1. Numerical methods of Lyapunov coefficient calculation

$$d(t) = d_0 e^{\lambda_1 t}$$
$$\ln d(t) = \ln d_0 + \lambda_1 t$$

Consider a dynamical system:

$$\begin{cases} \dot{x} = f_1(x, y), \\ \dot{y} = f_2(x, y) \end{cases}$$

A trajectory consisting of points that are stable by Lyaponov is is called an attractor.

A time series can be split into $z_i = [x_i, ..., x_{i+m-1}]$. This is a transition from time series to a dynamical system. It can be transitiioned forward by Δt and then be taking the last z-vector one can have a new time series.

1. 1. Rosenstein algorithm

Let $x_1, ..., x_n$ be a dataset,

$$Y_i = [x_i, x_{i+1}, ..., x_{i+m-1}]$$
$$Y_i = \left[x_i, x_{i+\tau}, ..., x_{i+\tau(m-1)}\right].$$

Step-by-step:

1. $\{x_i\} \to \{Y_i\}$.
2. Finding nearest neighbours.

$$\forall Y_i : \ \tilde{N}_i = \left\{Y_j \middle| \ \varepsilon_{\min} < \|Y_i - Y_j\| < \varepsilon_{\max} \text{ and } |i - j| > \varepsilon_t\right\},$$

3. Take k nearest neigbours:

$$N_i = \left\{Y_{j_1}, ..., Y_{j_k}\right\}.$$

4. Calculate the distances after k steps: $d_{ij}(k) = \|Y_{i+k} + Y_{j+k}\|$ for each $k = \overline{0, ..., T}$.
5. Average those distances:

$$S(k) = \frac{1}{M'} \sum_{i=1}^{M\prime} \frac{1}{|N_i|} \sum_{Y_j \in N_i} \in d_{ij}(k),$$

Where M' is the number of Y-vectors used in computations (not all of them can be used).

6. Calculate Lyapunov exponent using linear regression:

$$S(k) \approx \alpha + \lambda_i (k \cdot \Delta t).$$

Hyperparameters: $m, \tau, \varepsilon_{\min}, \varepsilon_{\max}, \varepsilon_t, k, T$. It is important to know that this algorithm is extremely sensitive to hypterparameter values.

1. 2. Kantz algorithm

This algorithm differs only in the way that data is averaged.

Step-by-step:

5. Average dataset:

$$S(k) = \frac{1}{M'} \sum_{i=1}^{M\prime} \frac{1}{|N_i|} \sum_{Y_j \in N_i} d_{ij}(k).$$

6.

$$S(k) \approx S(0) e^{\lambda_1 (k \cdot \Delta t)}$$
$$\ln S(k) \approx \ln S(0) + \lambda_1 (k \cdot \Delta t).$$

2. Lyapunov spectre estimation

2. 1. Local linear maps method

Assume that locally $Y_{i+1} \approx A_i B_i + b_i$.

Let

$$\begin{pmatrix} Y_{j_1+1}^T \\ Y_{j_2+1}^T \\ \vdots \\ Y_{j_k+1}^T \end{pmatrix} = \begin{pmatrix} Y_{j_1}^T & 1 \\ Y_{j_2}^T & 1 \\ \vdots & \vdots \\ Y_{j_k}^T & 1 \end{pmatrix} \times \begin{pmatrix} A_i^T \\ b_i^T \end{pmatrix}.$$

Step-by-step:

1. Reconstruction: $\{x_i\} \to \{Y_i\}$.
2. Search for the k nearest neigbours.

$$N_i = \left\{Y_{j_1}, ..., Y_{j_k}\right\},\ Y_{j_k} \in \left\{Y_j \mid \quad \|Y_i - Y_j\| < \varepsilon \text{ and } |i - j| > \varepsilon_t\right\}.$$

3. $\forall Y_j \in N_i : Y_{j+1} \approx A_i Y_j + b_i$. Then, minimize MSE:

$$\sum_{Y_j \in N_i} \|Y_{j+1} - A_i Y_{j_1} - b_i\| \to \min_{A_i, b_i}$$

4. Form an orthonormal basis.
   4.1. Y_{i_0}
   4.2. $Q_0 = [q_1^0, q_2^0, ..., q_m^0],\ Q_0^T Q_0 = I.$
   4.3. $L_j = 0,\ j = 1, ..., m.$
5. Find A_{i_n} for each Y_{i_n} (see step 3)

$$V_{n+1} = A_{i_n} Q_n.$$

Then, use QR decomposition of A_{i_n}: $V_{n+1} = Q_{n+1} R_{n+1}$, where Q_{n+1} is an orthonormal basis, R_{n+1} is an upper triangular matrix. Then, $L_j = L_j + \ln(R_{n+1})_{ji}$, $i_{n+1} = i_n + 1$.

6. Calculate the Lyapunov exponents:

$$\lambda_j = \frac{L_j}{N_{\text{iter}} \Delta t}.$$

Note that A_i is a jacobian matrix of our dynamical system.

2. 2. Wolf method

Step-by-step:

1. Reconstruction $\{x_i\} \to \{Y_i\}$.
2. Initialization. Let $Y_0 = Y_i$, define an orthonormal basis for it: $q_1^0 = [1, 0, ..., 0]^T$, $q_2^0 = [0, 1, 0, ..., 0]^T$, ...
3. Take Y_k and evolve it in time by $\left\{q_1^k, q_2^k, ..., q_m^k\right\}$, $Y_{k+1} = Y_{i+1}$, then $\forall q_i$:

$$\|Y_j - Y_k\| < \varepsilon_{\max}$$
$$|j - k| > \varepsilon_t$$
$$\delta = Y_j - Y_k : \alpha = \arccos\left(\frac{\delta \cdot q_i^k}{\|\delta\| \quad \|q_i^k\|}\right) < \varepsilon_{\min}$$

4. $v_j = Y_{j+\Delta} - Y_{k+\Delta} \Rightarrow \{v_1, v_2, ..., v_m\}$.
5. For $j = \overline{1, ..., m}$:

for $j = 1$: $u_1 = \frac{v_1}{\|v_1\|}$, $L_1^k = \ln\|v_1\|$

for $j = 2$: $w_2 = v_2 - (v_2 \cdot u_1)u_1$, $u_2 = \frac{w_2}{\|w_2\|}$. $L_2^k = \ln\|w_2\|$ and so on.

6. $q_1^{k+1} = u_1, ..., q_m^{k+1} = u_m$.

for j:

$$w_j = v_j - \sum_{i=1}^{j-1} (v_j u_i) u_i$$
$$u_j = \frac{w_j}{\|w_j\|}$$
$$L_j^k = \ln\|w_j\|.$$

Them,

$$\lambda_j = \sum_{k=1}^{\text{max\_iter}} \frac{L_{jk}^k}{\text{max\_iter} \cdot \Delta \cdot \Delta t}.$$

3. Entropy-complexity plane

Blah-blah-blah.

4. How to choose the size of reconstruction

False nearest neigbours approach. For $m = m_{\min}, \ldots, m_{\max}$.

1. Sample z^m vectors of size m.
2. $\#$ of NN $= |\left\{\left(z_i^m, z_j^m\right) \mid \|z_i^m - z_j^m\| < \varepsilon\right\}| \ \forall z_i^m, z_j^m$.
3. Sample z^{m+1} – vectors of size $m + 1$.
4.
$$\begin{aligned}\# \text{ of false NN} = |\Big\{\left(z_i^{m+1}, z_j^{m+1}\right) \mid \|z_i^m - z_j^m\| < \varepsilon, \\ \|z_i^{m+1} - z_j^{m+1}\| > \varepsilon, |i - j| > \tau\Big\}|, \forall z_i^{m+1}, z_j^{m+1}.\end{aligned}$$

FNN has a dip at the best m.