Deep Dynamic Factor Models for Stock Price Forecasting

Alessandro Ciancetta and Alessandro Tenderini

March 17, 2023

Abstract

Dynamic Factor Models (DFM) are a widespread class of linear models for macroeconomic and financial forecasting. We build on the framework introduced by Andreini, Izzo e Ricco (2020) to generalize the DFM to the non-linear case. The Deep Dynamic Factor Model (D2FM) that they propose uses autoencoders for approximating the non-linear mapping between the observed variables and the latent states. We extend the analysis using GRU networks for modelling the dynamics of the latent states. We calibrate the hyperparameters of the model by comparing the forecasting performances of the model on a grid of hyper-parameters. The calibrated model exhibits good forecasting performances on a dataset of daily prices of 978 stocks in the period from 2010-01-01 to 2023-03-01.

1 Introduction

A widely accepted theory in economics is that the co-movements of many financial and macroeconomic variables over time depend on a small number of underlying factors that drive the dynamics of the system. These factors can be either observable, as assumed for instance in the well-known Capital Asset Pricing Model or in the Fama-French model, or latent. A powerful class of latent-factor models is the Dynamic Factor Model (DFM). The DFM is a very common forecasting technique, especially in macroeconomics, with many central banks and financial institutions relying on it for their short-term forecasts. Roughly speaking, the DFM consists in: considering a large panel of time series; extracting some of its principal components; fitting a linear auto-regressive model on the principal components; get the predicted value of the latent state at the next period; mapping the prediction back to the original space.

Notice that the model works in two steps: first a static model maps the features to the latent space, then a dynamic model is fitted to get the predictions of the next state. A main advantage of this modeling strategy lays in its interpretability. Indeed, it allows to distinguish between the *systematic* and *idiosyncratic* components in the dynamics of the variables. The values predicted out of the latent states represent the systematic dynamics of the economy. Any departure from this prediction is variable-specific and represents an idiosyncratic shock affecting a single observable variables. This characteristic of the model is relevant both for investment and policy decisions, as, for example, it allows to understand if the price variation of an asset depends on a systematic trend or if it is specific to that asset.

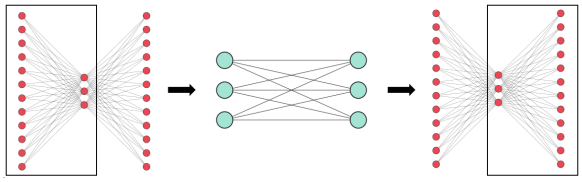
A main shortcoming of the DFM, however, is that it assumes linear relationships both in the static observed-latent and dynamic latent-latent mappings. To overcome this limit, Andreini, Izzo and Ricco (2020)[1] proposed a generalization of the the DFM based on the use of autoencoders for introducing a non-linear mapping between the observed variables and the latent space. The aim of this project is to extend this Deep Dynamic Factor Model (D2FM) to consider also a non-linear mapping between subsequent latent states. To do so, we implement a Gated Recurrent Unit (GRU) network to forecast the next realizations of the latent states estimated with the autoencoder. The model is tuned via an expanding-window forecast evaluation on a grid of hyper-parameters, and the calibrated model shows remarkable forecasting performances.[2]

2 The Deep Dynamic Factor Model

2.1 The origins: the linear Dynamic Factor Model

The DFM has the following state-space form:

$$x_t = \Lambda f_t + \xi_t$$
$$f_t = A_1 f_{t-1} + \cdots + A_p f_{t-p} + \eta_t$$



where x_t is the $n \times 1$ vector of observed variables in the dataset at time t, f_t is the $r \times 1$ vector of latent factors, Λ is the $n \times r$ matrix of factor loadings and $A_i, i = \{1, \ldots, p\}$ are the $r \times r$ autoregressive matrices defining the dynamics of the latent variables. Idiosyncratic terms and factors are assumed to be uncorrelated: $\mathbb{E}[f_{it}\xi_{js}] = 0, \forall i, j = 1, \ldots, n$ and $\forall t, s = 1, \ldots, T$.

[1] `https://arxiv.org/abs/2007.11887`

[2] We implemented the model in PyTorch without relying on any already-existing code. The code is attached to this report.

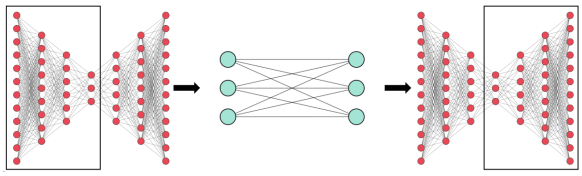
A common way of estimating the DFM is to first extract the first r principal components and their relative loadings from the data to get the estimators $\hat{f}_t, \hat{\Lambda}$, and to then fit a Vector Auto-Regressive (VAR) model to the principal components to get $\hat{A}_1, \ldots, \hat{A}_p$. This simple and straightforward method is not however the most efficient. Since direct maximization of the likelihood function of the problem is intractable, we can rather exploit the EM algorithm to maximize the complete likelihood of the model. Once the parameters of the model have been estimated, forecasts are obtained as

$$x_{t+1|t} = \hat{\Lambda}\hat{f}_{t+1|t}$$

with

$$\hat{f}_{t+1|t} = \hat{A}_1\hat{f}_t + \cdots + \hat{A}_p\hat{f}_{t-p+1}$$

2.2 The D2FM with linear latent dynamics



Andreini, Izzo e Ricco (2020) propose to generalize the DFM framework to account for possible non-linearities between the observed and latent variables. To do so, they consider the DFM as a special case of the following more general model:

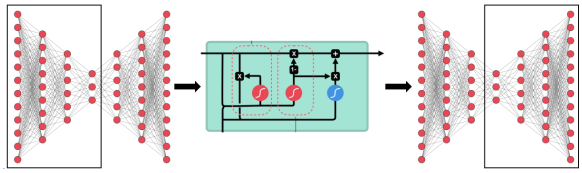
$$\begin{aligned} x_t &= D(f_t) + \xi_t \\ f_t &= A_1 f_{t-1} + \cdots + A_p f_{t-p} + \eta_t \end{aligned}$$

where D is a generic function mapping the r-dimensional latent space into the observed n variables. In general, D does not need to be linear, and the common component $D(f_t)$ can be rewritten as:

$$x_t = D(E(x_t)) + \xi_t$$

This form makes the connection between factor models and autoencoders quite natural. The map in the previous equation can be seen as an autoencoder, and the linear factor model is a special case that assumes linearity both in $D(\cdot)$ and $E(\cdot)$.

2.3 The generalized D2FM



A potential problem in the specification of the D2FM lays in the dynamics of the factors still being linear. Even if the autoencoder can retrieve significantly more complex structures linking the observed and the latent variables, subsequent observations of the factors could still be linked by more complex relationships than the linear autoregressive structure assumed by the model. In order to capture potential non-linear dynamics in the state equation, we generalize the model to the following form:

$$x_t = D(f_t) + \xi_t$$
$$f_t = R(f_{t-1}, \dots, f_{t-p}) + \eta_t$$

where $R(\cdot)$ is a generic function mapping the sequence of the p vectors $f_{t-1}, \dots, f_{t-p}$ to the subsequent observation f_t. In order to estimate this model, we use a GRU network designed to take in input a set of r sequences and to output an r dimensional vector predicting the next latent state.

3 Empirical exercise

We test the model using financial markets data from Yahoo Finance. We collected daily prices data of 978 stocks from 2010/01/01 until 2023/03/01. Stocks were selected from the NASDAQ, NYSE, and AMEX exchanges.

3.1 Calibration of the model

We calibrate the model with a grid search on different combinations of hyperparameters. We use the following expanding window forecast evaluation:

---

1. Train the model using observations $x_1, \dots, x_{T-\text{test size}}$
2. Predict $x_{T-\text{test size}+1}$
3. Add x_{t+1} to the training set
4. Repeat until the end of the sample

At each step, the model is retrained using also the new observation in the window. At the end of the procedure, we compare the 1-day ahead forecasts with the observed prices and pick the set of hyper-parameters minimizing the RMSE of the forecast. During the calibration, we considered a test set of 40 days. The details of the optimization procedure and the results of the calibration procedure are reported in Tab. 1 and 2. The optimization algorithm used is ADAM, with the default parameters provided in PyTorch. At each step, the number of epochs is 50, meaning that each time a new point is added to the training set, the parameters of the model are updated accordingly by going through the whole training set other 50 times.[3] We decrease the learning rate over the epochs according to an exponential-decay schedule equal to 0.99^{epoch}.

Optimization used in calibration	
Algorithm	ADAM (with default parameters)
Number of updating epochs	50
Batch size	500
Initial learning rate	0.01
Schedule	0.99 (Exponential decay)

Table 1: *Optimization details for the calibration of the model based on the expanding-window forecast evaluation.*

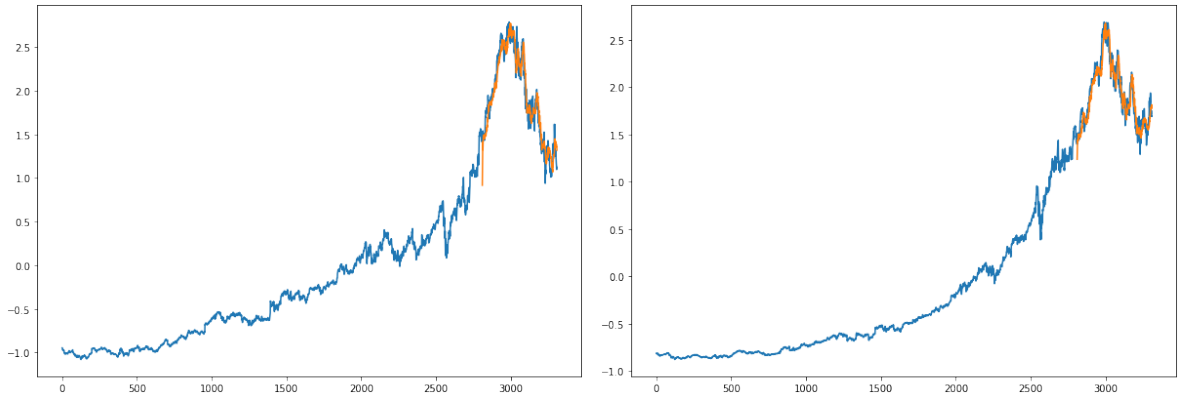
3.2 Forecast evaluation

We evaluate the calibrated model on the last 500 observations available in the dataset. The average RMSE across the scaled stock prices is 0.29, which is a fairly good result for this kind of exercise. Visual inspection of the series and the predictions in Fig. 1 shows how close the predictions are to the observed prices in the case of the stock price of Google and Microsoft. However, a closer look to the test period in Fig. 2 also shows some limits of the model. In particular, the forecasts show a worrying delay with respect to the realized prices. This means that even if the model is always quite close to the true realized value (the maximum forecast error in the figure corresponds to about half of the standard deviation of the series), still it is not able to anticipate the movements of the series of prices precisely.

[3]Of course, a better calibration would rely on a wider test set for longer training time. However, since we ran the model on our local machines, this was particularly hard due to the computational burden. We ran sequentially the calibration and the final forecast evaluation training the calibrated model with 100 epochs for each 1-day ahead prediction on 500 subsequent test points. This exercise took about 16 hours to run on a machine with CPU Intel i7-11370H and GPU NVIDIA GeForce RTX 3050.

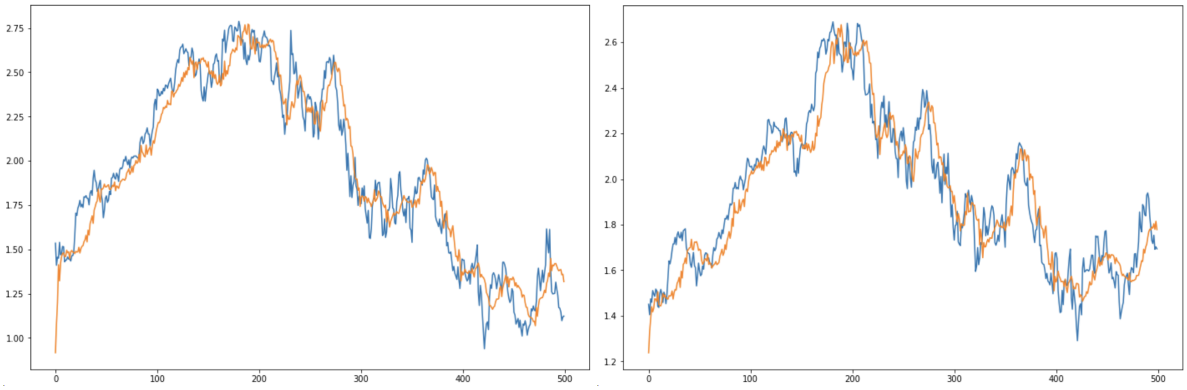
Model structure	Hyperparameter	Choice taken
Autoencoder	Number of hidden layers	4
	Number of neurons	500, 200, 100, 50
	Dimension of the latent state	50
	Decoder type	Symmetric
GRU	Sequence length	7
	Hidden size	128
	Number of layers	2

Table 2: *Specification of the calibrated model obtained via forecast evaluation on a grid of hyperparameters.*



(a) *Google stock price (standardized)* (b) *Microsoft stock price (standardized)*

Figure 1: *Observed and predicted prices using the calibrated D2FM. The blue line denotes the observed data, both in the initial training set and in the testing period. The orange line denotes the 1-day ahead forecast obtained by training the model using the data up to the day before and getting the output of the model from the sequence of the last week.*



(a) *Google* (b) *Microsoft*

Figure 2: *Observed and predicted prices in the test set. The blue line denotes the observed data, the orange line denotes the 1-day ahead forecasts.*

4 Conclusions and further work

In this project, we implemented an extension to the D2FM proposed by Andreini, Izzo and Ricco (2020), by considering a GRU architecture for predicting subsequent latent states obtained from the autoencoder. We conducted an empirical exercise using daily data about the prices of 978 stocks over the last 13 years. The hyperparameter of the model are chosen via evaluation of the out-of-sample performances of the model on a grid of hyperparameters. The calibrated model is shown to have good forecasting performances, even if some examples show that the forecasts tend to have a delay over the realized prices. A potential solution to this problem could consist in refining the calibration and training of the model, by considering a wider grid of hyperparameters and training the model for more epochs. Another potentially useful extension involves introducing an attention mechanism in the prediction of the latent state. Implementing a transformer could indeed endow the model with longer memory, thus allowing it to take advantage of medium- and long-term patterns of the business cycle for achieving improved forecasting performances.