1. Classification and feature engineering

We'll now discuss one of the most common categories of machine learning problems: classification. We'll also discuss the concept of feature engineering in the context of time series data.

2. Always visualize raw data before fitting models

Before we begin, let's take a moment to once again visualize the data we're dealing with. There is a lot of complexity in any machine learning step, and visualizing your raw data is important to make sure you know where to begin.

3. Visualize your timeseries data!

To plot raw audio, we need two things: the raw audio waveform, usually in a 1- or 2-dimensional array. We also need the timepoint of each sample. We can calculate the time by dividing the index of each sample by the sampling frequency of the timeseries. This gives us the time for each sample relative to the beginning of the audio.

4. What features to use?

As we saw in the introduction, using raw data as input to a classifier is usually too noisy to be useful. An easy first step is to calculate summary statistics of our data, which removes the "time" dimension and give us a more traditional classification dataset.

5. Summarizing timeseries with features

Here we see a description of this process. For each timeseries, we calculate several summary statistics. These then can be used as features for a model. We have expanded a single feature (raw audio amplitude) to several features (here, the min, max, and average of each sample).

6. Calculating multiple features

Here we show how to calculate multiple features for a several timeseries. By using the "axis equals -1" keyword, we collapse across the last dimension, which is time. The result is an array of numbers, one per timeseries.

7. Fitting a classifier with scikit-learn

In the last step, we collapsed a two-dimensional array into a one-dimensional array for each feature of interest. We can then combine these as inputs to a model. In the case of classification, we also need a label for each timeseries that allows us to build a classifier.

8. Preparing your features for scikit-learn

In order to prepare your data for scikit-learn, remember to ensure that it has the correct shape, which is samples by features. Here we can use the column_stack function, which lets us stack arrays by turning them into the columns of a two-dimensional array. In addition, the labels array is 1-dimensional, so we reshape it so that it is two dimensions. Finally, we fit our model to these arrays, X and y.

9. Scoring your scikit-learn model

Now that we've fit our model, we'll score the classifier. There are many ways that we can score a classifier with scikit-learn. First, we show how to generate predictions with a model that has been fit to data. If we have separate test data, we can use the "predict" method to generate a predicted list of classes for each sample. We can then calculate a score by dividing the total number of correct predictions by the total number of test samples. Alternatively, we can use the accuracy_score function that's built into scikit-learn by passing the test set labels and the predictions.