Chapter 1

Time series analysis

1.1 catch22 features

Table 1.1 lists *catch22* features.

1.2 UMAP hyperparameters

brief intro to explain what UMAP does ("dimensionality reduction") + references

UMAP has several hyperparameters, of which four have major effects on the embedding:

- The *number of neighbours* (n) to consider when approximating the local metric controls how the method balances local and global structure in the data. With low values of this parameter, the algorithm concentrates on very local structure, potentially to the detriment of the big picture. As the value increases, the algorithm 'glues' more nodes together to form clusters.
- The *largest embedding dimension* (d) controls the number of dimensions the data is reduced to. In other words, it controls whether the resulting map is one-dimensional, two-dimensional, three-dimensional, or of higher dimensions.

Feature name	Description
DN_HistogramMode_5	Mode of z-scored distribution (5-bin histogram)
DN_HistogramMode_10	Mode of z-scored distribution (10-bin histogram)
SB_BinaryStats_mean_longstretch1	Longest period of consecutive values above the mean
DN_OutlierInclude_p_001_mdrmd	Time intervals between successive extreme events above the mean
DN_OutlierInclude_n_001_mdrmd	Time intervals between successive extreme events below the mean
first_1e_ac	First 1/e crossing of autocorrelation function
firstMin_acf	First minimum of autocorrelation function
SP_Summaries_welch_rect_area_5_1	Total power in lowest fifth of frequencies in the Fourier power spectrum
SP_Summaries_welch_rect_centroid	Centroid of the Fourier power spectrum
FC_LocalSimple_mean3_stderr	Mean error from a rolling 3-sample mean forecasting
CO_trev_1_num	Time-reversibility statistic, $\langle (x_{t+1} - x_t)^3 \rangle_t$
CO_HistogramAMI_even_2_5	Automutual information, $m = 2, \tau = 5$
IN_AutoMutualInfoStats_40_gaussian_fmmi	First minimum of the automutual information function
MD_hrv_classic_pnn40	Proportion of successive differences exceeding 0.04σ
SB_BinaryStats_diff_longstretch0	Longest period of successive incremental decreases
SB_MotifThree_quantile_hh	Shannon entropy of two successive letters in equiprobable 3-letter symbolization
FC_LocalSimple_mean1_tauresrat	Change in correlation length after iterative differencing
CO_Embed2_Dist_tau_d_expfit_meandiff	Exponential fit to successive distances in 2-d embedding space
SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1	Proportion of slower timescale fluctuations that scale with DFA (50% sampling)
SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1	Proportion of slower timescale fluctuations that scale with linearly rescaled range fits
SB_TransitionMatrix_3ac_sumdiagcov	Trace of covariance of transition matrix between symbols in 3-letter alphabet
PD_PeriodicityWang_th0_01	Periodicity measure of Wang et al. (2007)

Table 1.1: *catch22* features, adapted from Lubba et al. (2019).

- The *minimal distance* (min_dist) controls the desired separation between close points in the embedding space. Specifically, this parameter controls how tightly the algorithm is allowed to pack points together. With low values, the visualisation forms 'clumps'.
- The previous hyperparameters are numerical, but the *metric* hyperparameter instead specifies the distance metric that is used to compute distances in the ambient space of the input data. For example, this metric can be the Euclidean distance, the cosine distance, or other metrics used to compute the distances between two vectors of numerical data.

1.3 Classification pipeline

In machine learning, classification is defined as the process of identifying a category that a piece of input data belongs to. In this section, the classification task is identifying whether a time series (input data) is oscillatory (belongs to one category of two) or non-oscillatory (belongs to the other category of two).

A typical classification pipeline can be described by the following steps:

1. *Pre-processing of data:* Input data is cleaned or normalised. For example, to classify oscillatory time series, the input time series may be normalised to give similar dynamic ranges.
2. *Labelling:* Each piece of input data has a label assigned to it to denote which category it belongs to. For example, to classify oscillatory time series, a human can subjective assign the label '0' for non-oscillatory time series and '1' for oscillatory time series, for a total of two categories.

3. *Featurisation:* Input data converted to feature vectors in the process of featurisation. This process uses domain knowledge related to the type or origin of the data to define characteristics of the data that may be useful for classification.
4. *Train-test split:* The input data set is then randomly divided into a training data set and a test data set.
5. *Training of model:* The machine learning model is then fit on the (featurised) training data set and its labels to fit parameters in the model.
6. *Evaluation of model on test dataset:* The model, trained on the training dataset, is used to predict the labels of data in the (featurised) test data set. The performance of the model is then evaluated on the test data set. This evaluation is based on computing quantities that express how well the model assigns labels to data, compared to the labels defined earlier.

1.4 Gillespie ~~noise~~ algorithm for stochastic chemical systems

To define the Gillespie algorithm, consider such a system with M reactions $R_1, \ldots, R_j, \ldots R_M$ involving N species $S_1, \ldots, S_i, \ldots S_N$ in a fixed volume V at thermal equilibrium. Let $X_i(t)$ represent the number of molecules of S_i at time t, and the state vector

$$\mathbf{X}(t) \coloneqq [X_1(t), \ldots, X_N(t)] \tag{1.1}$$

thus gives the state of the system at any given time t.

Each reaction R_j is described by two quantities:

1. A state-change vector $\mathbf{v}_j \coloneqq [v_{1,j}, \ldots, v_{N,j}]$ which defines how the stoichiometry of the system changes if the reaction occurs. $v_{i,j}$ represents the change in the stoichiometry of S_i when R_j occurs.
2. A propensity function a_j, which gives the probability, given a the state $\mathbf{X}(t) = \mathbf{x}$, that one R_j reaction occurs in the volume V within the following short time interval $[t, t + dt)$. This function is defined by

$$a_j(\mathbf{x})dt \overset{?}{=} k_j \prod_{n=1}^{N} \mathbf{v_n} S_n \tag{1.2}$$

not sure this is correct

where k_j is the rate constant of reaction R_j.

The Gillespie algorithm aims to ~~estimate~~ predict the temporal evolution of the state vector given the initial state $\mathbf{X}(t_0) = \mathbf{x}_0$. It does so by iteratively choosing the next reaction that occurs, based on its ~~probability~~ propensity funct., and then choosing its firing time based on a probability distribution. Combining these simulations gives a trajectory of state vectors across the time course of interest. In detail, the direct Gillespie algorithm can be defined as stated in algorithm 1 (Gillespie, 2007):

Algorithm 1: Direct method of the Gillespie algorithm

```
Input: Stochastic model (with species $S_1, \ldots, S_i, \ldots S_N$ and reactions
       $R_1, \ldots, R_j, \ldots R_M$, along with a state-change vector $\mathbf{v_j}$ and a rate
       constant $k_i$ for each reaction $R_j$); initial time $t_0$; and initial model
       state $\mathbf{X}(t_0) = \mathbf{x}_0$
Output: Trajectory of state vectors $\mathbf{X}(t)$, with $t$ taking discrete values in
        $[t_0, t_{\max}]$
while $t < t_{\max}$ do
    Calculate the propensities $a_j(\mathbf{x})$ based on the current state $\mathbf{x}$;
    Calculate the combined propensity $a_0(\mathbf{x}) = \sum_j a_j(\mathbf{x})$;
    Generate two random numbers $r_1$ and $r_2$, both from the uniform
      distribution $U(0, 1)$;
    Choose the next reaction $R_j$, with $j$ given by the smallest integer that
      satisfies $\sum_{j'}^{j} a_{j'(\mathbf{x})} > r_1 a_0(\mathbf{x})$;
    Calculate the time to the next reaction $\tau = \frac{1}{a_0(\mathbf{x})} \ln(\frac{1}{r_2})$;
    Simulate the next reaction by updating the state vector $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v_j}$ and
      store the new vector in $\mathbf{X}(t)$;
    Update the time by $t \leftarrow t + \tau$ and store the new time;
end
return Trajectory of state vectors $\mathbf{X}(t)$ for a vector of times $t$;
```