Study Note on *Topological machine Learning for Multivariate Time Series*

PingYao Feng

2022.5.8

Form article *Topological Machine Learning for Multivariate Time Series*, by Chengyuan Wu and Carol Anne Hargreaves, published on *JOURNAL OF EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE,* here: https://arxiv.org/abs/1911.12082

Table of Contents

"hello, sliding window algorithm!"

Sliding window algorithm: Find the floating mean value, just look like a window

```
time_series=[1,5,0,2,8,1,7,9,0,0,1,2,5]
```

```
time_series = 1×13
     1     5     0     2     8     1     7     9     0     0     1     2     5
```

```
s=3;a="";
for i=1:length(time_series)-2
    for u=1:i
        a=strcat(a,"  ");
    end
    fprintf('%s %d,%d,%d\n',a,time_series(i),time_series(i+1),time_series(i+2))
    a="";
end
```

```
   1,5,0
     5,0,2
       0,2,8
         2,8,1
           8,1,7
             1,7,9
               7,9,0
                 9,0,0
                   0,0,1
                     0,1,2
                       1,2,5
```

Sliding window for multivariate time series: Converting the multivariate time series into a point cloud

Suppose there are d 1-dimensional time series:

```
% here d=4
x=randi([10,99],[4,11])
```

```
x = 4×11
    44    57    61    24    24    72    30    58    19    83    33
    61    80    52    81    64    77    92    99    96    88    82
    16    94    11    38    33    50    23    17    10    17    48
    14    21    40    57    68    17    84    49    79    45    91
```

Fix a sliding window of size w. For each time define a point $x(t_n) = (x_n^1, \ldots, x_n^d) \in \mathbb{R}^d$, thus we can obtain a point cloud $X_n = (x(t_{1+s(n-1)}), x(t_{2+s(n-1)}), \ldots, x(t_{w+s(n-1)}))$, where s denote the stride of the sliding window.

```
% here s=2, w=3
a="";
for i=1:5
    for j=1:4
        b=sprintf("%d, %d, %d",x(j,2*i-1),x(j,2*i),x(j,2*i+1));
        fprintf("%s%s\n",a,b);
    end
```

```
    a=strcat(a,"        ");
end
```

```
44, 57, 61
61, 80, 52
16, 94, 11
14, 21, 40
        61, 24, 24
        52, 81, 64
        11, 38, 33
        40, 57, 68
                24, 72, 30
                64, 77, 92
                33, 50, 23
                68, 17, 84
                        30, 58, 19
                        92, 99, 96
                        23, 17, 10
                        84, 49, 79
                                19, 83, 33
                                96, 88, 82
                                10, 17, 48
                                79, 45, 91
```

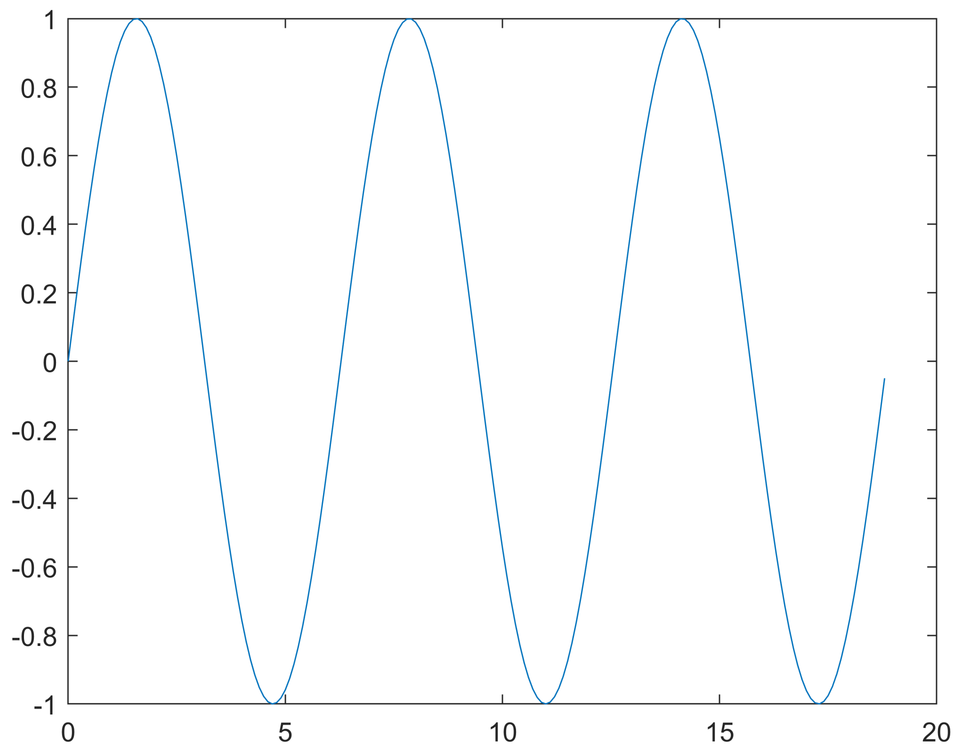
Motivation for SWE (sliding window embedding):

From *Cyclicality, Periodicity and the Topology of Time Series,* written by Pawel Dlotko, Wanling Qiu, and Simon Rudkin, in 2019; here: https://arxiv.org/abs/1905.12118

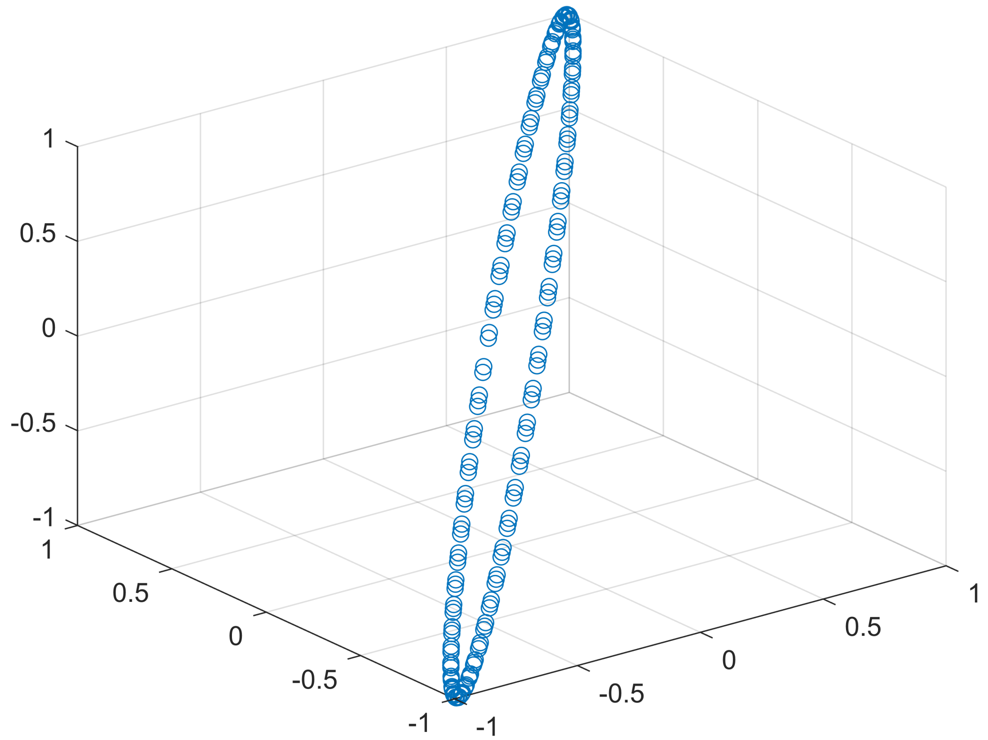
To overcome some drawbacks in traditional definition of 'periodic'. Just like encapsulation of the local behaviour of the time series f.

If f is continous and periodic with a period p, there exist k such that $p - (x_k - x_0)$ is minimal, detecting if the embedding point is 'closed' to each other.

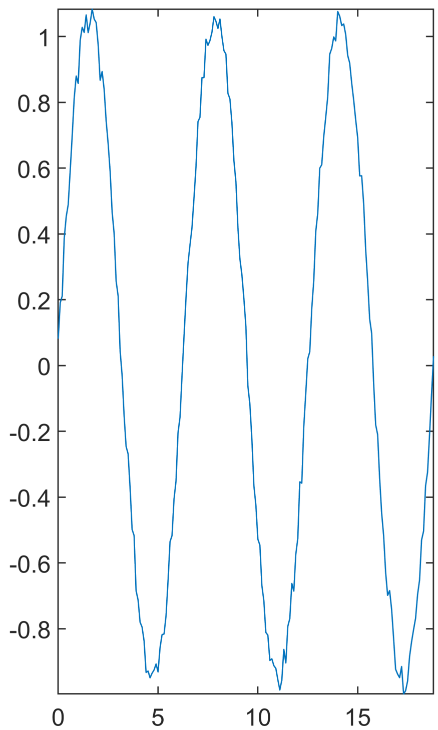
```
clear;close all;clc
t=0:.1:6*pi;
y=sin(t);l=length(t);
plot(t,y)
```

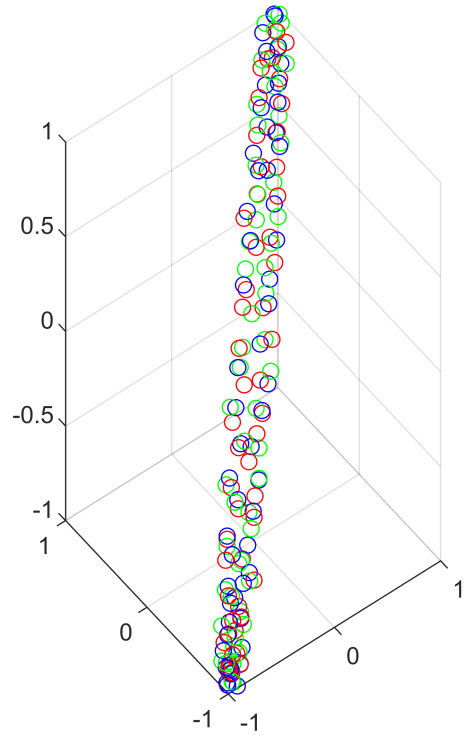


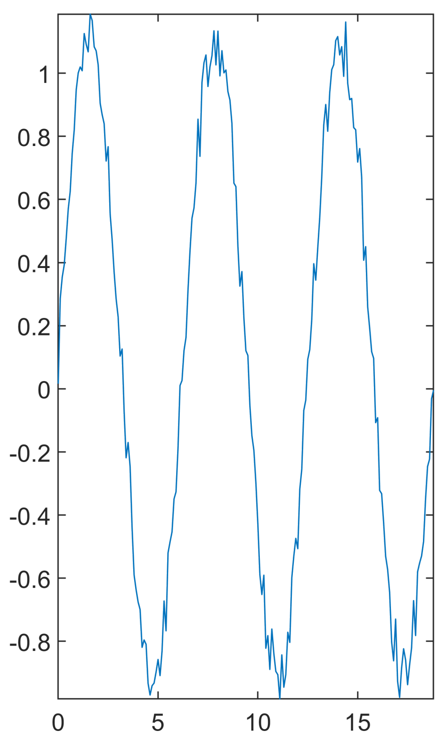

```
sw_emb=zeros(3,l-2);
sw_emb(1,:)=y(1:(end-2));
sw_emb(2,:)=y(2:(end-1));
sw_emb(3,:)=y(3:(end));
scatter3(sw_emb(1,:),sw_emb(2,:),sw_emb(3,:))
```

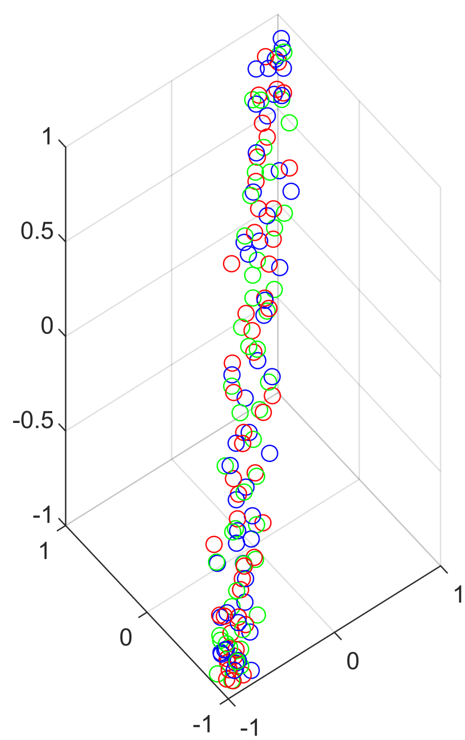


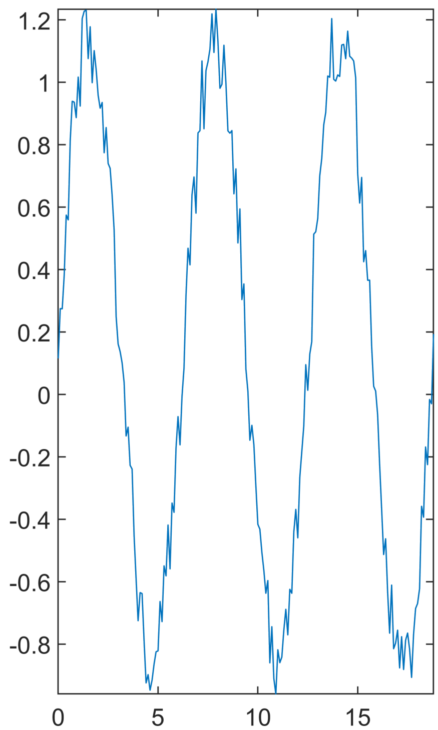

```
for i=linspace(0.1,0.3,3)
    y_ptbt=rand([1,l])*i;
    y_final=y+y_ptbt;
    sw_emb_ptbt(1,:)=y_final(1:(end-2));
    sw_emb_ptbt(2,:)=y_final(2:(end-1));
    sw_emb_ptbt(3,:)=y_final(3:(end));

    %create .csv file to find homology group
    file_name=convertStringsToChars(sprintf("ptbt_%.1f.txt",i));
    m=sw_emb_ptbt';
    var={'x','y','z'};
    my_table=table(m(:,1),m(:,2),m(:,3),'VariableNames',var);
    writetable(my_table,file_name);

    figure
    subplot(1,2,1)
    plot(t,y_final)
    axis tight;
    subplot(1,2,2)
    hold on;
    scatter3(sw_emb_ptbt(1,1:63),sw_emb_ptbt(2,1:63),sw_emb_ptbt(3,1:63),'MarkerEdgeColor','r')
    scatter3(sw_emb_ptbt(1,64:126),sw_emb_ptbt(2,64:126),sw_emb_ptbt(3,64:126),'MarkerEdgeColor
    scatter3(sw_emb_ptbt(1,127:end),sw_emb_ptbt(2,127:end),sw_emb_ptbt(3,127:end),'MarkerEdgeCo
    xlim([-1 1]);ylim([-1 1]);zlim([-1 1])
    view(3);grid on;
end
```

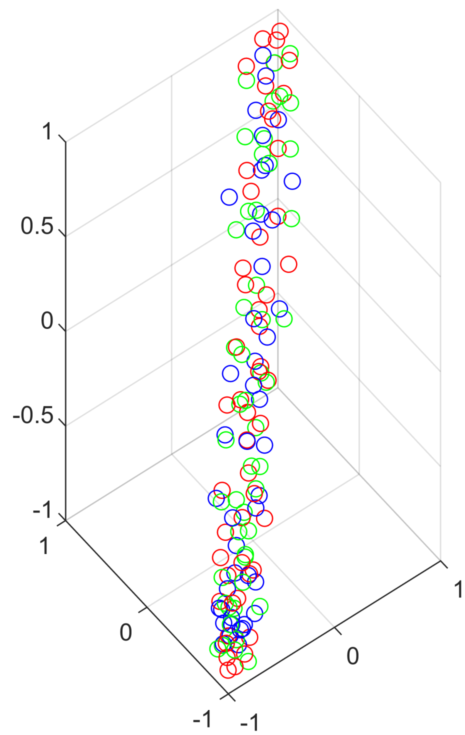

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
5
10
15


1
0.5
0
-0.5
-1
1
0
-1
-1
0
1


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
5
10
15


1
0.5
0
-0.5
-1
1
0
-1
-1
0
1





Symmetry-breaking and anchor-points

The symmetry of coordinate makes classical TDA unable to distinguish certain point cloud, for example, $X_1 = \{(0,0,0,0,0),(1,0,0,0,0)\}$, $X_2 = \{(0,0,0,0,0),(0,1,0,0,0)\}$. However, in real scene, different coordinant might indicate different features, so it's natural that we should distinguish the two point clouds.

Symmetry-breaking: $X' = \{x + v \mid x \in X\}$, where x, v belong to $\mathbb{R}^d$ and v is fixed vector.

Anchor points: Let $A = \{a_1, ..., a_n\}$ be a set of point in $\mathbb{R}^d$, call it anchor point.

For example, let $v = (0,1,2,3,4) \in \mathbb{R}^5$ to be the fixed vector, $A = \{(0,0,0,0,0)\}$ to be the anchor point. Then the above two point clouds become

$$Y_1 = X_1' \cup A = \{(0,1,2,3,4),(1,1,2,3,4),(0,0,0,0,0)\}$$

$$Y_2 = X_2' \cup A = \{(0,1,2,3,4),(0,2,2,3,4),(0,0,0,0,0)\}$$

Which are now different.

Wasserstein distance: Measure distance between persistence diagram

p-th Wasserstein distance between two persistence diagrams D_1 and D_2 is defined as

$$W_p(D_1, D_2) = (\inf_{\varphi:D_1 \to D_2} \sum_{x \in D_1} \|x - \varphi(x)\|_\infty^p)^{1/p}$$

where the infimum is taken over all bijections φ between D_1 and D_2. However, in real case, we can compute Wasserstein distance for which φ may not be a bijection.

There are more than one way to define distance between two persistence diagram, e.g the bottleneck distance is given by

$$W_\infty(D_1, D_2) = \sup_{\varphi:D_1 \to D_2} \sum_{x \in D_1} \|x - \varphi(x)\|_\infty$$

Which makes the distance of persistence diagram stable under perturbation.

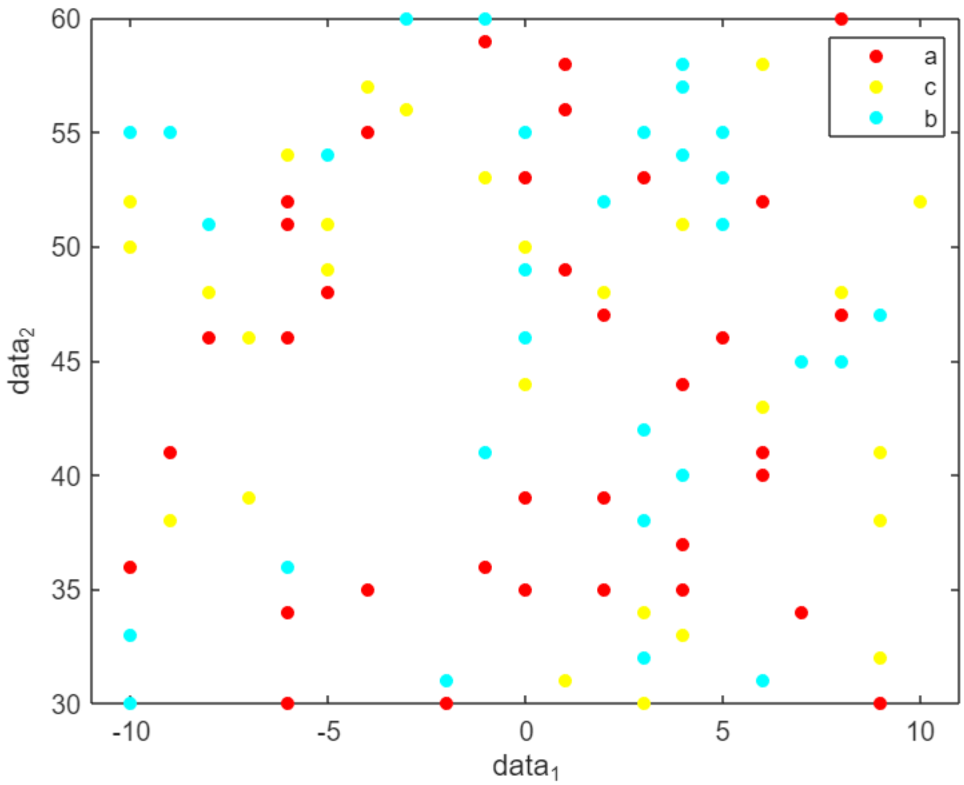
Classification model: partition of the space using indicator

kNN(k nearest neighbor) method: clasisfy to be the same class as the nearest known example, developed by Evelyn Fix and Joseph Hodges in 1951.

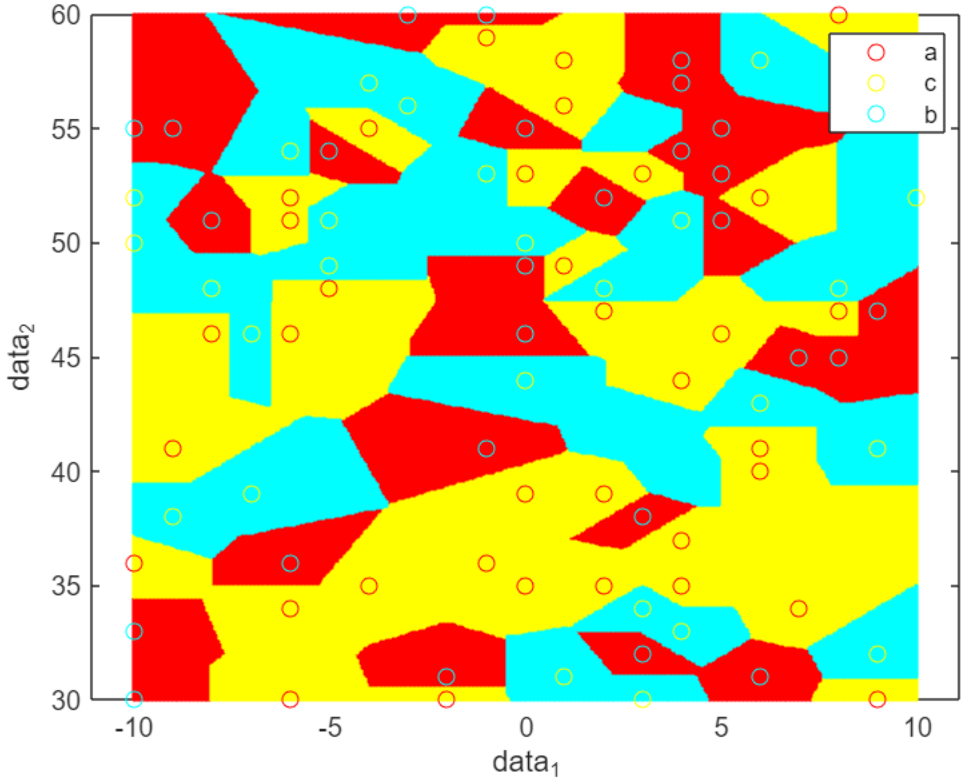
This serve as an example:

```
% Language: Matlab
% knn performance on 3 indicator
clear;close;clc
data_1=randi([-10,10],[1 100])';
data_2=randi([30,60],[1 100])';
indicator={'a','b','c'}';
index_array={indicator{randi([1,3],[1 100])}}';
```

```
gscatter(data_1,data_2,index_array,'ryc');
```




```
[X,Y ]=meshgrid(-10:.05:10,30:.05:60);
feather=table(data_1,data_2,index_array);
knnmodel=fitcknn(feather,"index_array");
predictions=predict(knnmodel,[X(:) Y(:)]);
gscatter(X(:),Y(:),predictions,'ryc');
hold on;gscatter(data_1,data_2,index_array,'ryc','ooo');
```



60
55
50
45
40
35
30
-10
-5
0
5
10
data$_2$
data$_1$
a
c
b