Lecture Timeseries (1)

Lesson goals

- The student recognizes time series data as a distinct type of data and knows the techniques for making predictions using this type of data.
    - Bloom: weten
    - Miller: weten
    - Final qualifications: C. Data analytics: > The Master's graduates apply appropriate data science and machine learning algorithms and techniques for complex data analysis. They prepare data, including selecting, cleaning, and combining data from various sources. They determine algorithms suitable for data analysis and establish the test design. They build models and evaluate them by interpreting model results, predefined success criteria, and the test design. They decide on follow-up actions based on the evaluation results of the model.

Preparation

- Read Géron ch. 15 from "Forecasting a time series" until but not including "Preparing the Data for Machine Learning Models".

Lesson duration

- 2:30 including 3 15 min breaks

Block 1 (50 min)

- Energizer: what did you read in Géron?
    - What is a time series? data with values at different time steps, usually (should be: always except in *very* special circumstances) at regular intervals.
    - Used for forecasting
    - Example application: forecast number of passengers of a public transport system
    - Seasonality: pattern repeats.
    - Naive forecasting: copying a past value to make forecasts.
    - Lagged time series
    - Measure the quality of a forecast: calculate mean absolute error / mean absolute percentage error
    - ARMA Model (autoregressive moving average): weighted sum of lagged values, adds moving average
    - Differencing: approximates the derivative (gives slope at each time step). This eliminates linear trends. dd consecutive rounds of differencing computes an approximation of the d^{th}dth order derivative of the time series.
    - ARMA is a family of models. Examples of other ARMA models: SARIMA (seasonal ARIMA)
- Overview

    - A time series is a sequence of observations of one or more variables. These observations occur in successive order over some period of time, usually (but occasionally not) at regular intervals.
    - In many time series observations are closely related to each other.
    - This means an observation made at some point in the past can help predict an observation that is made at some later point in time: forecasting.
    - The default forecasting technique is to simply state that the next observation will be identical to the last observation. This works surprisingly well. Later on during this lesson we will learn what to do when it doesn't.

- Demo 1: naive forecast with example.

- Slightly less naive forecasting: ARMA

    - Apply some function to all previous observations or to a subset of previous observations (only the last 10 observations, for example): $y_t = f(y_1, ..., y_{t-1}) + \epsilon_t$yt = f (y1, ..., yt–1) + ϵt (ϵϵ is some error). The forecast value is $\hat{y}_{t+1} = f(y1, ..., yt)$y^t+1 = f (y1, ..., yt).
    - This technique is called **ARMA**: **a**uto**r**egressive **m**oving **a**verage:
        - **Autoregressive**: looking back at itself.
        - **Moving average**: calculate an average of all previous values *that updates as we add more values*.
        - AR: $Yt = \beta_1 y_{t-1} + \beta_2 y_{t-2} ... + \beta_k y_{t-k}$Y t = β1yt–1 + β2yt–2... + βkyt–k
        - MA: $Yt = \alpha_1 \epsilon_{t-1} + \alpha_2 \epsilon_{t-2} ... + \alpha_k \epsilon_{t-k}$Y t = α1ϵt–1 + α2ϵt–2... + αkϵt–k : this is the sum of the residuals / errors for each observation done at time $t - k$t – k (the difference between the observed value and the predicted value).
        - ARMA combines the two, by summing them:
          $Yt = \beta_1 y_{t-1} + \alpha_1 \epsilon_{t-1} + \beta_2 y_{t-2} + \alpha_1 \epsilon_{t-2} ... + \beta_k y_{t-k} + \alpha_k \epsilon_{t-k}$Y t = β1yt–1 + α1ϵt–1 + β2yt–2 + α1ϵt–2... + βkyt–k + αkϵt–k

- Demo 2: manual ARMA

- Demo 3: manual ARMA, does not work

    - The reason demo 3 does not work is because ARMA assumes the time series is *stationary*. "Stationary" means there is no *seasonality* (recurring patterns) in the data.

- Demo 4: reducing seasonality using differencing

    - Differencing does not completely undo the effects of seasonality, but it does reduce its effects.
    - Differencing can also help remove underlying larger trends. In economics, for example, long term trends (continuous growth) are often less interesting than smaller trends.
    - Better results can be had using a technique called Fourier analysis, but that is outside of the scope of this lecture.

- It is possible to combine differencing and ARMA to make predictions: **ARIMA** : autoregressive integrated moving average. There is also ARIMA + seasonality: **SARIMA**.

- The Python library `statsmodels.tsa` is the library to use for working with time series.

Block 2 (50 min)

- Reminder: how to measure the quality of a forecast?
    - **MAE** (Mean Absolute Error): the mean of the absolute difference between predicted values and actual values: $MAE = \frac{1}{n}\sum_{t=1}^{t=n}|y' - y|$ MAE = n1 t=1∑t=n ∣y′ − y∣.
        - Units are the same as the units of the values themselves.
        - Does not give you the relative size of the difference.
    - **MAPE** (Mean Absolute Percentage Error): the mean of the *relative* differences (ie as a percentage) between the real values and the predicted values. $MAPE = \frac{1}{n}\sum_{t=1}^{t=n}\frac{|y' - y|}{y} * 100\%$ MAPE = n1 t=1∑t=n y∣y′−y∣ ∗ 100%
        - Note you divide by the original values, so these can not contain 0 values.
    - **MSE** is occasionally used, if some of the predictions are very far off and you want to penalize these stronger than other predictions.

- Now: some demos of the `statsmodels.tsa` and the `statsmodels.graphics` library:
    - Demo 5: using statsmodels.tsa to show trends and seasonality
        - Autocorrelation function plots help in data exploration:
            - They show you trends and seasonality in the data.
            - They help you decide which models to use. If data contains no seasonality at all, ARMA is fine. Otherwise you may need SARIMA or something else.
            - They help you decide how to preprocess the data.
            - They show anomalies / oddities.
    - Demo 6: Using statsmodels.tsa to decompose a time series

- Trend and seasonality analysis using autocorrelation function plots is not foolproof, however. An infamous example is the sunspots data set.

- Demo 7: Using statsmodels.tsa to analyse and decompose sunspot dat a
    - Note the performance of the simple function (the 11 year sinus cycle) is certainly no worse than that of the TS analysis
    - Note the explanatory power of the simple function (the 11 year sinus cycle) is much greater.

Block 3 (50 min)

- Most of our data will be somewhere between the synthetic data from the demos and the sunspot data in terms of complexity. The tools in statsmodels.tsa may therefore work very well - or at least reasonably well.

- Often our data is noisy, however.

- To remove noise, we can apply filters to our data.
    - A filter smoothes out small variations in the data.
    - You can think of them in terms of audio data. Small variations can be considered to be high frequency noise (hiss). An audio filter (like an equalizer) can remove those high frequencies.
    - In fact, one of the filters in the demos is used in audio applications! (synthesizers).

- Demo 8: smoothing data using filters

    - For synthetic data, filters work very well - because the noise is likely just that: noise.
    - For real data, the noise may in fact not be noise and may even contain additional information.

- Demo 9: smoothing sunspot data using filters

- A possible solution to the problem of uncovering patterns in complex data is something called *Fourier analysis*

    - In Fourier analysis a series of sine waves are added together to approximate a signal.
    - These approximations are often much better than the simple "trend + seasonality" analyses we have seen so far.
    - Performing Fourier analysis is computationally expensive.
    - Performing Fourier analysis brings with it a risk of overfitting: *any* signal can be created if you use enough sine waves, also overfitted ones.
    - If you must use Fourier analysis, filtering beforehand is often a good idea.
    - Make sure to provide a large test set to check if you're overfitting.