Semester Project Suggestion "Forecasting timeseries from business and finance"

Overview. Financial forecasting is a field of obvious economical relevance. It is also scientifically and intellectually stimulating, because financial timeseries are *extremely* hard to predict. The evolution of financial and business data is impacted by almost everything happening on this planet – politics, natural disasters, technological discoveries, fashions and everything else. Economical timeseries are as complex, stochastic, irregular and idiosyncratic as can be.

Machine learning researchers have been naturally attracted to develop prediction models for financial timeseries. Their models tend to be sophisticated, reflecting the state of the art in machine learning / neural networks. Independent and outside of machine learning research, a scientific field of *Financial Forecasting* has grown and matured over many decades in economics research departments, with its own traditions, methods, journals and conferences. The modeling methods in the forecasting field are generally simpler than what machine learners would use, rooting in elementary statistics and heuristic insight into the nature of financial dynamics.

I can see two reasons why machine learning / neural network models have not (yet?) made a big impact in forecasting. First, machine learning models (like, for instance, an LSTM network) are complex, handling them adequately needs specific professional training, and the obtained outputs are hardly interpretable. Thus, a forecaster cannot easily explain or justify the obtained forecasts to the economic decision-maker who pays his/her salary. Second, somewhat disconcertingly, the simpler models used in the forecasting community give results that are at least as good as the results obtained with sophisticated machine learning techniques, and often better. I found it very interesting to read more about this split between communities and methods in Green & Scott Armstrong (2015).

In the forecasting field there seems to be a classical benchmark dataset, the *M3 forecasting competition* dataset. It was published in the year 2000 and according to the Editor-in-Chief of the International Journal of Forecasting (IJF), Rob J. Hyndman, *"The M3 data have continued to be used since 2000 for testing new time series forecasting methods. In fact, unless a proposed forecasting method is competitive against the original M3 participating methods, it is difficult to get published in the IJF."* (cited from https://en.wikipedia.org/wiki/Makridakis_Competitions). The dataset consists of altogether 3003 relatively short timeseries, sorted into 24 groups according to the domain of the data (microeconomics, industry, macroeconomics, finance, demographic, other) and recording interval (monthly, quarterly, yearly, other). The sources of these timeseries are kept secret. The dataset can be downloaded from https://forecasters.org/resources/time-series-data/m3-competition/ as an Excel file, as well as forecasting results made with 24 standard forecasting methods of that time. Figure 1 shows a random pick from the M3 dataset. The difficulty of forecasting such timeseries is obvious.

The M3 competition was one in a series of similar competitions, held over decades (still ongoing), helping to establish the forecasting discipline, community, and standards. The organizers found it curious that only few submissions were based on machine learning methods, in particular neural networks; and competition entries that were based on such methods did not perform better (and mostly worse) than the "elementary" statistical and heuristic methods originating from within the forecasting community. In order to explore this phenomenon they also organized a parallel competition in the year 2005, the *NN3 competition* (http://www.neural-forecasting-competition.com/NN3/results.htm) where only methods from "computational intelligence" (AI and machine learning, most such entries were based on neural networks) were invited and ranked. The outcome was that *"the best ANN-based forecasts performed comparably with the best known forecasting methods, but were far more computationally intensive. It was also noted that many ANN-based techniques fared considerably worse than simple forecasting methods, despite greater theoretical potential for good performance."* (cited from https://en.wikipedia.org/wiki/Makridakis_Competitions). The final report of this investigation is given in Crone et al (2008).

At that time I organized a graduate seminar in my previous university, Jacobs University Bremen, in which we developed an RNN-based forecasting pipeline that we submitted to the competition a few minutes before the upload deadline. We did win the competition (https://www.ai.rug.nl/minds/teaching/highlight/), that is, our RNNs outperformed all other "computational intelligence" entries, but three "non-intelligent" standard forecasting methods, which were applied to the competition data for baseline results, performed better. And indeed, our method was computationally intensive – it combined the predictions made by an ensemble of 1000 neural networks.

My personal take-home message from all of this: financial (and related real-world) timeseries just contain very little information in the past that is informative about the future. Most of the prediction-relevant information can

be captured by exploiting generously averaged, smooth *trends*, combined with monthly, weekly, or daily *seasonality* components (weekend drops in power consumption, for instance). These two kinds of regularities are nicely visible plots 1, 2, 3, and 9 in Figure 1, and they can be extracted from the past of a timeseries with elementary methods.

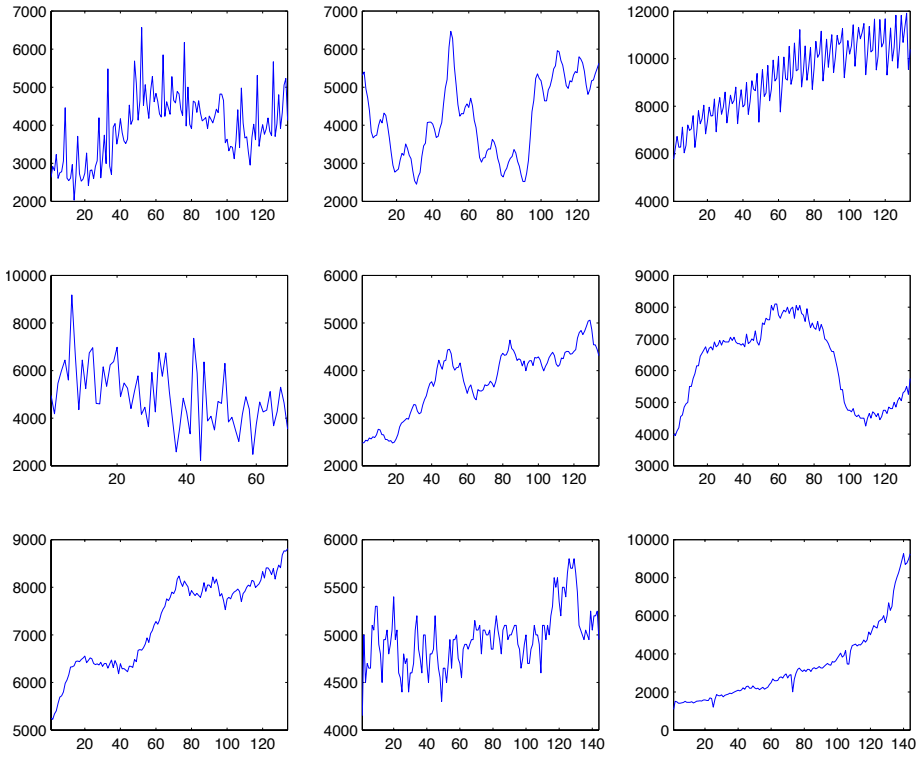

Figure 1. A random pick of 9 timeseries from the M3 competition dataset.

Given all of this background, I would be interested in exploring how *very simple* neural networks, with maybe only 2 – 5 hidden neurons, and skillfully regularized, can perform. Previous attempts in forecasting made by machine learners might have suffered from professional hubris – researchers in ML/NN tend to like "state of the art", intricate neural network architectures and maybe they just don't think of using very simple and small ones. Thus, my suggestion for this semester project is to try to come (at least) close to published results on the M3 competition, using *baby networks* in a clever way.

Suggestions for concrete project goals. Trying to do the full M3 competition is too much for a semester project. I suggest to pick one the 24 groups of data in the competition dataset and train a baby network to forecast the series found in that group. Some remarks:

- Financial forecasting is of obvious economical importance and thus has been researched for a long time. It is almost a scientific discipline of its own standing, with forecasting methods that are specifically tuned for financial timeseries. There is a standard textbook Makridakis, Wheelwright and Hyndman (see references at end) which gives a very elementary and readable introduction. You should at least read the first chapters and use the methods you find there ("de-trending" and "de-seasonalizing") to *preprocess* your data. These are elementary methods that have no machine learning or neural network flavor. After you have computed the "trend" and the "seasonality" components, subtract them from your original timeseries, retaining a *residual* timeseries which contains all the information (if any) that is difficult to predict and for which it makes sense to try squeezing predictive information out of them with neural networks (or other power-methods). The overfitting problem is particularly strong in financial data.
- The downloadable dataset contains the timeseries which at their end include the timepoints that have to be predicted, and which were withheld when the competition was originally run. The prediction horizon (number of final timepoints to be predicted) differs between the data groups. The information how many points are to be predicted can be found in the second column of the Excel file containing result predictions, downloadable from https://forecasters.org/resources/time-series-data/m3-competition/. When you train your models, *use only the data up to and excluding the prediction time points*, and only after you have tuned your models to the best level you can achieve (checked by cross-validation), test them on the prediction points. Specifically, if you set up a cross-validation scheme for model optimization, your validation data must come from the pre-prediction part of the timeseries that you downloaded.

- For quantifying the goodness of your predictions, use the metrics that are standard in the field. They are outlined in Appendix A of Makridakis and Hibon (2000).
- There are two principal ways to compute time series predictions, namely "jump-forward" predictions and iterated single-step predictions. For instance, if the task is to predict the next 6 data points, a NN trained according to the jump-forward scheme would have 6 output units which are trained on the 6 future time point data. An iterated single-step predictor network has only a single output unit which is trained to predict the single next value of the timeseries. In order to compute the second prediction point, the result from the first-timestep prediction is appended to the timeseries and a new input for the single-step prediction network is taken from the incremented timeseries. There is no general rule which of the two methods is preferable in which application scenario.
- You can use any type of ML model.
    - When you opt for MLPs, it is not obvious what should be the input vector to the network. You can use a fixed-size window of previous values, possibly unequally spaced, or you can first extract some self-designed features from the data (for instance average slope of the 5 previous time points) and use them as sole or auxiliary input.
    - When you use RNNs, the natural input signal is the very timeseries. However, the data range of the original data is unsuited for NNs, which like to work on data with limited amplitude, say in the range [0, 1] or [– 1, 1]. Thus you must first normalize the timeseries in some way.
- A very good idea is to first remove the *trend* from the data before feeding them to a NN predictor. The trend is a smoothed version of the original data. If the original timeseries is $u_1, \ldots, u_N$, the trend is another timeseries $v_1, \ldots, v_N$ which represents a smoothed version of $u_1, \ldots, u_N$. There are many ways to compute smoothed versions of a timeseries. For instance, you can approximate the original timeseries by a polynomial which then is the trend; or you can compute the Fourier transform of $u_1, \ldots, u_N$, cancel all high-frequency components and transform back to the time domain; or you can apply a smoothing filter (overview of smoothing methods in https://en.wikipedia.org/wiki/Smoothing).

  Once you have the trend $v_1, \ldots, v_N$, subtract it from $u_1, \ldots, u_N$ to get a *detrended* timeseries $d_1, \ldots, d_N = u_1 - v_1, \ldots, u_N - v_N$. Train your network only to predict the detrended timeseries, obtaining future detrended points $d_{N+1}, \ldots, d_{N+h}$. You also need to forecast your trend signal $v_1, \ldots, v_N$ to get trend predictions $v_{N+1}, \ldots, v_{N+h}$. How you can do this depends on the smoothing method that you used. The final prediction is then $u_{N+1}, \ldots, u_{N+h} = d_{N+1} + v_{N+1},, \ldots, d_{N+h} + v_{N+h}$.
- An idea worth following up is to train the network not on predicting the absolute next values, but the differences between successive values. This reduces the problem that NNs are not good at dealing with drifts or trends in the data (slow changes of local amplitude averages).
- Don't be frustrated when your predictor does not seem to give very nice-to-look-at results when you overlay your predictions with the correct continuations. Financial timeseries simply are almost impossible to predict. It is hard to be better than even the "naïve predictor" which essentially just repeats the last observed point, adjusted by a seasonal component. In stock trading you make money in the long term if your predictor is just a tiny little bit better than your competitors', in a statistical average over a large number of predictions. Note: don't try to get rich with financial forecasting: the fees charged by stock exchanges will eat up the small gains that your method might otherwise yield.

References

Crone, S. F., Nikolopoulos, K., Hibon, M. (2008). Automatic Modelling and Forecasting with Artificial Neural Networks – A forecasting competition evaluation. Final report for an IIF/SAS Grant. https://www.academia.edu/download/38607505/IIF-SAS-Report-Apr2008.pdf (also attached to the project suggestion package as file 3354_Croneetal08.pdf)

Green, K. C., Scott Armstrong, J. (2015), Simple versus complex forecasting: The evidence. Journal of Business Research 68, 1678–1685. http://www.academia.edu/download/54570109/Simple_versus_complex_forecasting_The_ev.pdf (included in package as file 3352_GreenArmstrong15.pdf)

Makridakis, S., Hibon, M. (2000), The M3-Competition: results, conclusions and implications. International Journal of Forecasting 16, 451–476. http://www.forecastingprinciples.com/files/pdf/Makridakia-The%20M3%20Competition.pdf (file 2038_MakridakisHibon00.pdf)

Makridakis, S., Wheelwright, S. C., Hyndman, R. J. Forecasting: Methods and Applications. John Wiley and Sons, Hoboken, NJ, 1998 (book available in the university library)