Algoritmo de detección de caos basado en la creación de imágenes de las muestras subrogadas

Abstract

El objetivo del algoritmo que se describe a continuación es evaluar si una serie temporal observada es determinista / caótica o estocástica de manera sencilla y automática.

1 Algoritmo

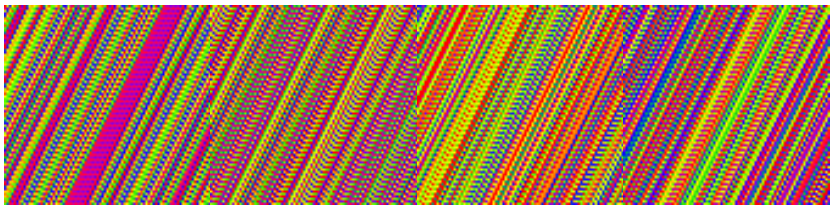
1. Sea $\{x_t\}$ con $t = 1, ..., T$, la serie temporal observada sobre la que se quiere determinar si procede de un sistema caótico (determinista) o estocástico. En primer lugar, se divide dicha serie temporal en k vectores de dimensión $l = \frac{T}{k}$, construidos tal que:

$$x_0 \begin{cases} x_0^1 = (x_1, ..., x_l) \\ x_0^2 = (x_{l+1}, ..., x_{2l}) \\ ... \\ x_0^k = (x_{(k-1)l+1}, ..., x_{kl}) \end{cases}$$

2. Una vez dividida la serie temporal en k tramos de longitud l se toman m vectores a partir de una muestra aleatoria simple (sin reposición) de l elementos de la serie temporal original $\{x_t\}$, tal que se obtiene

$$x_1 \begin{cases} x_1^1 = (x_{j_0^1}, x_{j_1^1}, ..., x_{j_l^1}) \\ x_1^2 = (x_{j_0^2}, x_{j_1^2}, ..., x_{j_l^2}) \\ ... \\ x_1^m = (x_{j_0^1}, x_{j_1^1}, ..., x_{j_l^1}) \end{cases}$$

3. Se transforma cada uno de los vectores en una imagen a partir del embebimiento de Takens y una coloración adecuada. Por ejemplo:



donde puede apreciarse -mínimamente- a simple vista que las dos primeras (de la s. original -ec. logística-) tienen una coloración algo distinta a la de dos submuestras de la s. original, porque el patrón cambia.

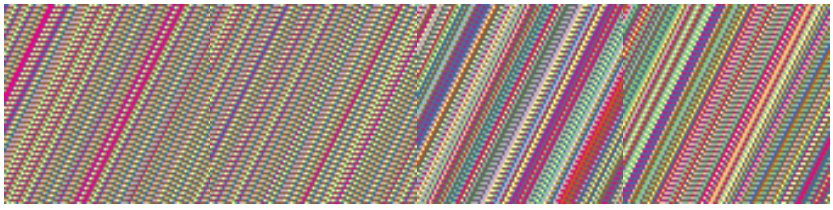
4. Ahora se implementa un algoritmo de clasificación de imágenes (en este caso, se ha utilizado una red neuronal de convolución). Para crear los datos de entrenamiento, se toman aleatoriamente (la mitad?) de los vectores especificados en el primer paso x_0 y todos los pertenecientes a x_1. El resto de vectores de x_0 del modelo se reservan para la predicción y los denotamos como y. Se crea un vector de etiquetas e asociado a la matriz de datos de entrenamiento $X = [x_0^{i_1}|x_0^{i_2}|...|x_0^{i_{k/2}}|x_1^1|...|x_1^m]$ tal que $e(x) = 0$ si el elemento x proviene de x_0 y $e(x) = 1$ en otro caso.

5. Una vez entrenado el modelo con X y e se trata de clasificar las imágenes del vector y. Se espera que, en caso de que la serie original sea estocástica, el poder clasificatorio del modelo sea nulo, dado que no existe ninguna diferencia entre el patrón de la serie otriginal y el de sus muestras (i.e. muy malos resultados de la predicción), mientras que en un proceso determinista ese patrón sirve como medio de clasificación y los resultados son mejores (i.e. >0).

2 Resultados de la implementación

Se prueba con una ecuación logistica, donde se obtiene un 0.60 de precisión en la red neuronal y una serie aleatoria proviniente de una uniforme entre 0 y 1, que obtiene 0 de precisión. Probablemente se tienda a obtener exactamente 0 en casos alearorios y >0, aunque posiblemente bajo, en procesos deterministas.

Se implementa el mapa Gaussiano. Con la coloración *prism* los resultados son semejantes a la aleatoriedad (0), mientras que en caso de utilizar la coloración *Accent,* estos resultados pasan al 0.69. El hecho de que exista una coloración en que el modelo tenga una potencia tan alta (0.60 o 0.69) es un indicio suficientemente fuerte de que el proceso es determinista. Este es el caso y la búsqueda puede ser optimizable[1]:



3 Creación de "nnchaos"

"nnchaos" es la función principal del paquete *img4chaos*, disponible en GitHub[2], que sirve para recorrer automáticamente las 82 posibilidades de coloración implementadas nativamente en el paquete *matplotlib*. En teoría, el hecho de que la clasificación sea *buena* en una de las coloraciones es un indicio suficiente de la existencia de caos. Por ejemplo, los resultados de una prueba realizada en Google Colab[3] con la mencionada ecuación logística, el mapa de Gauss y el proceso aleatorio de valores procedentes una uniforme $U(0, 1)$ indican que existen coloraciones para los que el modelo clasifica bien la ecuación logística pero no el mapa de Gauss y viceversa. En cambio, ninguna coloración consigue distinguir la serie original de las muestras en el proceso aleatorio.

La función *"nnchaos"* recibe un argumento además de la serie temporal, el número de *epochs* o iteraciones de la red neuronal (iteraciones más altas se recomiendan -en principio- pero lógicamente el proceso de computación es más largo).

[1] En la sección 3 se implementa una solución.
[2] `https://github.com/i-rb/img4chaos`
[3] `https://colab.research.google.com/drive/12FgayYG8NzXajFbwc3qAk1MkqJB1jCrS?usp=sharing`