RP_tutorial

February 21, 2020

1 How to use Recurrence.py module

Please save the Recurrence.py module in your working folder

[51]:
```
import matplotlib.pyplot as plt
import numpy as np
from Recurrence import mi,fnn,rp
```

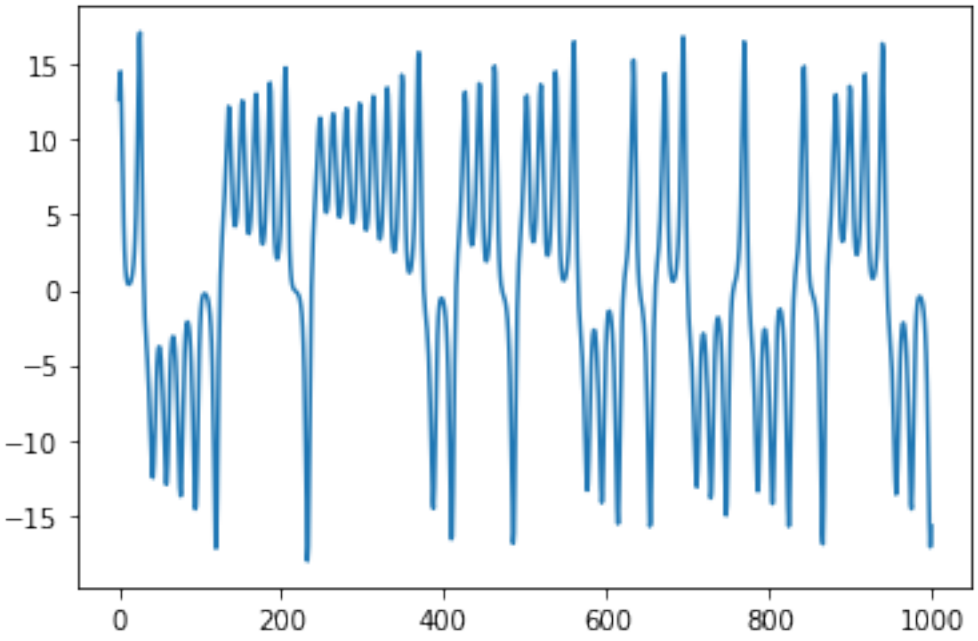
[52]:
```
# Preparing the dataset
#this is a test dataset I have, we can use your own dataset


data=np.loadtxt('lorenz.dat')[:,1][::4]

# data=np.loadtxt('file')
```

[53]:
```
plt.plot(data)
```

[53]:
```
[<matplotlib.lines.Line2D at 0x7f06fc26d550>]
```



1.1 Now calculate the Mututal information and embedding dimension:

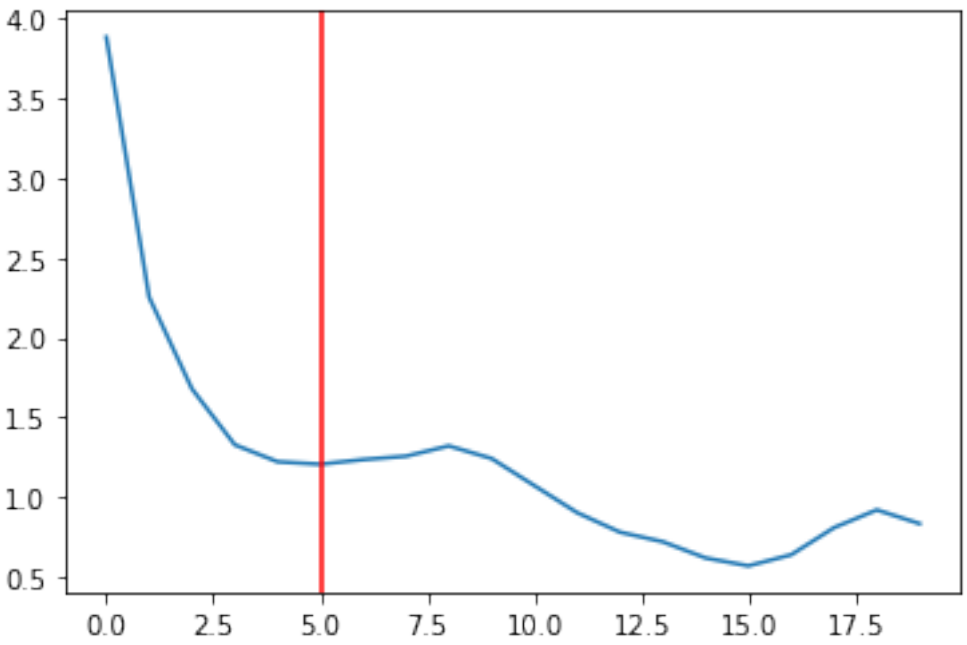
From Mutual information (**mi**) we can find the τ, from False nearest neighbours (**fnn**) we find the embedding dimension (**m**)

[54]:
```
# let's find the delay for this time series

lags,taus=mi(data,20 )
plt.plot(lags,taus)
plt.axvline(np.argmin(taus[:10]),color='r')
print("time delay for the system is:\n")
print("Tau=%d"%np.argmin(taus[:10]))
```

```
time delay for the system is:

Tau=5
```



1.2 Now we need to find out the dimension:

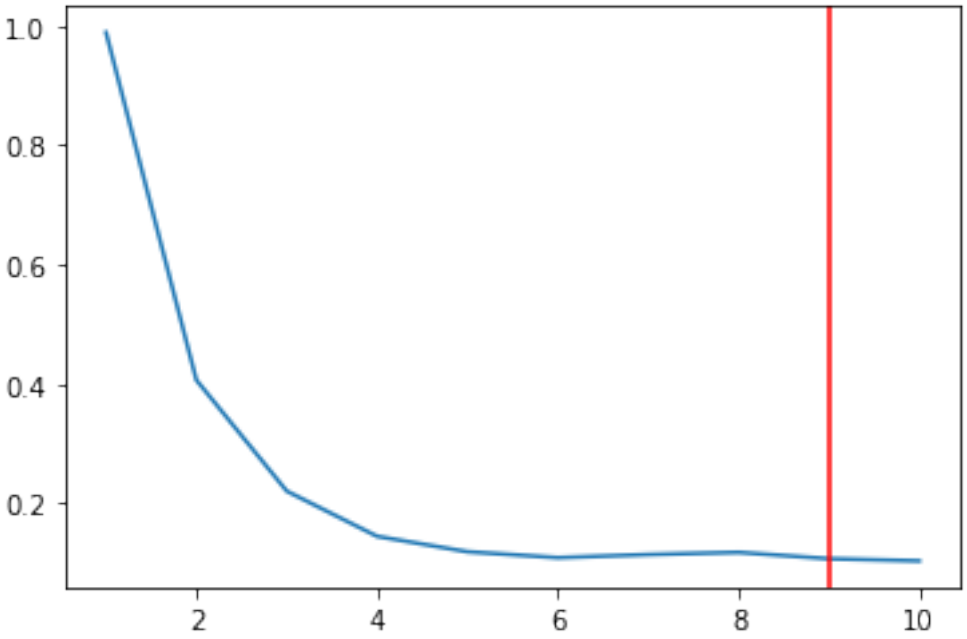
we need to use **fnn** function and need to pass the τ value which we have found from **mi**

[55]:
```
Dims,M=fnn(data,5,10)

plt.plot(Dims,M)
plt.axvline(np.argmin(M[:10]),color='r')
print("Embedding dimension of the system is:\n")
print("m=%d"%np.argmin(M[:10]))
```

```
Embedding dimension of the system is:

m=9
```



1.3 Now we are ready to calculate Recurrence plot:

With this module(Recurrence.py), you no need to do any embedding stuff, it's already there. You need to pass 4 arguments

Recurrence_plot=rp(data,embedding dimension, delay,threshold, threshold_by="mode") for threshold part, we need to select it properly.

Here we have three modes to select the threshold type:

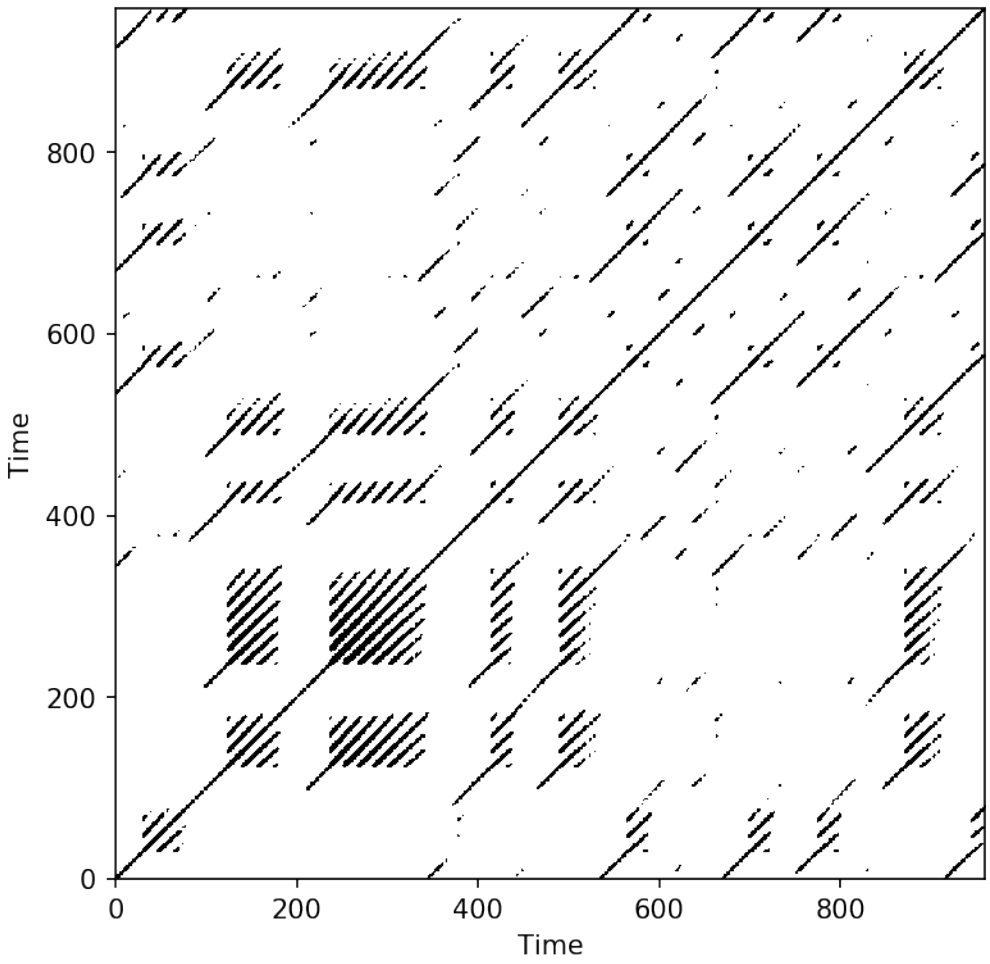
threshold_by="distance"
threshold_by="frr" it counts certain percentile of the distance matrix

[56]:
```
# Lets plot the recurrence plot

Recurrence_plot=rp(data,9,5,0.05,threshold_by='frr')
```

[57]:
```
plt.figure(figsize=(6,6),dpi=150)
plt.imshow(Recurrence_plot,cmap='binary',origin='lower')
plt.xlabel("Time")
plt.ylabel("Time")
```

[57]:
```
Text(0, 0.5, 'Time')
```



2 Recurrence Quantification Analysis

For this exercise, please download the recurrence_quantification.py script and save it in your working directory. The **"det"** function returns the determinism of the system. For this quantification we need to pass two arguments, **det(Recurrence matrix,Lmin)**.

2.1 Determinism =det(RM,Lmin=2)

We also need to compute the maximal diagonal line length **maxDL**. For this quantification we need to use **diagonal_lines_hist** and need to pass only one argument **diagonal_lines_hist(Recurrence_matrix)**. This function returns num_lines,bins,line_lengths. ## Num_lines,bins,line_len=diagonal_lines_hist(RM) ### so the maximum diagonal line length maxDL=max(line_len)

[59]:

[60]:

[70]:
```
from recurrence_quantification import det,diagonal_lines_hist
Det=det(Recurrence_plot,lmin=2.0)
nL,bi,LL=diagonal_lines_hist(Recurrence_plot)
```

```
estimating line length histogram…
diagonal lines histogram…
estimating DET…
diagonal lines histogram…
```

[ ]:

[ ]:

[ ]:

[ ]: