Definition

In this case we choose soft margin SVM to investigate the classification of time series in the big data context. It seeks to find a decision boundary, a linear model represented by a line in 2D or a hyperplane in higher dimensions, that maximizes the margin between the two classes, ensuring that all data points are correctly classified but at the same time allows for some points to be on the wrong side of the decision boundary, introducing a margin of tolerance for classification errors, while KNN does not explicitly define a decision boundary, it computes distances between the query point and all training points and assigns the majority class label among the K nearest neighbors instead. SVM is also superior in handling of outliers and noisy data due to its focus on maximizing the margin and its dependence on support vectors. Outliers have less influence on the decision boundary as long as they are not among the support vectors. In contrast, KNN is sensitive to outliers and noisy data, especially when the value of K is small. Outliers can significantly affect the majority class among nearest neighbors, leading to misclassifications. SVM also has better performance in memory usage for it only needs to store the support vectors, while KNN requires storing the entire training dataset, making it memory-intensive, especially for large datasets.

Preprocession

Here we use utility function to extract numeric values from input data, handle any errors by coercing them to NaN, then forward fill any missing values with the most recent non-null value that occurred before it in the same column or row, aiming to facilitate data cleaning and standardization tasks within a data processing pipeline. then we record the characteristics of the cleaned data, including the number of time-series, their lengths, number of unique values, and any other relevant features.

However the cleaned data still have redundant data, which makes it necessary to find the starting line number of actual data in a text file and skip any comment lines. We use series of no-label function to assure that it starts from the identified data start line, followed by parsing the data, determining the number of columns, creating a DataFrame, and converting all feature columns to numeric types. And extract function are applied again to each cell, converting the entire DataFrame to numeric types, and filling missing values in both the training and test datasets with the mean of each column. Lastly, we merge the cleaned text and arff datasets, ensuring consistent column names and removing duplicated features from the arff dataset.

We divides the features into subsets to create multiple subplots, each containing boxplots as visualization of the distribution of a subset of features, which helps in identifying outliers and understanding the spread of data within each feature.(PIC: Boxplot)

We transform both the training and testing features using the computed mean and standard deviation, and replace the original feature values with the standardized values, known as mean feature values, in both the training and testing datasets.(PIC: mean feature value for each label)

We visualize the standardized features by plotting histograms of the original and standardized feature values, which helps in understanding how the distribution of feature values changes after standardization, making it easier to compare features and detect any shifts in data distribution.(PIC: Histogram)

Integrate DTW with SVM

calculate the distance

Implement the Dynamic Time Warping (DTW) algorithm to compute the distance between pairs of time series arrays, combined with soft margin Support Vector Machine (SVM) for classification. First we initialize the parameters with provided default values, and choose radial basis function (RBF) kernel to initialize SVM model. Then we try to train the model using the input data (x) and corresponding labels (l). Then we calculate the DTW distance matrix between every pair of samples in the training data, which is then used as input features to train the SVM model with same method.

We take a set of testing data x as input, then use the trained SVM model to predict the class labels for the testing data based on the computed DTW distances.

practice

We randomly select 50 samples from the training and test datasets, extract features and labels from the sampled training and testing datasets.

Fits the model (m) using the sampled training data, for which predict labels with trained models

Evaluation

We compute a classification report, which includes precision, recall, F1-score, and support for each class, which allows for assessing the performance of the classifier based on this subset of data. (Table for precision, recall, f1-score, support)Meanwhile the confusion matrix is also used to provide a summary of the predictions versus the actual labels.(Table for confusion matrix and explain its evaluation)