Exercise: MA8702, Spring 2020

Dynamic time warping

We consider two time series that should represent related signals with random noise. They are not aligned and dynamic time warping (DTW) is used for coherent comparison. In our case we simulate two time series datasets with a given warping path. The goal is then to get back to the 'true' warping path from the time series data.

In MATLAB the code command for performing DTW is *dtw*. In R there is a *dtw package* (see also, Toni Giorgino (2009). Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. Journal of Statistical Software, 31(7), 1-24). In python there is a packge called dtw-python.

The tasks are as follows:

- Simulate a length 520 autoregressive process of order 1; $x(1), \dots, x(520)$, with mean 0, stationary variance 1 and autocorrelation parameter $\phi = 0.9$. This means that $x_1 \sim N(0,1)$ and $x_{i+1} = \phi x_i + \delta_i$, for independent variables $\delta_i \sim N(0, 1-\phi^2)$, for $i = 1, \dots, 520$.

- Construct a warping path such that $j(1) = 51, j(2) = 52, \dots, j(200) = 250$. $j(i) = 250$ for $i = 201, \dots, 240, j(241) = 251, j(242) = 252, \dots j(500) = 510$.

- Simulate another time series $y_i = x_j + \epsilon_i$, $\epsilon_i \sim N(0, 0.5^2)$, $i = 1, \dots, 500$.

- Use DTW to extract the most likely path.

The warping path is illustrated in Figure 1. Figure 2 shows realizations of two time series, using this warping path.

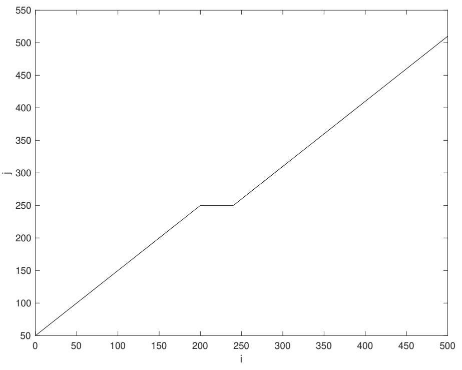

Figure 1: True warping path between two misaligned time series.

1. Repeat the simulations (with the same path) to study the variability in the warping.

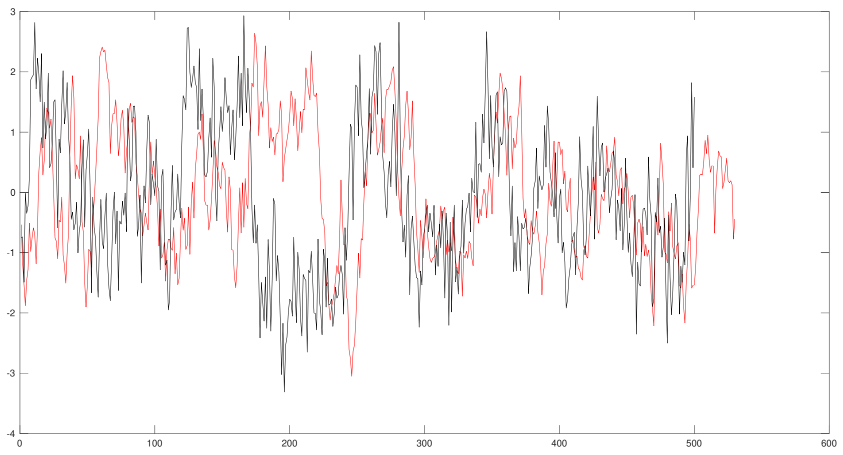

Figure 2: Realizations of the two time series.

2. For each run, plot the aligned signals to see the differences in the time series data.

Multidimensional scaling and process clustering

In this exercise we will apply multidimensional scaling (MDS) to process realizations and use this for k-means clustering. We run two varieties of random walks of length 200 on the line and compare the results. One model of a random walk (50 first runs) has 0.5 probability of walking left / right. The other model one (50 last runs) has 0.6 probability of walking to the right, and 0.4 probability of walking left. In total there are then 100 process realizations. Since we know the mechanism we can easily extract the labels (1: for the first 50, 2: for the last 50), but in the clustering we will assume the labeling is unknown.

In the MDS plus k-means clustering method we do here, the process realizations will not always split in the two right cluster. Instead, we expect some to be falsely clustered.

In Matlab and R, MDS is done by cmdscale. In Python there are scikit learn packages for MDS. K-means clustering is done by kmeans commands in Matlab, R and python, usually with some kind of built-in form of initialization.

1. Generate 50 process of length 200 from both types of processes. Plot the runs of different processes. Can you see tendencies of differences?

2. Conduct MDS and visualize all the 100 datasets in a 2D plot.

3. Perform K-means clustering of the MDS data.

4. Check the performance of the clustering compared with the 'true' process labels 1 and 2.