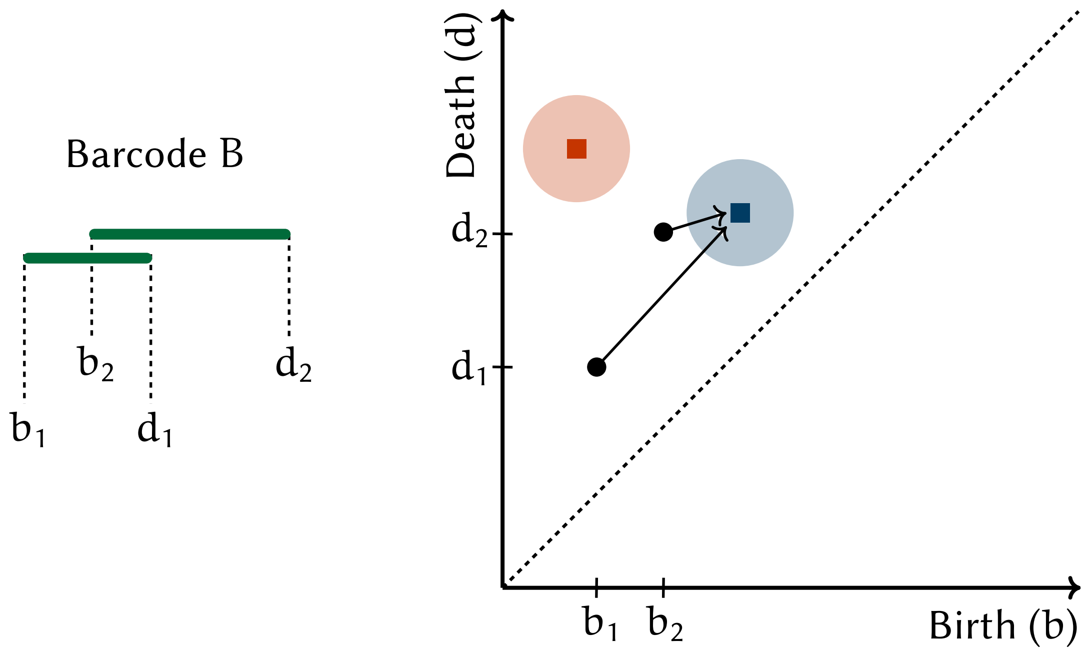
$$\left.\begin{aligned} c_1 &= \sum\nolimits_{p\in B} s_{\theta_1}(p) \\ c_2 &= \sum\nolimits_{p\in B} s_{\theta_2}(p) \end{aligned}\right\} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

e.g., $s_\theta(b, d) = \exp(-(\sigma_b(\mu_b - b)^2 + \sigma_d(\mu_d - d)^2))$, $\theta = (\mu, \sigma)$

Vectorizing persistence barcodes

Barcodes are **multi-sets** of (birth, death) tuples.

To **vectorize** the multi-set we choose an approach from [Hofer et al., 2019]



$$\left.\begin{aligned} c_1 &= \textstyle\sum_{p\in B} s_{\theta_1}(p) \\ c_2 &= \textstyle\sum_{p\in B} s_{\theta_2}(p) \end{aligned}\right\} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

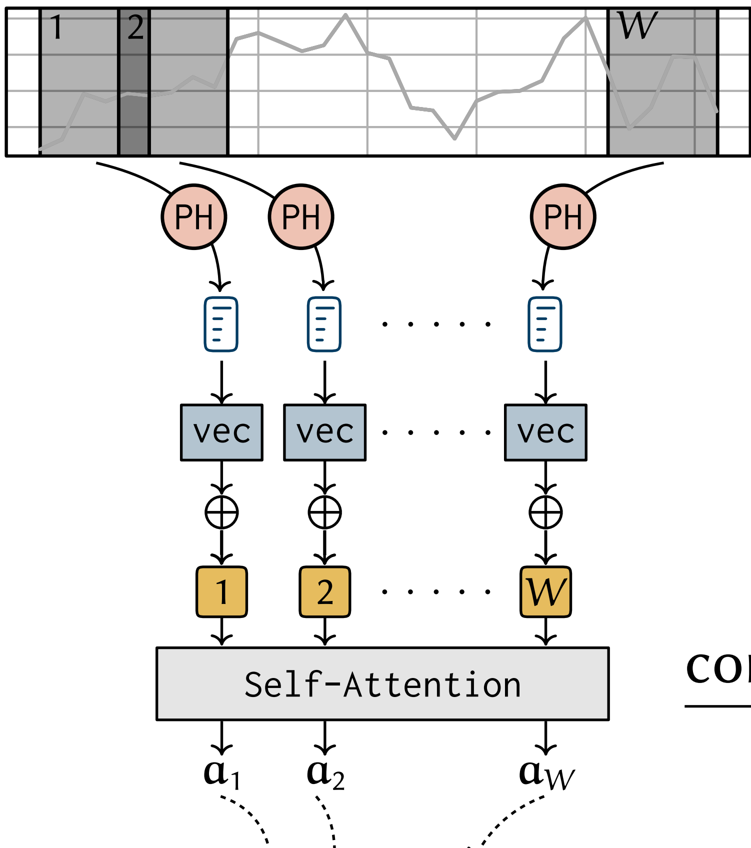
In general, we have differentiable map $B \mapsto \mathrm{vec}_\Theta(B)$; here: $\Theta = (\theta_1, \theta_2)$

e.g., $s_\theta(b, d) = \exp(-(\sigma_b(\mu_b - b)^2 + \sigma_d(\mu_d - d)^2)), \theta = (\mu, \sigma)$

vec

Attending to local topological features

We aim to allow attendening to **local** topological features.



$$\boldsymbol{v} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_W)^\top \in \mathbb{R}^{We}$$

- PH: Persistent homology of time series function graph
- vec: Barcode vectorization (into $\mathbb{R}^e$)
- Positional encoding
- Multi-head attention [Vaswani et al., 2016]

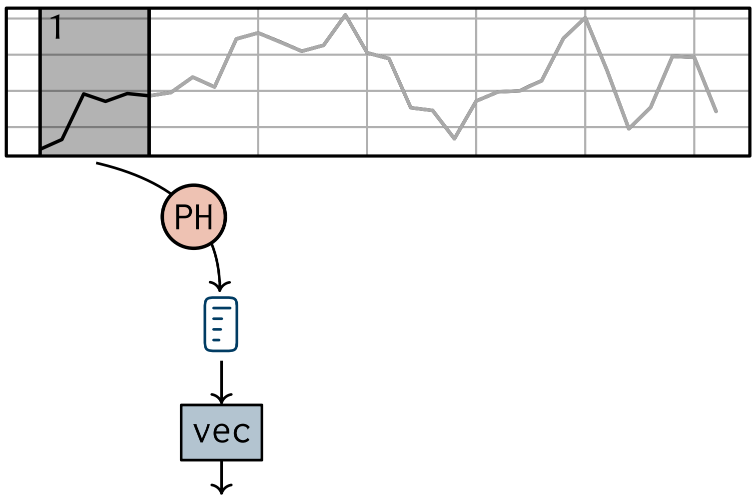
We will use the representation $\boldsymbol{v}$ as a complementary signal to a forecasting model

Topological Attention (TAN)

TAN

Attending to local topological features

We aim to allow attendening to **local** topological features.

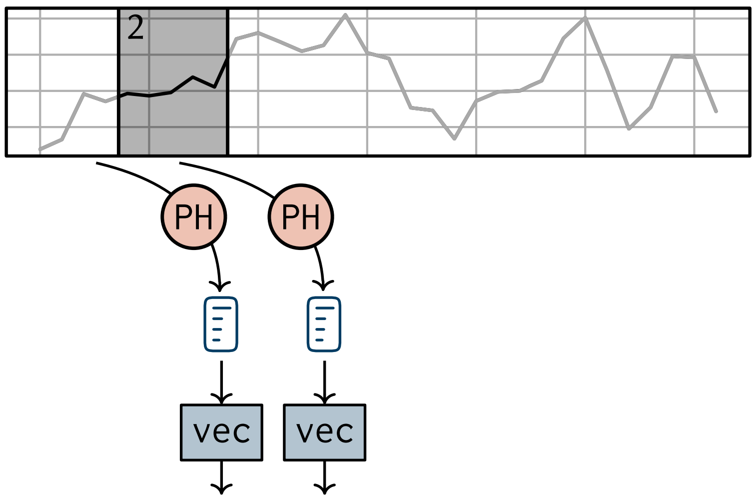


PH Persistent homology of time series function graph

vec Barcode vectorization (into $\mathbb{R}^e$)

Attending to local topological features

We aim to allow attendening to **local** topological features.

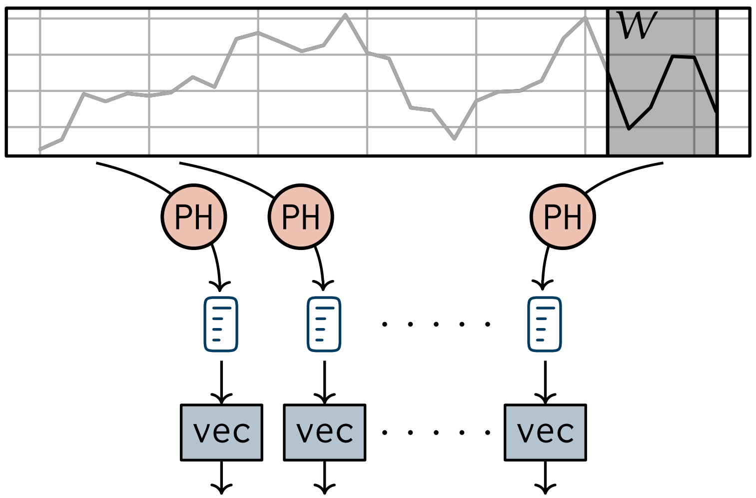


PH Persistent homology of time series function graph

vec Barcode vectorization (into $\mathbb{R}^e$)

Attending to local topological features

We aim to allow attendening to **local** topological features.

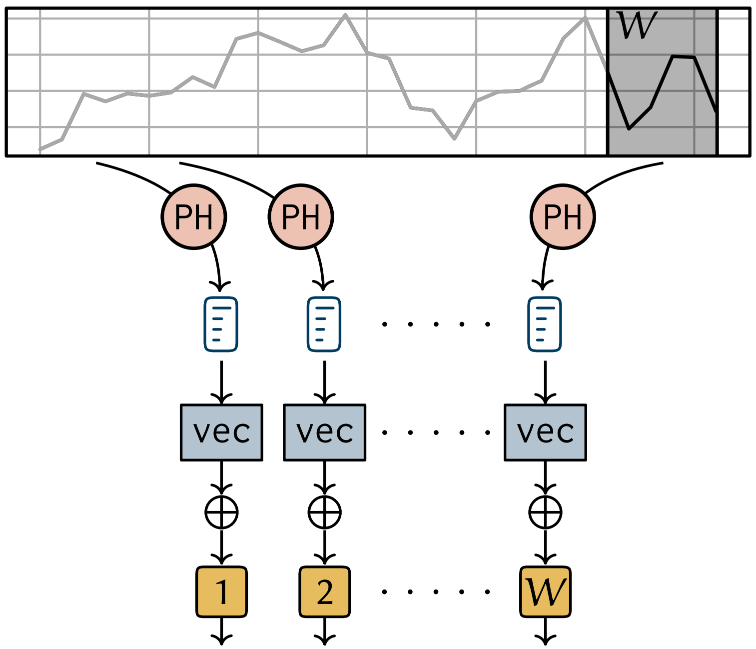


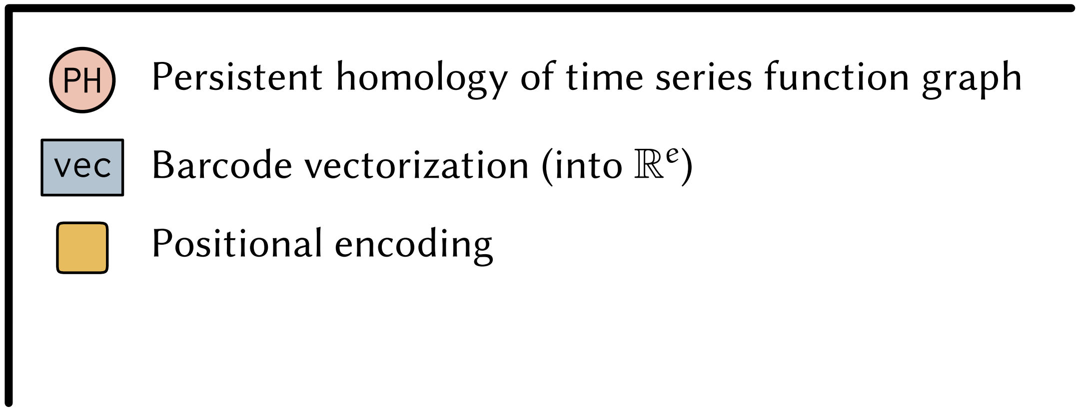
PH Persistent homology of time series function graph

vec Barcode vectorization (into $\mathbb{R}^e$)

Attending to local topological features

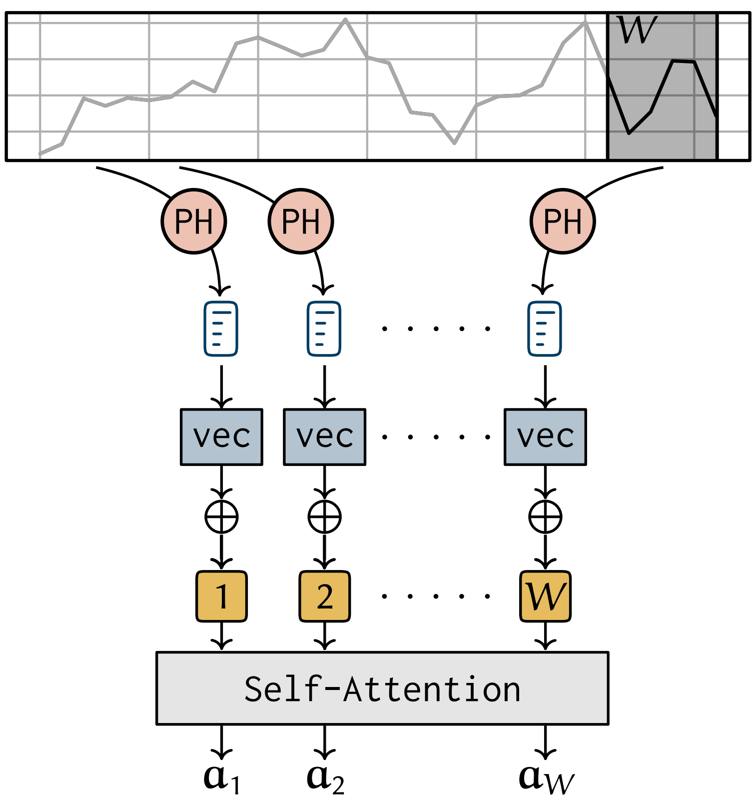
We aim to allow attendening to **local** topological features.





Attending to local topological features

We aim to allow attendening to **local** topological features.

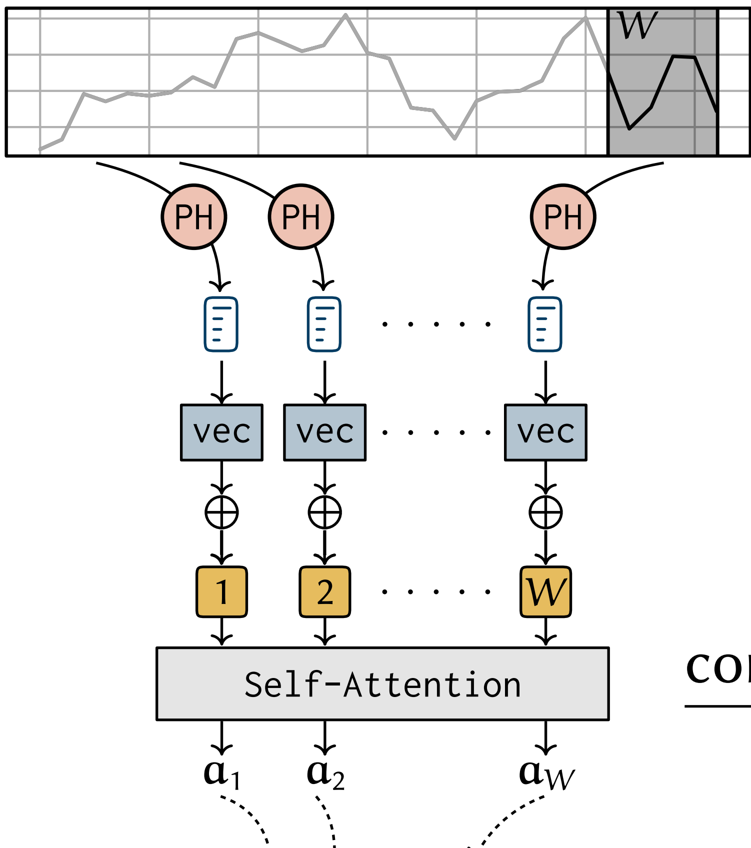


$$\boldsymbol{v} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_W)^\top \in \mathbb{R}^{We}$$

- PH: Persistent homology of time series function graph
- vec: Barcode vectorization (into $\mathbb{R}^e$)
- Positional encoding
- Multi-head attention [Vaswani et al., 2016]

Attending to local topological features

We aim to allow attendening to **local** topological features.



$$\boldsymbol{v} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_W)^\top \in \mathbb{R}^{We}$$

- PH — Persistent homology of time series function graph
- vec — Barcode vectorization (into $\mathbb{R}^e$)
- Positional encoding
- Multi-head attention [Vaswani et al., 2016]

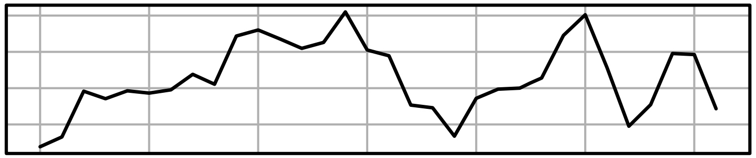
We will use the representation $\boldsymbol{v}$ as a complementary signal to a forecasting model

Topological Attention (TAN)

TAN

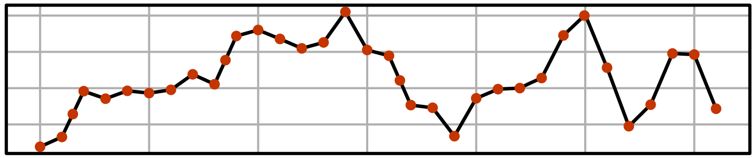
An incarnation of TAN

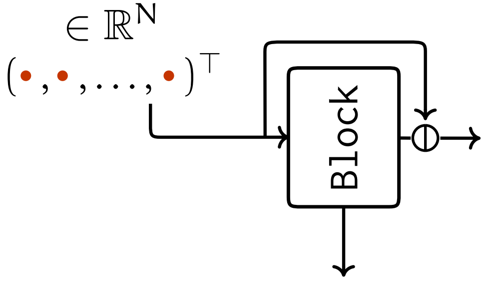
TAN + (generic) N-BEATS [Oreshkin et al., 2019]



An incarnation of TAN

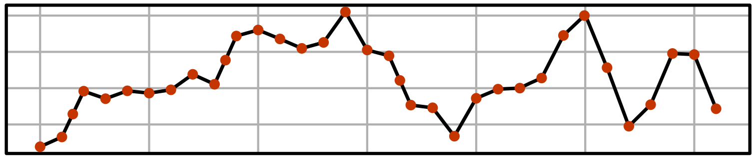
TAN + (generic) N-BEATS [Oreshkin et al., 2019]

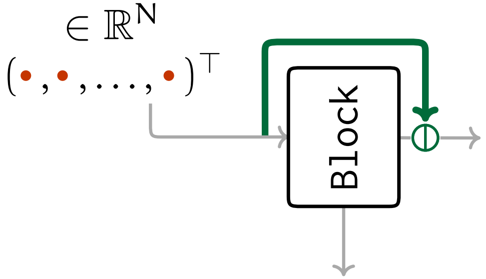




An incarnation of TAN

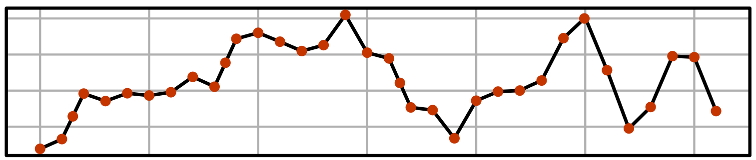
TAN + (generic) N-BEATS [Oreshkin et al., 2019]

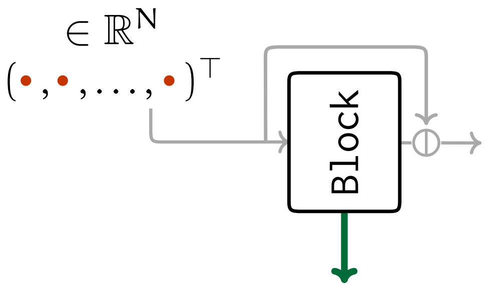




An incarnation of TAN

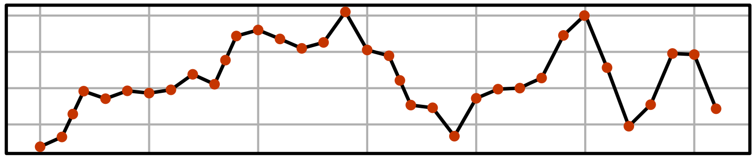
TAN + (generic) N-BEATS [Oreshkin et al., 2019]

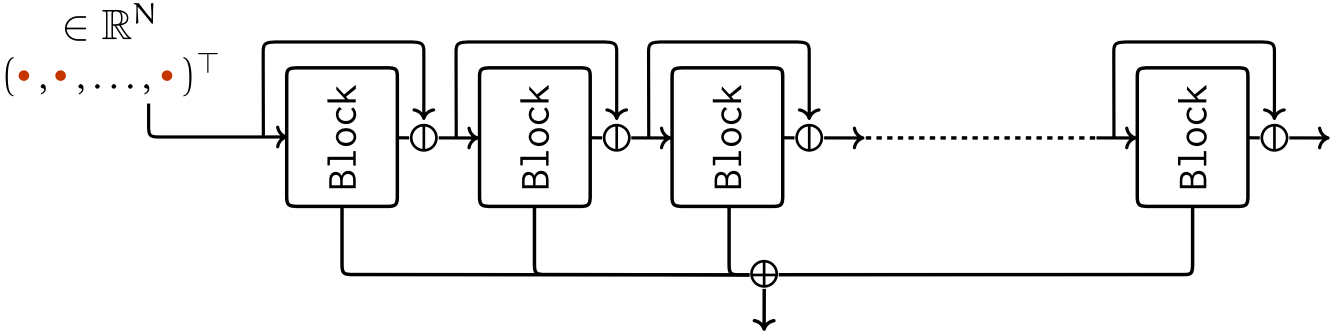




An incarnation of TAN

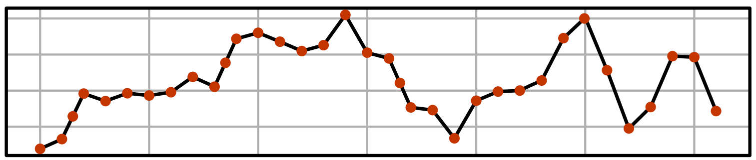
TAN + (generic) N-BEATS [Oreshkin et al., 2019]

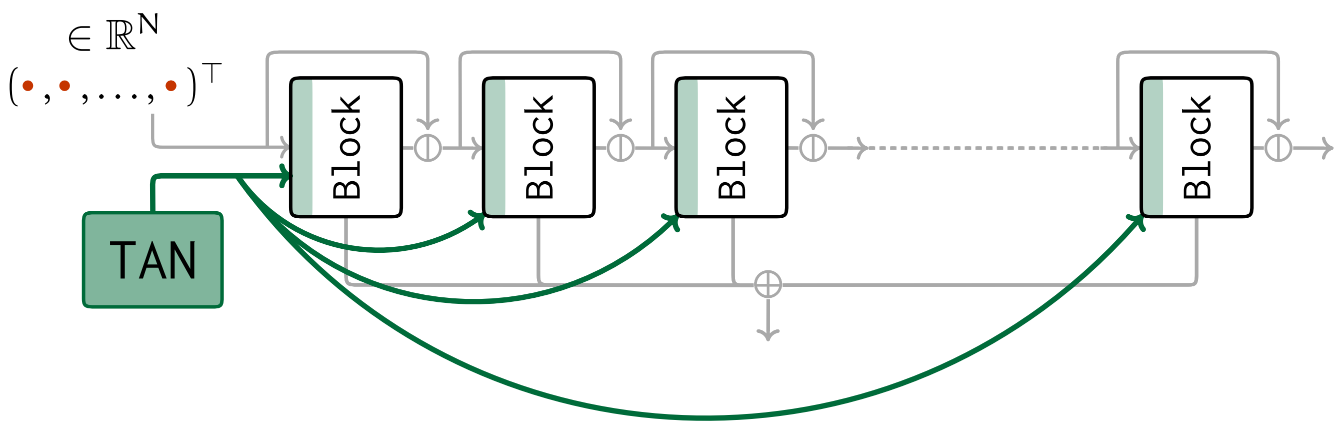




An incarnation of TAN

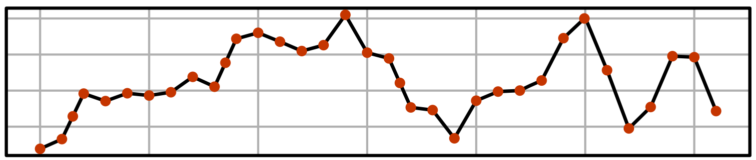
TAN + (generic) N-BEATS [Oreshkin et al., 2019]

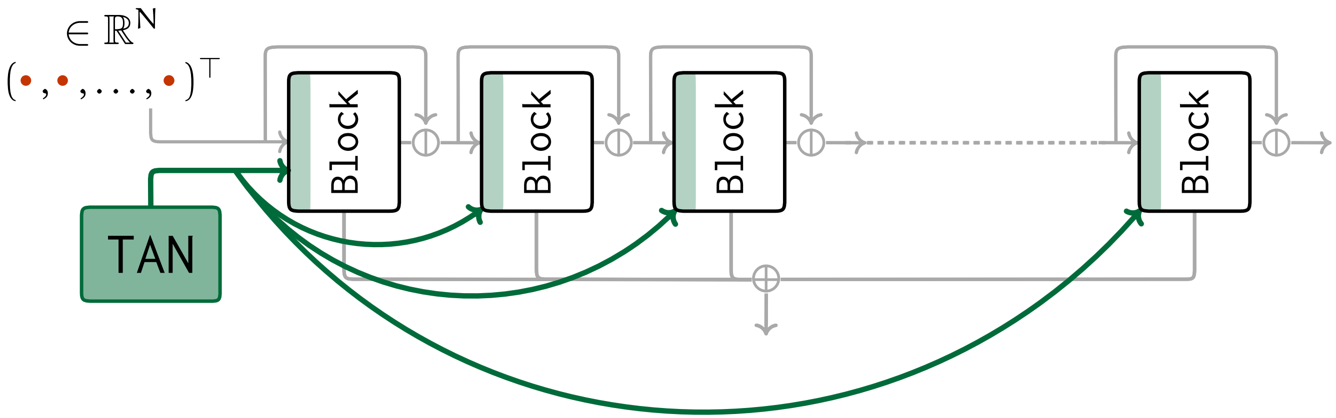




An incarnation of TAN

TAN + (generic) N-BEATS [Oreshkin et al., 2019]





Setup:

- ▷ We follow the **ensemble** approach of [Oreshkin et al., 2019]
- ▷ Different (1) random seeds, (2) historical time horizons and (3) losses
- ▷ Forecast = **median** across ensemble forecasts

Experiments

Dataset: M4 competition data [Makridakis et al., 2018]

Experiments

Dataset: M4 competition data [Makridakis et al., 2018]

- Corpus of 100k time series from various subgroups (yearly, monthly, ...)

Experiments

Dataset: *M4* competition data [Makridakis et al., 2018]

- ▷ Corpus of 100k time series from various subgroups (yearly, monthly, ...)
- ▷ Different forecasting horizons (H) / subgroup

Experiments

Dataset: M4 competition data [Makridakis et al., 2018]

- Corpus of 100k time series from various subgroups (yearly, monthly, ...)
- Different forecasting horizons (H) / subgroup
- Fix evaluation protocol & scores (sMAPE, MASE, OWA)

Overall performance comparison (across all 100,000 time series):

Method	sMAPE $\downarrow$	OWA $\downarrow$
N-BEATS	11.324	0.814

Overall performance comparison (across all 100,000 time series):

Method	sMAPE ↓	OWA ↓
N-BEATS	11.324	0.814
N-BEATS + **TAN**	**11.291**	**0.811**

Overall performance comparison (across all 100,000 time series):

Method	sMAPE ↓	OWA ↓
N-BEATS	11.324	0.814
N-BEATS + **TAN**	**11.291**	**0.811**
†Winner M4	11.374	0.821
†Benchmark	12.555	0.898
†Naive2	13.564	1.000

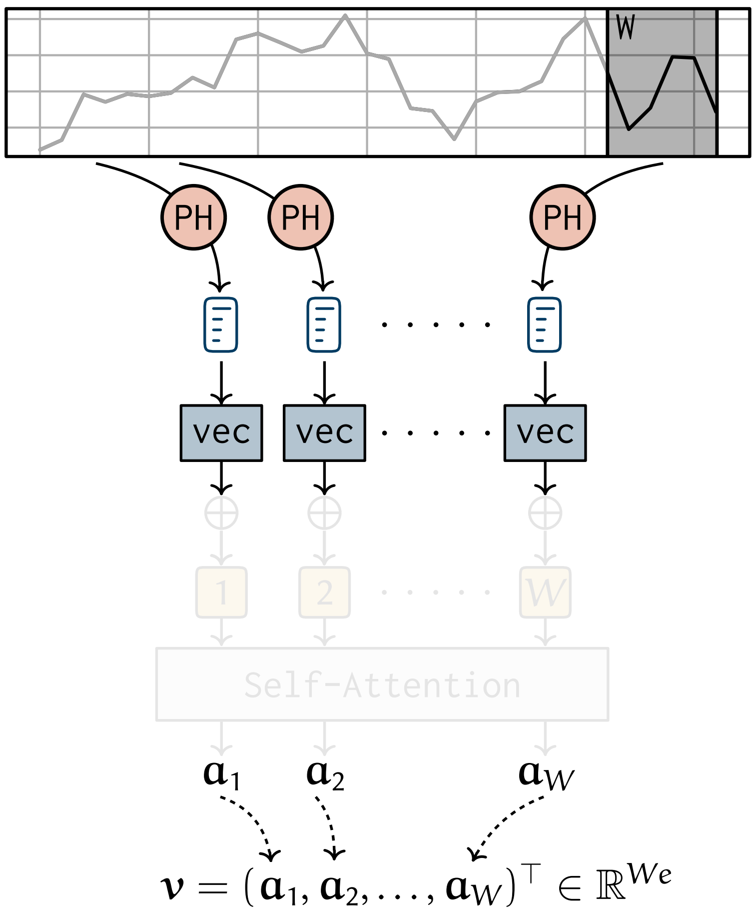
see paper for final results

† from [Makridakis et al., 2018]

Next, we successively deactivate **TAN** parts (on a smaller ensemble):

Method	sMAPE$\downarrow$	OWA$\downarrow$
N-BEATS	11.488	0.827

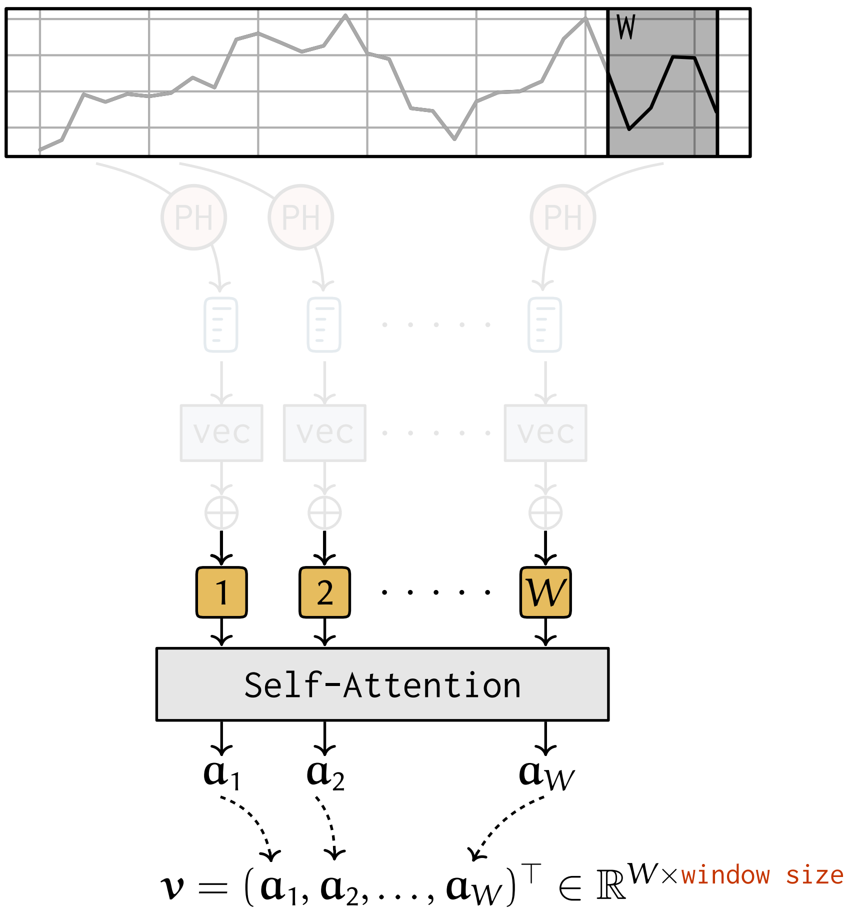
Next, we successively deactivate **TAN** parts (on a smaller ensemble):



Method	sMAPE ↓	OWA ↓
N-BEATS	11.488	0.827
+ Top	11.505	0.920

Experiments – Ablation

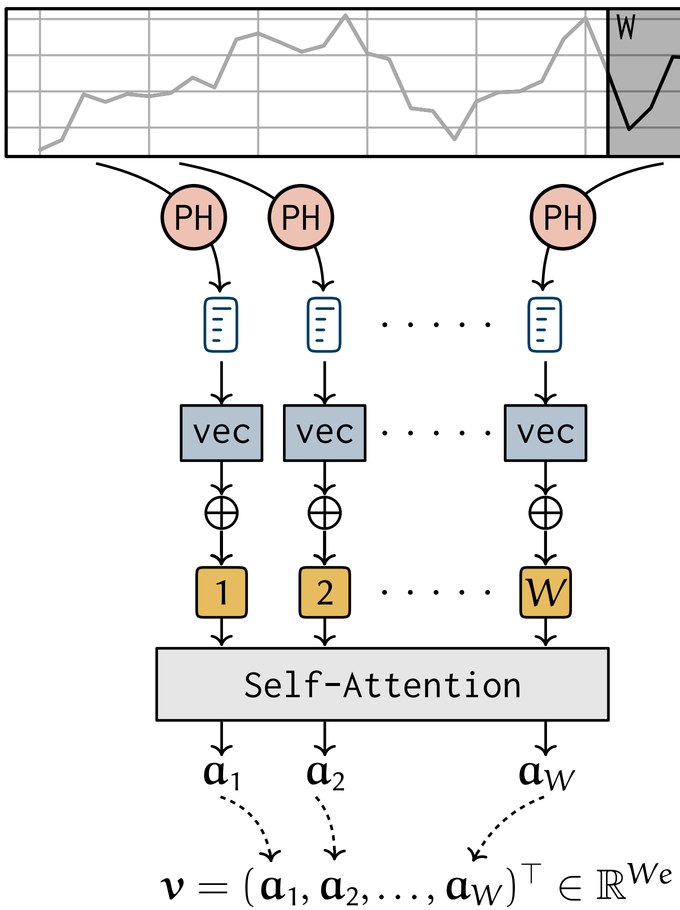
Next, we successively deactivate **TAN** parts (on a smaller ensemble):



Method	sMAPE ↓	OWA ↓
N-BEATS	11.488	0.827
+ Top	11.505	0.920
+ Attention	11.492	0.826

Experiments – Ablation

Next, we successively deactivate **TAN** parts (on a smaller ensemble):



Method	sMAPE ↓	OWA ↓
N-BEATS	11.488	0.827
+ Top	11.505	0.920
+ Attention	11.492	0.826
+ TAN	**11.466**	**0.824**

see paper for final results

Experiments – Runtime

Runtime of 0-dim. persistent homology as a function of window size m:

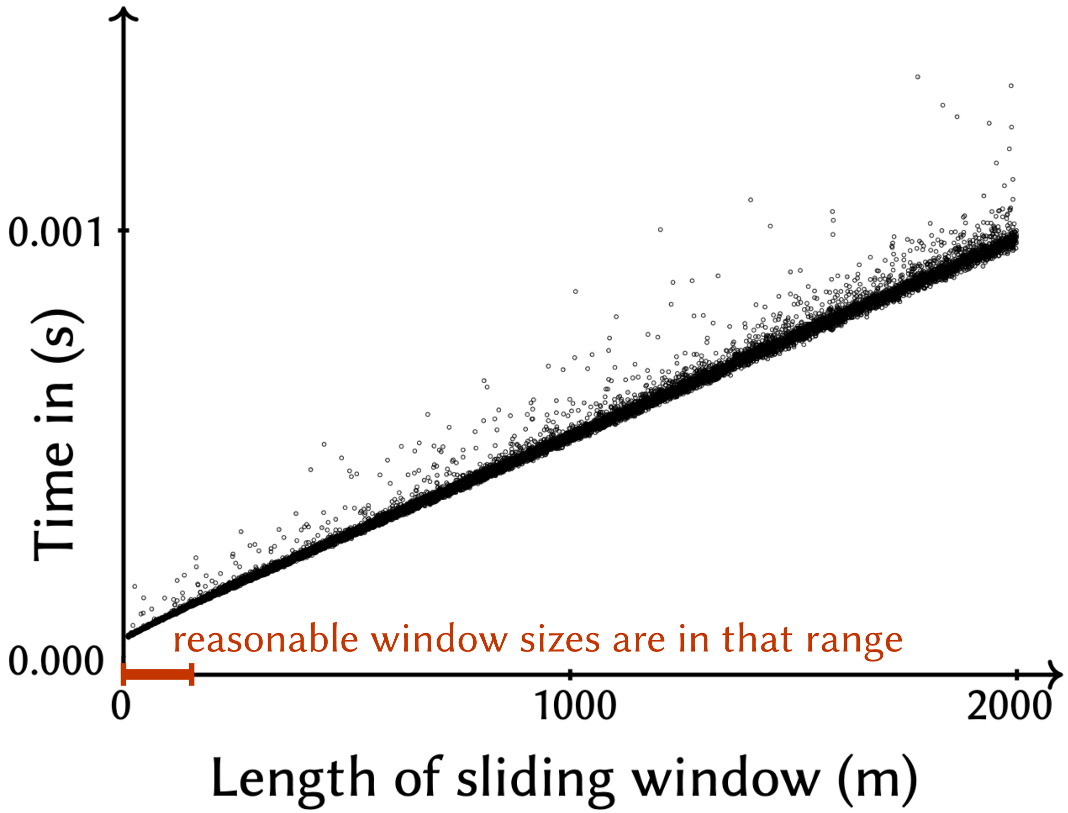
Complexity: $\mathcal{O}(m\alpha^{-1}(m))$

α^{-1} = inverse Ackerman function

m = window size

Experiments – Runtime

Runtime of 0-dim. persistent homology as a function of window size m:



Complexity: $\mathcal{O}(m\alpha^{-1}(m))$

α^{-1} = inverse Ackerman function

m = window size

Conclusion

We propose an approach ...

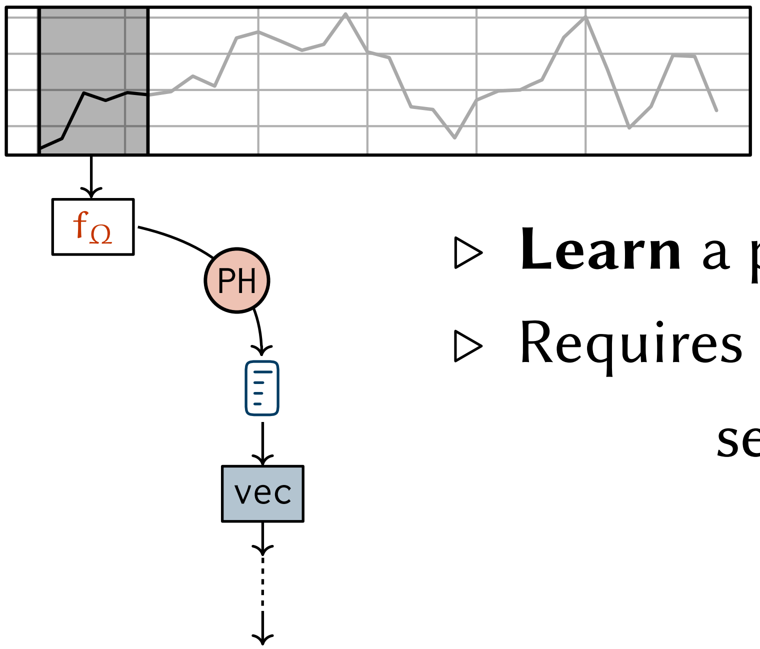
- ▷ . . . to attend to local topological time series features
- ▷ . . . that is easy to integrate at moderate computational overhead
- ▷ . . . that supplies complementary information beyond raw observations

Conclusion

We propose an approach ...

- ▷ . . . to attend to local topological time series features
- ▷ . . . that is easy to integrate at moderate computational overhead
- ▷ . . . that supplies complementary information beyond raw observations

Possible future direction



- ▷ **Learn** a parameterized "filter" f_Ω for PH
- ▷ Requires to backprop through PH computation

see [Hofer et al., 2020], [Carrière et al.,2021]

Thank You!

Source code & Slides are available at

https://github.com/plus-rkwitt/TAN