Machine learning II

Master Data Science

Winter Semester 2019/20 **Prof. Tim Downie**

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN

University of Applied Sciences

Workshop 11
Time series forecasting using neural networks

In this workshop you will fit forecasting models for time series data. This includes:

- Loading and graphically representing time series data.
- Restructuring the data to so that a time delay model can be fitted as a fully recurrent neural network.
- Fitting recurrent neural networks in the form of the Jordan and Elman RNNs using the package `RSNNS`.

A starter source file (`Workshop9.R`) and two data files are available in Moodle. Work through this worksheet in conjunction with the source code.

Exercise 1 Formatting and viewing time series data

Start R/R-Studio as usual. Load the libraries and read in the *shampoo sales* data using (and modifying) the code in the source file.

This dataset describes the monthly number of sales of shampoo over a 3 year period. The units are a sales count and there are 36 observations. The original dataset is credited to Makridakis, Wheelwright and Hyndman (1998). Note that the months are relative starting with year 1 month 01, so the actual dates are unavailable. As the shampoo sales data is quite short we will use all the data for the training set.

The following commands plot the time series, and rescale it. Note that the data show a strong negative autocorrelation with "lag" one. An increase is usually followed by a decrease and vice versa.

In order to get the data into a form for the TDNN you will use the function `lead()` from the `dplyr` library. There is a similar function called `lag()`. There are a couple of simple examples to demonstrate how these functions work. Note that the end of each output vector contains `NA`s. These usually need to be trimmed or extrapolated afterwards.

Exercise 2 Fitting a Time Delay NN

(a) The first NN Model uses no **no time delay**. The input is just the time series data using the value of the current quarter to predict the next quarter's sales.

► Run the code for this part step by step, which is complete. `mlp()` (multi-layer perceptron) is the function in `RSNNS` to fit a NN. The arguments `inputsTest` and `targetsTest` are not required to fit the NN, but they help for the function `plotIterativeError()`. This function plots the SSE for the test set (black) *and* the validation set (red) against training sweep iteration.

► Increase the number of nodes in the hidden layer, and try multiple hidden layers, and investigate the validation fit, and RMSE.

(b) The next model uses one time delay. The input data frame consists of two columns one with the current value and one with the previous value.

► The code to format the data is given. Note that the input data frame is one observation shorter, compared to when no time delay is implemented.

► As in part (a) investigate finding a combination of hidden nodes that gives a low RMSE, and compare the result to part (a)

(c) Repeat part (b) to fit a model with two time delays.

(d) With K-Step ahead forecasting, we often want to predict K values of the time series at each point. In this example K=2, so we define the output as a matrix with 2 columns; at time t we predict the values x_{t+1} and x_{t+2}.

Exercise 3 Sunspot data

This dataset describes a monthly count of the number of observed sunspots for just over 230 years (1749-1983). Each value is the number of daily sunspots averaged over the month and there are 2 820 observations. The source of the dataset is credited to Andrews & Herzberg (1985).

As these data contain the actual month and year, we can coerce the date text into a variable with class `Date`.

► Read in the data and define a test (the first 2000 observations) and a validation data set. These data show a strong cycle in the sunspot activity, but the period is not constant. Fit and compare a time delayed NN with no, one, two and three delays respectively.

Exercise 4 Recurrent NN using the Elman and Jordan models
In the lecture you learnt about the basic fully recurrent NN and the long short-term model. Another name for the fully recurrent NN is the Elman RNN (the name used in `RSNNS`).

The other type of recurrent model available in `RSNNS` is the Jordan RNN, which feeds the the output activation back to the hidden layer.

► Use the example code to fit and compare these models.