1. Time-delayed features and auto-regressive models

One of the most important steps in a machine learning pipeline is feature extraction. Defining high-quality and relevant features gives your model the best chance at finding useful patterns in the data. In this lesson, we'll cover some techniques for extracting features from data.

2. The past is useful

Perhaps the biggest difference between timeseries data and "non-timeseries" data is the relationship between data points. Because the data has a linear flow (matching the progression of time), patterns will persist over a span of datapoints. As a result, we can use information from the past in order to predict values in the future.

3. A note on smoothness and auto-correlation

It's important to consider how "smooth" your data is when fitting models with timeseries. The smoothness of your data reflects how much correlation there is between one time point and those that come before and after it. The extent to which previous timepoints are predictive of subsequent timepoints is often described as "autocorrelation", and can have a big impact on the performance of your model.

4. Creating time-lagged features

Let's investigate this by creating a model in which previous timepoints are used as input features to the model. Remember that regression models will assign a "weight" to each input feature, and we can use these weights to determine how "smooth" or "autocorrelated" the signal is.

5. Time-shifting data with Pandas

First we'll create time-shifted versions of our data. This entails "rolling" your data either into the future or into the past, so that the same index of data now has an different timepoint in it. We can do this in Pandas by using the dot-shift method of a DataFrame. Positive values roll the data backward, while negative values roll the data forward.

6. Creating a time-shifted DataFrame

Here we use a dictionary comprehension that creates several time-lagged versions of the data. Each one shifts the data a different number of indices into the past. Since our data is recorded daily, this corresponds to shifting the data so that each index corresponds to the value of the data N days prior. We can then convert this into a DataFrame where dictionary keys become column names.

7. Fitting a model with time-shifted features

We will now fit a scikit-learn regression model. Note that in this case, "many_shifts" is simply a time-shifted version of the timeseries contained in the "data" variable. We'll fit the model using Ridge regression, which spreads out weights across features (if applicable) rather than assign it all to a single feature.

8. Interpreting the auto-regressive model coefficients

Once we fit the model, we can investigate the coefficients it has found. Larger absolute values of coefficients mean that a given feature has a large impact on the output variable. We can use a bar plot in Matplotlib to visualize the model's coefficients that were created after fitting the model.

9. Visualizing coefficients for a rough signal

Here we see the coefficient values for a relatively non-smooth signal. On the left the signal is clearly jumping around, and on the right we see the model coefficients (one per time lag) drop to zero very quickly.

10. Visualizing coefficients for a smooth signal

And here is a timeseries that is more smooth. On the left we can see more "structure" to the data, and on the right we see that the coefficients for time lags drop off to zero smoothly.