Nonlinear time series forecast using the artificial neural network and the local linear regression combined with dimensionality reduction techniques

Simon Jung (425108)
Bachelor Thesis (Econometrics and OR); supervised by M. Girth and assessed by D.J.C. van Dijk
Erasmus School of Economics, Erasmus University Rotterdam

Abstract

This paper gives a detailed guide to building a forecasting method when high-dimensional predictors and possible nonlinear issues exist. From the predictors, a certain number of factors are extracted which will form predictive indices using the sliced inverse regression. This paper considers mainly two models that can be applied in nonlinear time series forecast: the artificial neural network (ANN) and the local linear regression (LLR). The paper describes a detailed procedure for building and training ANNs customized for time series forecast that uses the aforementioned dimension reduction techniques. In the simulation studies, by various evaluation criteria we examine the in-sample and out-of-sample performance of the ANNs and LLRs, which will also be compared to the conventional ordinary least squares (OLS). We find that when nonlinearity, such as interaction between factors, exists, the ANNs and LLRs perform superior to the OLS while the OLS shows the best performance in presence of the linearity. In the empirical application, the ANN and LLR also show superior forecasting performance compared to the OLS.

1. Introduction

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces. It is a phenomenon econometricians and deep learning developers worry about. In the deep learning development, as the dimension of inputs gets higher, the total number of required training sets exponentially increases. In econometrics, a large set of variables is also an obstacle for the ideal regression analysis, since it increases not only the required amount of data but also the estimator's variances, which deteriorate the reliability of the estimators. Studies in computer science have pointed out numerous algorithms for input variable selection, which typically consist of filtering unnecessary variables. In econometrics, there are several variable selection tests and methods that let you know which variables can be ignored.

However, in the era of big data, ignoring seemingly unnecessary variables does not seem to be a smart choice. The best option would be extracting a core information across all available variables and then creating new variables that represent this extracted information. The factor model is the conventional way to do this. Through factor models, the original dimension of the variables, p, can be reduced to the number of factors, K. Another benefit of using factor models that should not be underestimated is that it can remove noise, confounding, measurement errors to some extent. Because we extract common features from the variables, the individual errors can be effectively ignored. There has been put much effort into applying factor models in economics theories and time series forecasting, for example Fama and French (2015) suggest the five-factor model in financial asset pricing and Schumacher and Breitung (2008) adopt a factor model for short-term forecasting of the German GDP growth. Stock and Watson (2002) considers forecasting a single time series using factors estimated by the principal component analysis. There exist various methods for estimating factors, for example Forni et al. (2000) develop the generalized dynamic factor model and Kapetanios and Marcellino (2009) suggest a parametric estimation method for factors that is still computationally feasible for a very large data set. Doz et al. (2011) show a two-step estimation of the factors in a dynamic approximate factor model, in which the parameters of the model are estimated from an OLS on principal components and then the factors are estimated via the Kalman smoother.

Yet, this factor-based approach is limited to linear forecasting and does not take into account the information of the target (Fan et al., 2017). Li (1991) establishes the sliced inverse regression (SIR), in which we first find a covariance matrix conditioned on the target and then form the predictive indices by linear combination of variables according to the eigenvectors of the conditional covariance matrix. Following this, Fan et al. (2017) suggested a new dimension reduction method using both a factor model and the sliced inverse regression. Here, we make L predictive indices out of factors that were first extracted from predictors. In this way, we are able to dramatically reduce the dimension of the predictors while keeping, or even improving, the predictive power.

The Artificial Neural Network (ANN) is a computing framework for modeling a broad range of nonlinear functions. One significant advantage of the ANN models over other classes of nonlinear models is that ANNs are universal approximators which can approximate a large class of functions with a high degree of accuracy (Zhang, 2003). With one hidden layer, we can represent any continuous function of the input signals,

and with two hidden layers even discontinuous functions can be represented (Negnevitsky, 2005). It does not impose any structural constraints on the data-generating process and the estimation can be done flexibly. There are a number of papers that discuss an application of ANNs in time series forecasting. Zhang (2003) suggests a hybrid methodology that combines both ARIMA and ANN models and Kaastra and Boyd (1996) give a detailed procedure for designing an ANN architecture for time series forecasting. Zhang and Qi (2005) investigate the issue of how to effectively model time series with both seasonal and trend patterns. Zhang and Berardi (2001) use a modified version of the neural network applied in predicting the exchange rate.

Besides the ANN, we will use the local linear regression (LLR) (Fan and Gijbels, 1996) to capture the nonlinearity of the data. The LLR is a popular tool used in nonparametric regression analysis in which at each point in the range of the data set a linear function is fitted into a subset of the data. This nonparametric regression does not require the specification of a function to fit a model to all of the data in the sample. Instead we only have to determine the length of the bandwidth and kernel function to use. Like ANNs, the LLR can be fitted very flexibly when we have a small bandwidth. In certain cases, the local linear regression performs even better than the ANN as shown by Shamim et al. (2016).

All the time series forecasting by the nonlinear models have confronted the curse of dimensionality. As mentioned before, as the dimension of inputs gets higher, the amount of data required to get a sufficient forecasting performance exponentially increases. Unfortunately, the amount of data is often limited in the financial sector. For this reason, the aforementioned dimension reduction technique consisting of the factor model and SIR will be very useful for time series forecasting by an ANN or LLR. A few papers adopt either the factor model or the sliced inverse regression as a way of reducing dimension of inputs; see Bai and Ng (2008), Yuan and Fine (1998, 1993).

2. ANN forecasting with dimension reduction

2.1. ANN forecasting with factor model and sliced inverse regression

See the following factor model with a target variable y_{t+1} that we want to forecast and x_{it} that is the i_{th} predictor at time t:

$$y_{t+1} = g(\boldsymbol{\phi}_1'\mathbf{f_t}, \ldots, \boldsymbol{\phi}_L'\mathbf{f_t}, \epsilon_{t+1}), \tag{1}$$

$$x_{it} = \mathbf{b_i'f_t} + u_{it}, \qquad 1 \le i \le p,\ 1 \le t \le T, \tag{2}$$

where p and T are respectively the number of predictors and the number of observations. b_i is a $K \times 1$ vector of factor loadings and $\mathbf{f_t} = (f_{1t}, \ldots, f_{Kt})'$ is a $K \times 1$ vector of common factors across the predictors. Other than the factors and loadings, x_{it} also has an error term, or an idiosyncratic component, represented by u_{it}. Target variable y_{t+1} is constructed by the unknown link function $g(\cdot)$ and the predictive indices $\boldsymbol{\phi}_1'\mathbf{f_t}, \ldots, \boldsymbol{\phi}_L'\mathbf{f_t}$. $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_L$ are orthogonal vectors of linear combinations in K-dimension which are called the sufficient dimension reduction (SDR) directions (Fan et al., 2017). ϵ_{t+1} is a stochastic error independent of $\mathbf{f}_t$ and u_{it}.

As we can see in model 1, the target variable only depends on the L predictive indices. Of course, we expect the number L to be lower than the number of the predictors, p, and the factors, K. The perfectly reduced variables, $(\boldsymbol{\phi}_1'\mathbf{f_t}, \ldots, \boldsymbol{\phi}_L'\mathbf{f_t})$, are seen to be as informative as the original $\mathbf{x}_t$, so these predictive indices are sufficient in forecasting y_{t+1}. Models 1 and 2 make it possible to reduce the dimension from p to L. The estimation methods for factors and SDR directions are given in section 2.2.

We still need to estimate the unknown link function $g(\cdot)$. This is a nonparametric function with the number of parameters specified but without any structural conditions imposed. Most previous researches restricted their estimations of the function into the linear format. However, these linear functions will not show satisfying performance when $g(\cdot)$ is actually nonlinear. Fan et al. (2017) establish a unique forecasting method, named *sufficient forecasting*, that takes the possible nonlinearity of $g(\cdot)$ into account while using the factor model and SIR. The *sufficient forecasting* adopts the LLR (Fan and Gijbels, 1996) to estimate $g(\cdot)$. Another possible option would be the ANN, which will be mainly discussed in this paper. In section 3 and 4, we will see which nonlinear model shows the best performance. The procedure of this forecasting method is given in algorithm 1. The prototype of the ANN forecasting architecture with the predictive indices is illustrated in figure 1.

Algorithm 1: ANN forecasting using factor models

Step 1: Obtain the estimated factors $\{\widehat{\mathbf{f}}_t\}_{t=1,\ldots,T}$

Step 2: Obtain the estimated SDR directions, $\widehat{\boldsymbol{\phi}}_1, \ldots, \widehat{\boldsymbol{\phi}}_L$

Step 3: Construct the predictive indices $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_\mathbf{t}, \ldots, \widehat{\boldsymbol{\phi}}_L'\widehat{\mathbf{f}}_\mathbf{t}$

Step 4: With the predictive indices from Step 3, use the ANN or the LLR to estimate $g(\cdot)$ and forecast y_{t+1}.

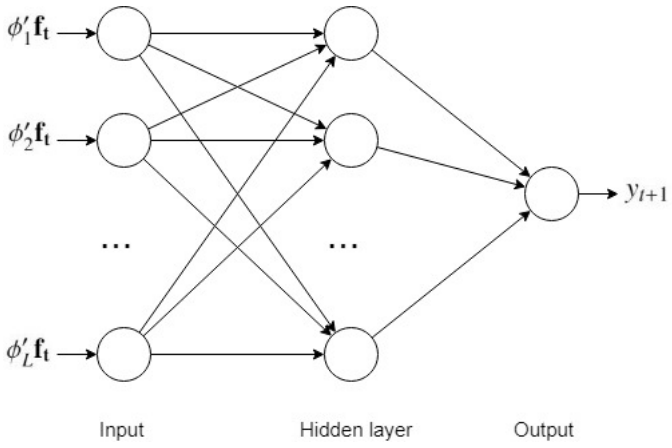

Figure 1: ANN architecture using factor model

2.2. Estimation of factors and SDR directions

In practice, the factors, factor loadings and SDR directions in models 1 and 2 should be estimated. The factors and loadings can be estimated as follows:

$$(\widehat{\mathbf{B}}_K, \widehat{\mathbf{F}}_K) = \underset{(\mathbf{B},\mathbf{F})}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{B}\mathbf{F}'\|_F^2 \tag{3}$$

subject to

$$T^{-1}\mathbf{F}'\mathbf{F} = \mathbf{I}_K, \qquad \mathbf{B}'\mathbf{B} \text{ is diagonal,}$$

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$, $\mathbf{F}' = (\mathbf{f}_1, \ldots, \mathbf{f}_T)$, and $\|\cdot\|_F$ denotes the Euclidean norm. Note that $\mathbf{X}$ is a $P \times T$ matrix and its t_{th} column, $\mathbf{x}_t$, is a vector of P dimension. Loading matrix $\mathbf{B}$ and factor matrix $\mathbf{F}$ are $P \times K$ and $T \times K$ matrices, respectively. This is a typical principal components problem. The columns of $\widehat{\mathbf{F}}_K/\sqrt{T}$ are the eigenvectors corresponding to the K largest eigenvalues of the $T \times T$ matrix $\mathbf{X}'\mathbf{X}$ and $\widehat{\mathbf{B}}_K = T^{-1}\mathbf{X}\widehat{\mathbf{F}}_K$. One can use some other robust estimations for factors and loadings such as Forni et al. (2000).

Traditional analysis of factor models typically focuses on the covariance with the forecast target $\operatorname{cov}(\mathbf{x}_t, y_{t+1})$ and the covariance within the predictors $\mathbf{x}_t$, denoted by a $p \times p$ matrix

$$\Sigma_x = \mathbf{B}\operatorname{cov}(\mathbf{f}_t)\mathbf{B}' + \Sigma_u \tag{4}$$

where Σ_u is the error covariance matrix of $\mathbf{u}_t$. However, this approach does not fully utilize the information of the target variable, y_{t+1}. So, in this research we will consider $E(\mathbf{x}_t|y_{t+1})$ that is the conditional expectation value of the predictors given the target variable.

Under model 1, Li (1991) showed that $E(\mathbf{f}_t|y_{t+1})$ is contained in the central subspace $S_{y|\mathbf{f}}$ spanned by $\boldsymbol{\phi}_1'\mathbf{f_t}, \ldots, \boldsymbol{\phi}_L'\mathbf{f_t}$. So, it is reasonable to estimate sufficient directions by investigating the top L eigenvectors of $\operatorname{cov}(E(\mathbf{f}_t|y_{t+1}))$. Estimation method for $\operatorname{cov}(E(\mathbf{f}_t|y_{t+1}))$ is developed by Li (1991) and is called the sliced inverse regression (SIR). In this estimation, we divide our data set into slices $I_1, \ldots, I_H$ such that the proportion of the y_t falls in slice I_h is $1/H$. Then we substitute $E(\mathbf{f}_t|y_{t+1})$ with $E(\mathbf{f}_t|y_{t+1} \in I_h)$ which leads to the following:

$$\Sigma_{f|y} = \frac{1}{H}\sum_{h=1}^{H} E(\mathbf{f}_t|y_{t+1} \in I_h)E(\mathbf{f}_t'|y_{t+1} \in I_h) \tag{5}$$

By conditioning on the target y_{t+1} in model 2, we obtain

$$\begin{aligned}\Sigma_{x|y} = \operatorname{cov}(E(\mathbf{x}_t|y_{t+1})) &= \operatorname{cov}(E(\mathbf{B}\mathbf{f}_t + \mathbf{u}_t|y_{t+1})) \\ &= \mathbf{B}\operatorname{cov}(E(\mathbf{f}_t|y_{t+1}))\mathbf{B}'\end{aligned}$$

and this gives us another form of 5:

$$\Sigma_{f|y} = \frac{1}{H}\sum_{h=1}^{H} \mathbf{\Lambda}_b E(\mathbf{x}_t|y_{t+1} \in I_h)E(\mathbf{x}_t'|y_{t+1} \in I_h)\mathbf{\Lambda}_b', \tag{6}$$

where $\mathbf{\Lambda}_b = (\mathbf{B}'\mathbf{B})^{-1}\mathbf{B}'$.

The simplest way to estimate the conditional covariance matrices in 5 and 6 would be to find $\widehat{\mathbf{f}}_t$ from model 3 and then respectively replace $E(\mathbf{f}_t|y_{t+1} \in I_h)$ and $E(\mathbf{x}_t|y_{t+1} \in I_h)$ with the sample mean of $\widehat{\mathbf{f}}_t$ and $\mathbf{x}_t$ within each slice. We can denote the ordered statistics of $\{(y_{t+1}, \widehat{\mathbf{f}}_t)\}_{t=1,\ldots,T-1}$ by $\{(y_{(t+1)}, \widehat{\mathbf{f}}_{(t)})\}_{t=1,\ldots,T-1}$ according to the values of y, where $y_{(2)} \le \cdots \le y_{(T)}$. Then we divide the range of y into H slices, where the first $H-1$ slices contain the same number of observations $c > 0$ and the last slice may have less than c observations. The ordered statistics which are sliced are denoted as follows:

$$\{(y_{(h,j)}, \widehat{\mathbf{f}}_{(h,j)}) : y_{(h,j)} = y_{(c(h-1)+j+1}, \widehat{\mathbf{f}}_{(h,j)} = \widehat{\mathbf{f}}_{c(h-1)+j}\}_{h=1,\ldots H;\ j=1,\ldots c}.$$

Then the estimators are as follows:

$$\widehat{\Sigma}^1_{f|y} = \frac{1}{H}\sum_{h=1}^{H}\left[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}\right]\left[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}\right]' \tag{7}$$

$$\widehat{\Sigma}^2_{f|y} = \widehat{\mathbf{\Lambda}}_b\left(\frac{1}{H}\sum_{h=1}^{H}\left[\frac{1}{c}\sum_{l=1}^{c}\mathbf{x}_{(h,l)}\right]\left[\frac{1}{c}\sum_{l=1}^{c}\mathbf{x}_{(h,l)}\right]'\right)\widehat{\mathbf{\Lambda}}_b', \tag{8}$$

where $\widehat{\mathbf{\Lambda}}_b = (\widehat{\mathbf{B}}'\widehat{\mathbf{B}})^{-1}\widehat{\mathbf{B}}'$. $\widehat{\Sigma}^1_{f|y}$ is a factor-based estimator and $\widehat{\Sigma}^2_{f|y}$ depends on observations and loadings. These two estimators look different at glance, but they converge to the same value as the number of observations of a slice goes to infinity. Either with many total observations or a great number of slices, the difference between the two estimators becomes negligible. The estimation of the SDR directions $\widehat{\boldsymbol{\phi}}_1, \ldots, \widehat{\boldsymbol{\phi}}_L$ are the eigenvectors of $\widehat{\Sigma}_{f|y}$ corresponding to the L largest eigenvalues.

Theorem 1. *The estimators $\widehat{\Sigma}^1_{f|y}$ and $\widehat{\Sigma}^2_{f|y}$ converge to the same value as $c \to \infty$.*

Remark. *As an exception, $\widehat{\Sigma}^1_{f|y}$ and $\widehat{\Sigma}^2_{f|y}$ are always equivalent under the factor model 2 (Fan et al., 2017). However, if you use another factor model estimation method such as the generalized factor model (Forni et al., 2000), theorem 1 holds.*

2.3. ANN design principles for time series forecasting

Although ANNs are widely used, there is no specific rule for designing the architectures. Thus, in sections 2.3 and 2.4, we present some practical methods for designing and training the ANN customized for time series forecasting.

Typically, ANNs consist of an input and an output layer, joined by hidden layers. Each layer has neurons that have an activation function and a threshold. The numbers of neurons in the input layer and in the output layer are equal to the number of inputs and outputs of the data set, respectively. The hidden layers can be freely designed by the users although there exists a rule of thumbs. All neurons in each layer are perfectly connected with the neurons of their neighbor layers with specified weights.

Let us say that we have L inputs and one output with s hidden layers $HL_1, ..., HL_s$. The h^{th} hidden layer HL_h has m_h neurons. Then the number of parameters (weights and thresholds) is $Lm_1 + I(s > 1)\sum_{h=1}^{s-1} m_h m_{h+1} + m_s + \sum_{h=1}^{s} m_h + 1$. As we can see, the number of parameters exponentially increases with the number of hidden layers. This means that the amount of

the data we must obtain for a proper training also exponentially increases. This is the curse of dimensionality. With the presence of a large number of parameters, we can face the notorious over-fitting problem, in which the model fits very well in a training set but fails to fit additional data or to predict future observations reliably. Hence, in designing an ANN, it is highly recommended to follow Occam's razor principle *the simpler the better*. In practice, a three-layer ANN (one hidden layer with input and output layers) is mostly used and four-layer in certain cases. There are also ANNs with more than four layers, but they are mostly experimental. In a financial time series, using only one hidden layer is recommended, as we usually focus on forecasting rather than fitting a model into data. In this paper, we will deal with a three-layer ANN in which the number of inputs and output is L and 1, respectively. The number of neurons in the hidden layer is usually chosen from in-sample simulations or cross-validations. In sections 3 and 4, we will see which one gives the best in-sample and out-of-sample performances among ANNs of one hidden layer with different numbers of hidden neurons.

The input layer accepts the inputs and distributes them into the neurons in the hidden layer. Then a neuron in the hidden layer accepts the value computed with the weights and creates its intermediate return value with the threshold and the activation function. The neuron in the output layer accepts the values the same way and returns the final output of the network.

An ANN without activation functions would simply be a linear regression model. In neural networks, the activation function of a neuron, which is an abstraction representing the rate of action potential that is firing in the cell, defines the output of the node given inputs. The reason of using the activation function is to prevent outputs from reaching very large values which could paralyze the neural network and thereby hinder training. Linear activation functions are not useful for nonlinear mapping and classification (Kaastra and Boyd, 1996). A number of researches such as Hsieh (1991) suggest that the data-generating process in the financial market is nonlinear, so nonlinear activation functions would be more appropriate in forecasting a financial time series. Among the nonlinear activation functions, the sigmoid $\varphi(x) = 1/(1 + e^{-x})$ is commonly adopted because it is nonlinear and differentiable in the whole domain. However, outputs of sigmoid function range between -1 and 1, so it is not appropriate to use it for the output neuron since our outputs can be out of the range. Hence, a neural network with sigmoid hidden layer neuron activation function and identity output neuron activation function $\varphi(x) = x$ is the most popular choice for many successful applications of financial forecasts (Qi and Zhang, 2001).

In conclusion, in a case of univariate time series forecasting, the ANN consists of one input layer of L neurons and an output layer of one neuron. The neurons in the hidden layer have the sigmoid activation function and the output layer neuron has the identity activation function. We will use only one hidden layer, in order not to have over-fitting problems, and the hidden layer has m hidden neurons. So, in this three-layer ANN, we have $Lm + m$ weights and $m + 1$ thresholds. These $Lm + 2m + 1$ parameters should be estimated from training.

2.4. ANN training

In supervised learning, the ANN is iteratively presented examples of the correct known answers. The aim of the supervised learning is to find the set of weights between the neurons that minimizes the error function. Mostly the error function is defined as the mean squared errors. Let us say we have a three-layer ANN. The number of neurons in the input, hidden and output layers is respectively L, m and 1. We have L input signals $\boldsymbol{\phi}_1'\mathbf{f_t}, \ldots, \boldsymbol{\phi}_L'\mathbf{f_t}$ which are created from the factor model and the SIR in section 2.2. For ease of notation, we denote $\boldsymbol{\phi}_1'\mathbf{f_t}, \ldots, \boldsymbol{\phi}_L'\mathbf{f_t}$ as $p_{1t}, \ldots, p_{Lt}$ and the vector of these as $\mathbf{p}_t$. The indices l and j refer to neurons in the input and hidden layers, respectively. The sigmoid activation function will be used in the hidden layer and the identity function in the output neuron.

Before starting the training, we need to initialize the weights and thresholds with random numbers. Haykin (2009) suggests that these parameters can be set to random numbers drawn from the uniform distribution $(-2.4/F_i,\ 2.4/F_i)$ where F_i is the total number of inputs of neurons i, i.e. the number of neurons in its left neighbor layer. In our case, F_i for the hidden and output neurons are L and m, respectively.

When we input a training set into the network, the ANN propagates $p_{1t}, \ldots, p_{Lt}$ to the hidden layer. The value a hidden neuron receives is the net-weighted input subtracted by the threshold:

$$P_{jt} = \sum_{l=1}^{L} p_{lt} w_{lj} - \theta_j \qquad \text{for } j = 1, \ldots, m,$$

where θ_j is the threshold of hidden neuron j, and w_{lj} is the weight between neuron l in the input layer and neuron j in the hidden layer. Next, this net-weighted value is passed through the sigmoid activation function:

$$y_{jt}^H = \varphi(P_{jt}) = \frac{1}{1 + e^{-P_{jt}}} \qquad \text{for } j = 1, \ldots, m.$$

Figure 2 depicts how the neuron j in the hidden layer takes this procedure. These intermediate returns $y_{1t}^H, \ldots, y_{mt}^H$ are used for the output neuron to calculate the final output of the ANN. Note that the output neuron uses not the sigmoid but the identity activation function, so the net-weighted intermediate result with the threshold subtracted is the output. Our final output is

$$\hat{y}_{t+1} = \sum_{j=1}^{m} y_{jt}^H w_j - \theta,$$

where w_j is the weight between neuron j and the output neuron and θ is the threshold of the output neuron. The difference between the output $\hat{y}_{t+1}$ and the actual value y_{t+1} is simply denoted as $e_t = y_{t+1} - \hat{y}_{t+1}$.

We have seen that the inputs are processed through the network. However, the parameters, weights and thresholds are nothing more than random numbers, so we have to adjust them to the optimal values that minimize the error function. Backpropagation is a training method used in ANN to calculate a

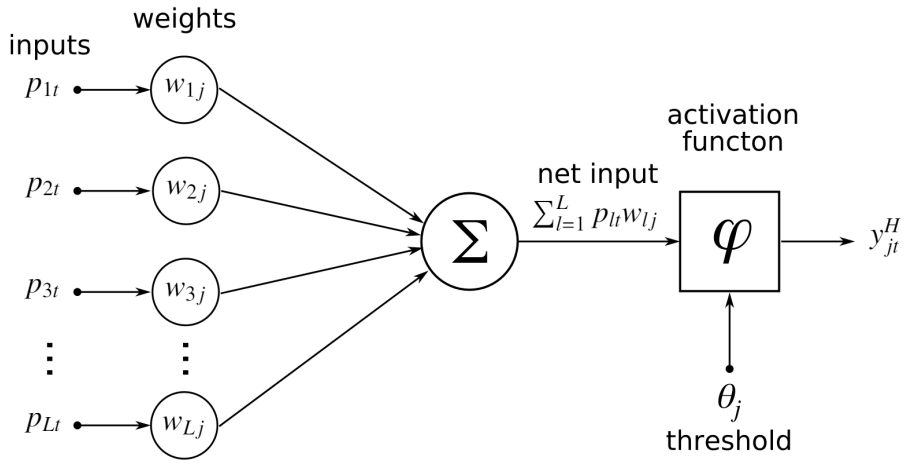

Figure 2: Propagation from the inputs to the j^{th} neuron in the hidden layer. Intermediate return y_{jt}^{H} is used to calculate the final output.

gradient that is needed for updating the weights and thresholds. It is called back-propagation as it propagates the errors backward and makes the parameters update accordingly. In this method, the parameters in the back side are updated first.

We calculate the error gradient for the neuron in the output layer. The error gradient in the output layer is determined as the derivative of the activation function multiplied by the error at the neuron output, i.e. $\delta = e_t \varphi'(y_{jt}^{H})$. With identity activation, the error gradient is equal to the error $y_{t+1} - \hat{y}_{t+1}$. The weights and threshold corrections are

$$\Delta w_j = \alpha \times y_{jt}^{H} \times \delta$$
$$\Delta\theta = \alpha \times (-1) \times \delta$$

and we update the weights and thresholds

$$w_j \leftarrow w_j + \Delta w_j$$
$$\theta \leftarrow \theta + \Delta\theta$$

Now we calculate the error gradient for the neurons in the hidden layer:

$$\delta_j = y_{jt}^{H} \times (1 - y_{jt}^{H}) \times \delta \times w_{ij}$$

Here the term $\delta \times w_{ij}$ is the error signal for which neuron j is responsible and this is multiplied by the derivative of the activation function at the neuron. The error corrections are as follows:

$$\Delta w_{lj} = \alpha \times p_l \times \delta_j$$
$$\Delta\theta_j = \alpha \times (-1) \times \delta_j,$$

and we update the weights and thresholds as before:

$$w_{lj} \leftarrow w_{lj} + \Delta w_{lj}$$
$$\theta_j \leftarrow \theta_j + \Delta\theta_j$$

We continue all these steps through the entire training data sets. One main difference of ANN training from the model estimation in econometrics is that the same data sets are iteratively being processed multiple times until we find a set of reasonable parameters. The training sets constitute one epoch which is iteratively used for training. After the training, we select the parameter set that has the lowest value of the error function. Unlike econometric nonlinear or linear regressions, the ANN training results in different estimation values of the parameters depending on the initialization.

When it comes to the number of iterations of training, Kaastra and Boyd (1996) show two main schools of thought. The first one stresses that the training should stop if there is no improvement in the error function a certain number of times. Here, the point at which the network does not improve is called convergence. The second one argues that the training should stop after a predetermined number of iterations.

Remark. *One can use modified back-propagation algorithms which will give you more efficiency of training. The back-propagation algorithm used in this paper is the most basic version which is rarely used in practice for its inefficiency. One can include a momentum constant that would incur a stabilizing effect on training (Watrous, 1987). Allowing learning rate α to change by heuristic methods can also speed up the training (Jacobs, 1988).*

2.5. Local linear regression

One can adopt another estimation method for the link function $g(\cdot)$ in model 1. One appropriate option would be the local linear regression which does not require the specification of a function. In LLR, we only have to determine the bandwidth parameter h and the kernel function to use. In addition, the LLR is very flexible, making it ideal for modeling nonlinear relations for which no theoretical models exist.

For the predictive indices, we use the same notation as in the last section, i.e., $p_{lt} = \boldsymbol{\phi}_l' \mathbf{f}_t$ and $\mathbf{p}_t = (\boldsymbol{\phi}_1' \mathbf{f}_t, \ \ldots, \ \boldsymbol{\phi}_L' \mathbf{f}_t)'$. The conventional linear regression model for model 1 is given by

$$\begin{aligned} y_{t+1} &= \beta_0 + \beta_1 p_{1t} + \ \ldots \ + \beta_L p_{Lt} + \epsilon_{t+1} \\ &= (1, \ \mathbf{p}_t') \, \boldsymbol{\beta} + \epsilon_{t+1}, \end{aligned}$$

where $\boldsymbol{\beta} = (\beta_0, \ \beta_1 \ \ldots, \ \beta_L)'$. The parameters in this linear model are globally fixed, therefore we can say this model is a global fitting.

On the other hand, the local linear regression allows the parameters $\beta_0,\ \beta_1,\ \ldots,\ \beta_L$ to change depending on the value of the predictive indices. Then the model is structured as follows:

$$y_{t+1} = \beta_0 + \beta_1(p_{1t})\ p_{1t} \ldots + \beta_L(p_{Lt})\ p_{Lt} + \epsilon_{t+1}$$

Here, the value of $\beta_l(\cdot)$ is a function of predictive index p_{lt} for $l = 1,\ \ldots, L$.

The estimation of the multivariate LLR model is based on the following minimization problem:

$$\min_{\boldsymbol{\beta}} \sum_{t=1}^{T-1} \{y_{t+1} - \beta_0 - \sum_{l=1}^{L} \beta_l(p_{lt} - p_l)\}^2 \frac{1}{|\mathbf{B}|} K(\mathbf{B}^{-1}(\mathbf{p}_t - \mathbf{p})) \quad (9)$$

where $K(\cdot)$ is the kernel function, and $\mathbf{B}$ and $|\mathbf{B}|$ are respectively the bandwidth matrix and its determinant. Bandwidth matrix $\mathbf{B}$ is an invertible $L \times L$ matrix, which is simply

$$\mathbf{B} = \text{diag}\{h_1,\ \ldots, h_L\}.$$

Here, h_l is the bandwidth for the l^{th} variable, which specifies how many nearest neighbors will be included for the local fitting. Note that here $\mathbf{p}$ and p_l are random variables of $\mathbf{p}_t$ and p_{lt}. The elements of $\boldsymbol{\beta}$ in 9 depend on $\mathbf{p}$. In this sense, $\boldsymbol{\beta}$ can be considered a function of $\mathbf{p}$.

Several types of kernel functions are commonly used: uniform, triangle, Epanechnikov, quartic, tricube, triweight, Gaussian, quadratic and cosine, among which Gaussian is the most popular (Fan and Gijbels, 1996). The Gaussian kernel is defined as:

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}, \quad (10)$$

where u is a real number. The structure of kernel functions are based on a symmetric probability density function. Hence, in most applications they satisfy the following two constraints:

1. $\int K(u)du = 1$
2. $K(u) = K(-u)$

The kernel function given in 10 is univariate and when we have more than one predictive index we need to transform this into the multivariate form. The multivariate kernel function of L dimension, $K_L(\cdot)$, is as follows:

$$K_L(\mathbf{u}) = K_L(u_1,\ \ldots,\ u_L) = K\left(\left(\sum_{l=1}^{L} u_l^2\right)^{\frac{1}{2}}\right)$$

The smoothing parameter h_l indicates the window size of the kernel defining its bandwidth. The amount of observations used for local fitting gets higher as h_l increases. Hence, with a large h_l you have a smoother regression curve and with a small h your regression curve fluctuates more. $h_l = 0$ results in an estimate which essentially interpolates the observations, while $h_l = +\infty$ is equivalent to a linear model (Fan and Gijbels, 1996). A popular rule-of-thumb procedure is to choose

$$h_l = c_l S(p_l)(T-1)^{-1/(4+L)} \qquad \text{for } l = 1,\ \ldots,\ L, \quad (11)$$

where c_l is an arbitrary positive constant, $S(p_l)$ is the sample standard deviation of $\{p_{lt}\}_{t=1,\ldots,T-1}$ and $T-1$ is the number of observations (Li and Racine, 2007). In practice, c_l is often chosen to be 1 or some other constant close to 1. An alternative selection method for bandwidth is given by Hurvich et al. (1998). This method is based on the AICc information criterion. We test with many different bandwidths and select the one with the best AICc. More details about this information criterion will follow in section 3.1.

Smoothing matrix $\mathbf{H}$ is a $(T-1) \times (T-1)$ matrix of kernel weights in which $(i,\ j)$ element is given by $H_{ij} = K_{h,ij} / \sum_{t=1}^{T-1} K_{h,it}$ where $K_{h,ij} = \prod_{l=1}^{L} h_l^{-1} K((p_{li} - p_{lj})/h_l)$ (Li and Racine, 2007). This matrix is needed to find the aforementioned AICc by Hurvich et al. (1998).

2.6. Selection of parameters

Selection of parameters is important in practice, since sometimes the results of the models differ considerably depending on it. In finding the conditional covariance matrix given in 7 and 8, the number of slices H has to be determined. The choice of H may affect the asymptotic variance of the estimators, but the difference is not significant for practical sample sizes (Li, 1991). Fan et al. (2017) show that we always have the same rate of convergence for $\widehat{\boldsymbol{\phi}}_l$ as long as $H \leq \max\{L, 2\}$.

In terms of the number of predictive indices L, the first L eigenvalues of $\Sigma_{f|y}$ must be significantly different from zero, compared to the estimation error. If the factors are normally distributed, the asymptotic distribution of the average of the smallest $K - L$ eigenvalues of $\Sigma_{f|y}$ is chi-square distribution (Li, 1991). However, the normality of factors is hardly satisfied, so if the normality cannot be confirmed by a statistical test such as the Jarque-Bera test, one can just choose L such that the ratio of the L^{th} eigenvalue to the sum of all eigenvalues is greater than 0.05 and $L + 1^{th}$ is not.

We should choose the number of factors K as small as possible, but so that the included factors explain at least a sufficiently high fraction of the total variance of the predictors. The ratio-based estimator given by Ahn and Horenstein (2013) can be a method, in which we maximize a ratio of two adjacent eigenvalues of $\mathbf{X}'\mathbf{X}$ arranged in descending order, i.e.

$$\widehat{K} = \underset{1 \leq i \leq kmax}{\text{argmax}} \quad \hat{\lambda}_i / \hat{\lambda}_{i+1},$$

where $\hat{\lambda}_1 \geq \ldots \geq \hat{\lambda}_{T-1}$ are the eigenvalues.

The learning rate of the ANN works as a controller of the speed of training. The ANN is trained by gradient descent on the weights and thresholds as mentioned in section 2.4. This means that at each iteration we use back-propagation to calculate the derivative of the loss function with respect to each weight and threshold and subtract it from them. However, without learning rate, the weights will vary far too much each iteration, which will make them over-correct and the loss will actually increase. Therefore, in practice, people usually multiply each derivative by a small value of learning rate before they update its corresponding weight. The choice of the learning rate is important, since a high value can cause too much change, leading to the minimum to be missed, while an overly low learning

rate slows down the training unnecessarily. This value must be between 0 and 1 and should be chosen differently depending on the activation function of the output neuron. Typically, the learning rate for a neuron with identity activation function is set to be 0.01 and one with sigmoid activation function is 0.1.

Remark. *The ratio-based method of choosing K in this section is introduced by Fan et al. (2017). However, we found that this method is not applicable for some empirical applications including the data used in section 4. Chances are that in the financial data the first eigenvalue is significantly greater than the second one, which leads the ratio of the first to the second eigenvalue to be the greatest among all the ratios. However, with $\widehat{K} = 1$, the conditional covariance in 7 and 8 is a scalar, that would make it impossible to find the SDR directions. Hence, when deciding on parameter K, it is important not only to use a predefined mathematical method but also to consider the individual characteristics of the data.*

2.7. Out-of-sample forecast without look-ahead bias

Testing a model's forecast performance is commonly conducted by splitting a data set into an in-sample period, used for the initial parameter estimation, and an out-of-sample period, used to evaluate forecasting performance. Typically, in the time series we estimate the model by using the observations before the target and use this estimated model to make a forecast. In order to get a precise statistical performance measure, one must not include any observations at or after the time of the target in estimating the parameters. Violating this principle is the so-called '*look-ahead bias*'.

Estimating the conditional covariance matrix in 7 and 8 inevitably includes a one-step-ahead target variable. When predicting the target at τ, you would need to include target y_τ in estimating the covariance matrix in order to make predictive indices at $\tau - 1$, which is definitely the look-ahead bias. Then how can we obtain the predictive indices at time $\tau - 1$ avoiding this problem? One feasible way would be to estimate the conditional covariance matrix with the factors for $t = 1, \ldots, \tau - 2$ and then to make the predictive indices at $\tau - 1$ with the factors at $\tau - 1$ and the SDR directions from the conditional covariance matrix. We can use this predictive indices for forecasting the target at τ. The procedure of this out-of-sample forecast can be summarized in the following algorithm 2.

In this algorithm, we should see that the estimation of $\Sigma_{f|y}$ includes only the factors until $\tau - 2$ and the targets from $t = 2$ to $\tau - 1$. The reason is that if we include the factors at $\tau - 1$, we also have to include the forecasting target at τ. The predictive indices at $\tau - 1$ are made out of the SDR directions $\widehat{\boldsymbol{\phi}}_1, \ldots, \widehat{\boldsymbol{\phi}}_L$ that are estimated without the factors at $\tau - 1$ and the target at τ.

Algorithm 2: Out-of-sample forecast

Step 1: Obtain estimated factors $\{\widehat{\mathbf{f}}_t\}_{t=1,\ldots,\tau-1}$ from $\{\mathbf{x}_t\}_{t=1,\ldots,\tau-1}$

Step 2: Estimate $\Sigma_{f|y}$ from 7 and 8 by using $\{\widehat{\mathbf{f}}_t\}_{t=1,\ldots,\tau-2}$ and $\{y_t\}_{t=2,\ldots,\tau-1}$

Step 3: Obtain the estimated SDR directions, $\widehat{\boldsymbol{\phi}}_1, \ldots, \widehat{\boldsymbol{\phi}}_L$, from $\Sigma_{f|y}$ in Step 2

Step 4: Construct the predictive indices $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t, \ldots, \widehat{\boldsymbol{\phi}}_L'\widehat{\mathbf{f}}_t$ for $t = 1, \ldots, \tau - 1$

Step 5: Use the ANN or LLR to estimate $g(\cdot)$ with $\{\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t, \ldots, \widehat{\boldsymbol{\phi}}_L'\widehat{\mathbf{f}}_t\}_{t=1,\ldots,\tau-2}$ and $\{y_t\}_{t=2,\ldots,\tau-1}$

Step 6: Make a forecast $\hat{y}_\tau$ with $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_{\tau-1}, \ldots, \widehat{\boldsymbol{\phi}}_L'\widehat{\mathbf{f}}_{\tau-1}$

3. Simulation studies

In this section, we generate data by the conventional time series models and the factor models. We will see both in-sample and out-of-sample performances which will be examined by various evaluation criteria. In this simulation, we will extract factors from predictors and then generate predictive indices out of the factors as explained in section 2.2. Then we will use three ANNs and local linear regression with five different bandwidths in order to estimate the link function $g(\cdot)$. ANN$(1, m)$ means the ANN with one hidden layer and m hidden neurons. The local linear regression is denoted as LLR(h_f) where h_f is a fraction of the bandwidth to the maximum distance between two opposite corners of the hypercube $(\mathbf{p}_1, \ldots, \mathbf{p}_{T-1})$, i.e. the Euclidean distance between the maximum and the minimum value of the matrix. The ordinary least squares (OLS) model is also used mainly for comparison with the ANNs and LLRs.

Note that we have $T - 1$ pairs of $\mathbf{x}_t$ and y_{t+1} when T observations are generated as $\mathbf{x}_t$ is one-lagged compared to y_{t+1}. Hence, $\{(\mathbf{x}_t,\ y_{t+1})\}_{t=1,\ldots T-1}$ is used in practice and the total number of observations is $T - 1$.

3.1. Model evaluation criteria

The in-sample goodness of fit can be evaluated by the conventional R^2 which describes how close the data are to the fitted model:

$$R^2 = 1 - \frac{\sum_{t=2}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=2}^{T}(y_t - \bar{y})^2}.$$

However, this measure can distort the real goodness of fit of the model since it almost monotonically increases, and never decreases as the number of included parameters increases. So, when fitting models, it is possible to increase the statistical measure by adding parameters, or more variables, but doing so may result in over-fitting. The Akaike Information Criterion (AIC) (Akaike, 1974) is one of the most popular evaluation methods for both linear and nonlinear models. The AIC is defined as the log-likelihood term penalized by the number of model parameters. A very common form of it is:

$$\mathrm{AIC} = \log(\hat{\sigma}^2_{\mathrm{MLE}}) + \frac{2m}{T-1}, \tag{12}$$

where m is degrees of freedom, and $\hat{\sigma}^2_{\text{MLE}}$ denotes the maximum likelihood estimate of the variance of the residual term,

$$\hat{\sigma}^2_{\text{MLE}} = \frac{\sum_{t=2}^{T}(y_t - \hat{y}_t)^2}{T-1}.$$

The degrees of freedom is a parametric concept but it is possible to approximate it in nonparametric regressions, resulting in numbers that are not necessarily integers. For the LLR, m can be approximated by tr($\mathbf{H}$), where $\mathbf{H}$ is the smoothing matrix (Hurvich et al., 1998). In the ANNs, we will consider the number of parameters as m, which is the way adopted by Qi and Zhang (2001). But it should be noted that the number of parameters cannot be an unbiased and consistent estimator for degrees of freedom for the ANN. Hence, all information criteria for the ANNs in this paper are intended for comparing in-sample performance between the ANN models, but not with another form of model such as LLR and OLS. See a more detailed explanation in the remark at the end of this section. For the OLS, the number of variables can be an estimator for the degrees of freedom, m, and this is consistent with the LLR's degrees of freedom estimation, which makes it possible to compare the information criteria of the OLS with those of the LLR. The first term of 12 measures the general fit of a given model and the second term penalizes over-parametrization. A lower AIC means a model is considered to be closer to the truth.

Although the AIC is a reasonable criterion which balances model fitting and model parsimony, the AIC often leads to a model with an unnecessarily large number of parameters in certain cases (Qi and Zhang, 2001). Especially, this over-parametrization problem of the AIC becomes more serious for nonlinear models (De Gooijer and Kumar, 1992). There are many modifications of AIC to fix this problem by adjusting the penalty term, of which AICc is one of the most popular. The application and validity of AICc was extensively discussed by Anderson et al. (1998) and Hurvich et al. (1998). AICc is constructed as follows:

$$\text{AICc} = \log(\hat{\sigma}^2_{\text{MLE}}) + 1 + \frac{2(m+1)}{T-m-3}.$$

As you can see, AICc gives more penalty on the number of parameters compared to AIC.

We also look at the Bayesian Information Criteria (BIC) (Schwarz, 1978). Like the AIC, the BIC attempts to resolve the over-fitting problem by introducing a penalty term for the number of parameters. The penalty term is larger in BIC than in AIC.

$$\text{BIC} = \log(\hat{\sigma}^2_{\text{MLE}}) + \frac{m\log(T-1)}{T-1}$$

We also adopted multiple out-of-sample performance evaluation criteria. The out-of-sample R^2_{OS} suggested by Campbell and Thompson (2008) measures how accurate the predictions by a model are compared to historical averages. This is defined as:

$$R^2_{OS} = 1 - \frac{\sum_{t=[T/2]}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=[T/2]}^{T}(y_t - \bar{y}_t)^2}, \tag{13}$$

where $\hat{y}_t$ is the value predicted by the model such as ANN, LLR and OLS and $\bar{y}_t$ is the historical average, both of which are computed using all information available by $t-1$.

The mean squared error (MSE) and mean absolute error (MAE) respectively measure the average of the squares and the absolute values of the errors or deviations, i.e. the difference between the real y_t and what is estimated, $\hat{y}_t$.

As our out-of-sample starts at $t = [T/2]$, the total number of evaluations is $T - [T/2] + 1$

$$\text{RMSE} = \sqrt{\frac{1}{T-[T/2]+1}\sum_{t=[T/2]}^{T}(y_t - \hat{y}_t)^2}$$

$$\text{MAE} = \frac{1}{T-[T/2]+1}\sum_{t=[T/2]}^{T}|(y_t - \hat{y}_t)|$$

The mean absolute percent error (MAPE) measures the size of the error in the ratio to y_t. It is calculated as the average of the unsigned percentage error, as shown below:

$$\text{MAPE} = \frac{1}{T-[T/2]+1}\sum_{t=[T/2]}^{T}\left|\frac{(y_t - \hat{y}_t)}{y_t}\right|$$

Remark. *In recent years, there have been many papers on the concept of degrees of freedom for general statistical estimators. For linear estimators, such as the nonparametric regression estimators we considered in this paper, tr($\boldsymbol{H}$) is the right choice, and is consistent with the literature on degrees of freedom, such as Efron (1986, 2004). However, there has been few research on any simple approximation for degrees of freedom in the ANN estimator, which is highly nonlinear. One would most likely need to work with the general definition of degrees of freedom as given by Efron, but unfortunately this is going to involve unknown quantities. An alternative approach might be possible using the AICI method of Hurvich et al. (1990), but a likelihood function corresponding to the ANN would be required. Developing degrees of freedom in the ANN would go beyond the scope of this paper and is left for future research.*

3.2. Linear case

We first consider a case where the target variable y_t is generated by a linear function of the latent factors. Here, we assume y_t depends on a single predictive index which is a linear combination of the factors. The data-generating process is as follows:

$$y_{t+1} = \boldsymbol{\phi}'\mathbf{f}_t + \sigma_y\epsilon_{t+1}, \tag{14}$$

$$x_{it} = \mathbf{b}_i'\mathbf{f}_t + u_{it}, \tag{15}$$

where $K = 5$ and $\boldsymbol{\phi} = (0.8, 0.5, 0.3, 0, 0)'$. The loadings $\mathbf{b}_i$'s are drawn from the standard normal distribution. In order to include autocorrelation in the series, we generate f_{jt} and u_{it} with AR(1) processes:

$$f_{jt} = \alpha_j f_{jt-1} + e_{jt}, \qquad u_{it} = \rho_i u_{it-1} + v_{it},$$

where α_j and ρ_i are drawn from $U[0.2, 0.8]$. In 14, σ_y is the standard deviation of $\boldsymbol{\phi}'\mathbf{f}_t$, so that the infeasible best forecast

using $\boldsymbol{\phi}'\mathbf{f}_t$ has an R^2 of approximately 0.5. The unconditional variance of f_{jt} can be found so that

$$\text{var}(f_{jt}) = \text{var}(\alpha_j f_{jt-1} + e_{jt}) = \alpha_j^2 \text{var}(f_{jt-1}) + 1$$
$$\rightarrow \text{var}(f_{jt}) = \frac{1}{1-\alpha_j^2}.$$

Then, the variance of $\boldsymbol{\phi}'\mathbf{f}_t$ is $\Sigma_{j=1}^{K}\phi_j^2/(1-\alpha_j^2)$. There is no cross terms when computing the variance of the linear combination of the factors, because all factors are assumed to be independent of each other. So, we have

$$\sigma_y = \sqrt{\sum_{j=1}^{K} \frac{\phi_j^2}{1-\alpha_j^2}}$$

We used three ANN models, local linear regressions with five different bandwidths and the ordinary least squares. Here we use a single estimated predictive index $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t$. The ANNs have only one hidden layer for the reasons mentioned in section 2.3 and the number of neurons is set to be two, three, and four. We train the networks until there is no improvement in the error function $\sum_{t=2}^{T}(y_t - \hat{y}_t)^2/(T-1)$, 50 times in a row.

We set the fraction bandwidth h_f of the local linear regressions to be 0.5, 0.4, 0.3, 0.2 and 0.1. In order to find the best forecasting local linear regression model, it is important to test with many different bandwidths. With a narrow bandwidth, it is possible to sophisticatedly fit a model in sample, but we are in danger to fall into the over-fitting problem which will lead to a poor out-of-sample forecasting performance. We expect the OLS to perform superior to other nonlinear models when the data is generated by the linear link function, but inferior to the others when nonlinearity exists.

In this simulation, the number of slices H, used for SIR, is set to 10. For the ANNs, the learning rate α is 0.01 for the output neuron and 0.1 for the hidden neurons. The Gaussian kernel function is used throughout the simulation for the LLR(h_f)s. We use only one predictive index for model estimation.

The result of the simulation is given in table C.2. The OLS shows very good performance for both in-sample and out-of-sample. In some cases, it even has a greater in-sample R^2 than the ANNs which have more parameters and LLRs that can be locally fitted. Regardless of the values of p and T, the OLS is also the best model according to AIC, AICc and BIC. Also, in the out-of-sample, the OLS performs quite well, although ANNs and LLRs have a better R^2_{OS} in some cases. This result is not surprising because the data is generated by a linear link function whose features can be perfectly captured by the linear model.

Interestingly, the ANNs have worse in-sample R^2s and out-of-sample R^2_{OS}s than the LLR(0.5), LLR(0.4), LLR(0.3) and LLR(0.2). This result implies that in the presence of linearity the artificial neural network is not a good option.

LLR(0.1) has the best in-sample fitting indicator R^2 across all p and T because it has a relatively narrow bandwidth that allows a dense fitting. But it seems that it has an over-fitting problem when $T = 100$ since its R^2_{OS} is very low compared to the others (and MSE, MAE, MAPE are accordingly high). When $T = 200, 500$, it performs better than or equal to the ANNs. This clearly shows that the nonparametric estimation needs more observations to adequately perform when the model has a narrow bandwidth fraction h_f.

3.3. Nonlinear case with factor interaction

Now we look into the case where the interaction between factors exists. The target y_t is generated as follows:

$$y_{t+1} = f_{1t}(f_{2t} + f_{3t} + 1) + \epsilon_{t+1} \tag{16}$$

where ϵ_{t+1} is drawn from the standard normal distribution. Here we can see interaction between f_{1t} and f_{2t}, f_{3t}. The data-generating process for x_{it} is the same as before, except that now we let $K = 7$. In the factor model estimation, seven factors are extracted from the predictors. In the model estimation, we first include two predictive indices. The result of the simulation is given in table C.3.

In this simulation, the nonlinear models perform way better than the linear model. Across p and T, all ANNs and LLRs have higher R^2 and R^2_{OS} than the OLS. This shows that when nonlinearity such as the factor interaction in 16 exists, a linear model cannot show a satisfying performance.

When $T = 100$, among the ANNs the ANN with one hidden layer and two hidden neurons can be considered the best in-sample by the information criterion AIC. This is because it shows a rather similar or even greater in-sample R^2 with keeping the number of parameters small. With AICc and BIC, the difference becomes even larger since those information criteria put more penalty on having more parameters. Among LLRs, for all p and T, the one with $h_f = 0.1$ is considered the best by AIC and AICc. However, by BIC the LLR with $h_f = 0.2$ is a better model when $T = 100$ while the LLR with $h_f = 0.1$ is still the best when $T = 200, 500$. This is because of the fact that the BIC puts more penalty on having more parameters which can be compensated by having more observations. Surprisingly, among the LLRs with the different bandwidths, the one having the best BIC has also the best R^2_{OS}, which shows BIC could be a model section criterion for out-of-sample forecasting.

The 'highly' nonlinear models such as ANN with four hidden neurons and the LLR with $h_f = 0.1$ improve a lot in out-of-sample when T becomes large. For example, when $p = 100$ and $T = 500$, the LLR with $h_f = 0.1$ and ANN with four hidden neurons have R^2_{OS}s of 0.665 and 0.622, respectively, which are the first and second highest while when $p = 50$ and $T = 100$ they have the worst R^2_{OS} among ANNs and LLRs, respectively. Especially, the LLR with $h_f = 0.1$ seems to have an over-fitting problem when $p = 50$ and $T = 100$ since its out-of-sample performance is far behind its in-sample one. However, its in-sample and out-of-sample performances are superior to all other models when $T = 500$. This shows again that when we use more complicated models, we must be able to rely on a large amount of data to expect a proper performance.

All models, including the OLS, have positive R^2_{OS}, which means they have a better predictive power than the average of the past target's observations. This forecasting performance is

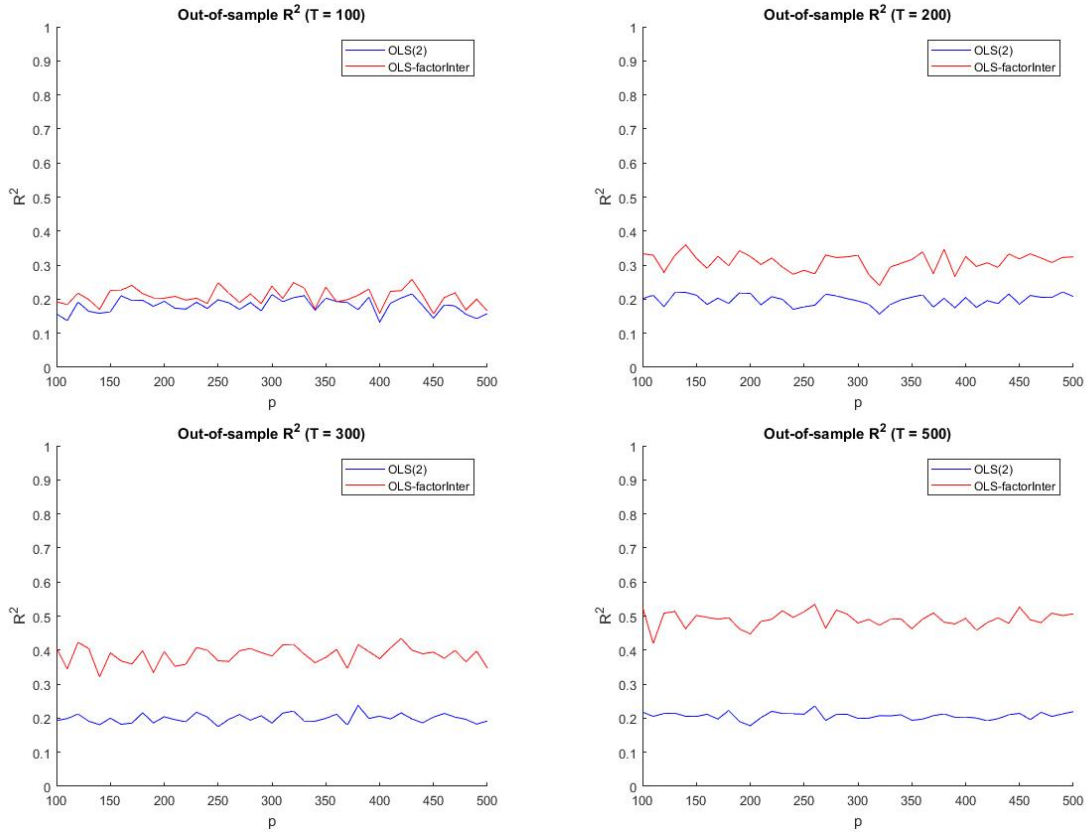

Figure 3: Out-of-sample R^2_{OS} over 100 replications. OLS(2) denotes the OLS including first two predictive indices $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t$ and $\widehat{\boldsymbol{\phi}}_2'\widehat{\mathbf{f}}_t$. OLS-factInter is the OLS with three regressors, $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t$, $\widehat{\boldsymbol{\phi}}_2'\widehat{\mathbf{f}}_t$ and the interaction term, $(\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t)\cdot(\widehat{\boldsymbol{\phi}}_2'\widehat{\mathbf{f}}_t)$.

not surprising if we consider the fact that the dimension of the predictors is reduced in a way that they somehow reflect their target y_{t+1} by the sliced inverse regression. Although having positive R^2_{OS}, the OLS does not seem a good option for nonlinear time series forecasting.

Secondly, we examine if the OLS can be improved by taking the interaction effect into account. Two OLS models are generated, one is the OLS with two predictive indices and the other includes not only the two predictive indices but also the product of these two terms. The modified OLS is constructed as follows:

$$y_{t+1} = \beta_0 + \beta_1\, \widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t + \beta_2\, \widehat{\boldsymbol{\phi}}_2'\widehat{\mathbf{f}}_t + \beta_3\, (\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t)\cdot(\widehat{\boldsymbol{\phi}}_2'\widehat{\mathbf{f}}_t) + \eta_{t+1},$$

where η_{t+1} is set to be standard normal. In figure 3, you can see that across different values of p the modified OLS performs better than the previous one in out-of-sample forecasting. When $T = 100$ both models have R^2_{OS} approximately between 0.15 and 0.25, while the OLS with the interaction term keeps performing slightly better. The difference between these two models becomes larger as T increases. When $T = 500$, the modified OLS's R^2_{OS}s are around 0.5 but the previous models' are around 0.2. OLS models can also perform well if they consider the nonlinearity.

4. Empirical application to financial data

Besides simulation, we will see how well the newly developed forecasting method performs. Here we adopt the data set from Stock and Watson (2012). The data set includes 185 quarterly observed macroeconomic variables from 1959 I to 2008 IV. The fact that it has a large number of variables, almost as many as the observations, makes it appropriate for this paper, which aims at forecasting with dimension reduction. This data set has been adopted by a few papers such as Fan et al. (2017) and a shortened version is used by Ludvigson and Ng (2009) and Bai and Ng (2008).

The time series are made stationary by taking logarithms and/or being differenced. If a variable includes negative values, it is only differenced, otherwise the logarithmed variable is differenced. Following the way in Fan et al. (2017), we treat each of them as a forecast target y_t, with all others forming the predictor set $\mathbf{x}_t$. The forecasting performance is examined by the out-of-sample R^2_{OS} proposed in 13. The learning rate of ANN α is set to be 0.001 for the output neuron and 0.01 for the hidden neurons as before. Following the second school of thought described in Kaastra and Boyd (1996), the ANNs are trained for 200 iterations and we select the weights and thresholds that have the lowest value of the error function. As before, we try with LLRs with five different bandwidth fractions. We extract seven factors from the predictors, which are found to explain

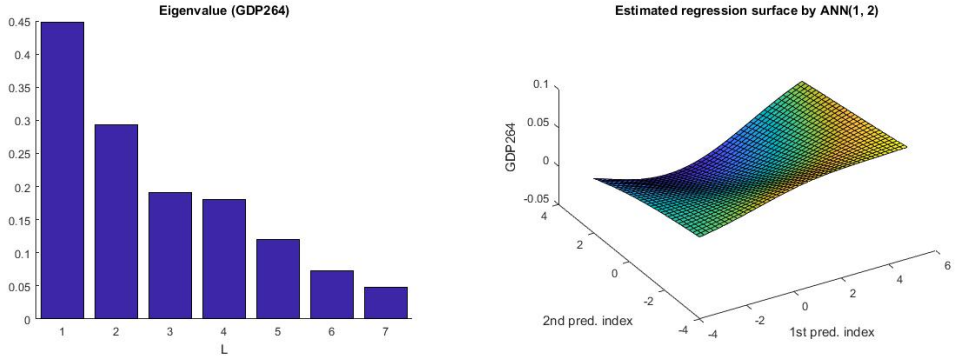

Figure 4: Forecasting results for GDP264 (index for imports). The left panel shows the eigenvalues of $\widehat{\Sigma}_{fy}$ and the right panel gives a 3D plot of the regression surface estimated by ANN(1, 2). The eigenvalue bar diagrams for the rest variables are given in figure D.5 in the appendix.

Table 1: Out-of-sample macroeconomic forecasting.

		ANN(1, 2)	ANN(1, 3)	ANN(1, 4)	LLR(0.5)	LLR(0.4)	LLR(0.3)	LLR(0.2)	LLR(0.1)	OLS
GDP components	GDP264	0.129	0.105	0.115	0.144*	0.144	0.141	0.127	0.053	0.143
IP	IPS13	0.062	0.064	0.098*	0.072	0.072	0.071	0.062	0.014	0.068
Employment	CES048	0.305	0.33	0.387*	0.302	0.302	0.304	0.31	0.331	0.301
Unemployment rate	LHU680	0.151	0.154	0.164*	0.161	0.16	0.16	0.16	0.139	0.161
Housing	HSSOU	-0.012	-0.02	-0.005	0.008	0.012	0.022	0.044*	0.035	0
Inventories	PMNO	-0.104*	-0.123	-0.118	-0.111	-0.11	-0.113	-0.136	-0.209	-0.12
Prices	GDP275_3	-0.021	-0.025	-0.02	-0.012*	-0.013	-0.014	-0.022	-0.081	-0.012
Wages	LBMNU	0.372	0.354	0.354	0.412	0.414	0.418	0.428	0.432*	0.408
Interest rates	FYFF	0.206	0.199	0.208*	0.174	0.175	0.178	0.186	0.205	0.172
Money	CCINRV	0.086	0.153	0.156*	0.042	0.042	0.043	0.051	0.077	0.042
Exchange rates	EXRCAN	0.023	0.02	0.003	0.03	0.031	0.034	0.042	0.065*	0.027
Stock prices	FSDJ	-0.046	-0.047	-0.049	-0.046	-0.047	-0.046	-0.037	-0.022*	-0.044
Consumer expectations	HHSNTN	-0.121	-0.119	-0.125	-0.111	-0.11*	-0.113	-0.136	-0.209	-0.12

Note: Out-of-sample R^2_{OS} for one-quarter ahead forecasts. All models include the first two predictive indices $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t$, $\widehat{\boldsymbol{\phi}}_2'\widehat{\mathbf{f}}_t$. The greatest numbers in each row are asterisked.

more than 40 percent of the variation in the data according to Bai and Ng (2013). The same approach is adopted by Fan et al. (2017). We include two predictive indices here.

One can see the result of the application in table 1. ANN(1, 4) has the greatest R^2_{OS}s in forecasting IP, employment, unemployment rate, interest rates and CCINRV. The LLR with $h_f = 0.1$ is the best-performing model in forecasting wages, exchange rates and stock prices. Since they have more parameters or a smaller bandwidth, they are more flexible than the others. The two models have the best R^2_{OS}s in predicting eight variables out of thirteen. Thus, we can see that adopting models that reflect nonlinearity is recommended in financial forecasting. In certain variables, the OLS does show a performance similar to these highly nonlinear models, e.g GDP components and prices. However, except in stock prices, the R^2_{OS}s of the OLS are less than or equal to the R^2_{OS}s of LLR(0.5), which has the largest bandwidth.

5. Conclusion

In this paper, we have applied the factor model and the sliced inverse regression to reduce the high-dimensional predictors, and then predicted the target by nonlinear models such as the ANN and LLR. Especially we have introduced the procedure for building and training the neural networks when using the factor model and SIR. The performance of the ANNs is compatible with the LLRs when nonlinearity exits between the extracted factors and the forecasting target, while the OLS is good enough when the relation is simply linear. Our application would be helpful for those who develop the real world forecasting model and algorithm. The reduction of dimension is very useful when using nonlinear models since chances are that the curse of dimensionality and over-fitting problems happen with high-dimensional data.

In the real-world forecasting, one would need to adopt the cross-fold validation. The basic idea is that in the in-sample data, we check the potential for generalization of the model in forecasting. For the ANN, the data is split into a training, a test and a validation set. The training set is used for estimating weights and thresholds, and this trained ANN is evaluated with the validation set. The final choice of the ANN is the one which performs best in the test set. For the detailed procedure of the ANN cross-validation customized for time series forecasting, see Kaastra and Boyd (1996). For the LLR, AICc (Hurvich et al., 1998) in section 3.1 can be applied as a selection criterion of the bandwidth. Li and Racine (2004) show that AICc tends to perform better than other cross-validation methods in a limited sample. Although we tested multiple ANNs with different numbers of neurons and LLRs with different bandwidths,

the complete cross-validation has not been done. This is left for future research.

Acknowledgments

This paper has been written as a graduation thesis for the international bachelor program *Econometrics and Operations research* at the Erasmus School of Economics. The author wants to especially thank Dr. Maria Girth (ESE), who supervised this research project with professionalism and kindness. Further thanks go to Prof. Clifford Hurvich (Stern, NYU), who gave detailed advice on using the AICc in section 3.1.

AppendixA. Proof of theorem 1

$$
\begin{aligned}
\widehat{\Sigma}^2_{f|y} &= \widehat{\mathbf{\Lambda}}_b\Big(\frac{1}{H}\sum_{h=1}^{H}\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{x}}_{(h,l)}\Big]\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{x}}_{(h,l)}\Big]'\Big)\widehat{\mathbf{\Lambda}}_b' \\
&= \widehat{\mathbf{\Lambda}}_b\Big(\frac{1}{H}\sum_{h=1}^{H}\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{B}}_{(h,l)}\widehat{\mathbf{f}}_{(h,l)}+\widehat{u}_{(h,l)}\Big]\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{B}}_{(h,l)}\widehat{\mathbf{f}}_{(h,l)}+\widehat{u}_{(h,l)}\Big]'\Big)\widehat{\mathbf{\Lambda}}_b' \\
&= \frac{1}{H}\sum_{h=1}^{H}\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}+\widehat{\mathbf{\Lambda}}_b\widehat{u}_{(h,l)}\Big]\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}+\widehat{\mathbf{\Lambda}}_b\widehat{u}_{(h,l)}\Big]' \\
&= \frac{1}{H}\sum_{h=1}^{H}\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}\Big]\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}\Big]' + \frac{1}{H}\sum_{h=1}^{H}\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}\Big(\frac{1}{c}\sum_{l=1}^{c}\widehat{u}'_{(h,l)}\Big)\widehat{\mathbf{\Lambda}}_b' \\
&\quad + \widehat{\mathbf{\Lambda}}_b\Big(\frac{1}{c}\sum_{l=1}^{c}\widehat{u}_{(h,l)}\Big)\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}'_{(h,l)} + \widehat{\mathbf{\Lambda}}_b\Big(\frac{1}{c}\sum_{l=1}^{c}\widehat{u}_{(h,l)}\Big)\Big(\frac{1}{c}\sum_{l=1}^{c}\widehat{u}'_{(h,l)}\Big)\widehat{\mathbf{\Lambda}}_b'\Big] \\
&= \widehat{\Sigma}^1_{f|y} + \widehat{D}(H,c),
\end{aligned}
\tag{A.1}
$$

where $\widehat{D}(H,c) = \widehat{\Sigma}^2_{f|y} - \frac{1}{H}\sum_{h=1}^{H}\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}\Big]\Big[\frac{1}{c}\sum_{l=1}^{c}\widehat{\mathbf{f}}_{(h,l)}\Big]' = \widehat{\Sigma}^2_{f|y} - \widehat{\Sigma}^1_{f|y}$. By the law of large numbers, as $c \to \infty$, $\frac{1}{c}\sum_{l=1}^{c}\widehat{u}_{(h,l)}$ converges to zero, which leads $\widehat{D}(H,c) = 0$. So, when $c \to \infty$, only $\widehat{\Sigma}^1_{f|y}$ remains in A.1. Therefore, $\widehat{\Sigma}^1_{f|y} = \widehat{\Sigma}^2_{f|y}$ as $c \to \infty$.

AppendixB. Proof of the remark of theorem 1

This proof is given by Fan et al. (2017). It is sufficient to show that $\widehat{\mathbf{F}}'_t = \widehat{\mathbf{\Lambda}}_b\mathbf{X}$. By construction of the factor model estimation 3, we have $\mathbf{X}'\mathbf{X}\widehat{\mathbf{F}} = \widehat{\mathbf{F}}$ where $\mathbf{M} = \text{diag}(\lambda_1, \ldots, \lambda_k)$ and λ_k are the largest K eigenvalues of $\mathbf{X}'\mathbf{X}$. Then,

$$
\begin{aligned}
\widehat{\mathbf{B}}'\widehat{\mathbf{B}} &= (T^{-1}\mathbf{X}\widehat{\mathbf{F}})'(T^{-1}\mathbf{X}\widehat{\mathbf{F}}) = T^{-2}\widehat{\mathbf{F}}'(\mathbf{X}'\mathbf{X})\widehat{\mathbf{F}} = \\
&= T^{-2}\widehat{\mathbf{F}}'\widehat{\mathbf{F}}\mathbf{M} = T^{-1}\mathbf{M}
\end{aligned}
$$

Now, you can see that

$$
(\widehat{\mathbf{B}}'\widehat{\mathbf{B}})^{-1}\widehat{\mathbf{B}}'\mathbf{X} = T\mathbf{M}^{-1}(T^{-1}\widehat{\mathbf{F}}'\mathbf{X}')\mathbf{X} = \mathbf{M}^{-1}(\mathbf{X}'\mathbf{X}\mathbf{F})' = \widehat{\mathbf{F}}'
$$

Hence, $\widehat{\mathbf{F}}'_t = \widehat{\mathbf{\Lambda}}_b\mathbf{X}$ under model 3 and this leads $\widehat{\Sigma}^1_{f|y} = \widehat{\Sigma}^2_{f|y}$.

AppendixC. Results of simulations

Table C.2: In-sample and out-of-sample performance of models in the linear case.

	ANN(1, 2)	ANN(1, 3)	ANN(1, 4)	LLR(0.5)	LLR(0.4)	LLR(0.3)	LLR(0.2)	LLR(0.1)	OLS
$p = 50$ $T = 100$									
R^2	0.484	0.483	0.481	0.489	0.491	0.494	0.499	0.512	0.486
AIC	-0.546	-0.484	-0.418	-0.65	-0.65	-0.649	-0.645	-0.629	-0.67
AICc	0.49	0.567	0.653	0.373	0.373	0.375	0.38	0.403	0.351
BIC	-0.362	-0.221	-0.077	-0.59	-0.586	-0.577	-0.553	-0.478	-0.644
R^2_{OS}	0.449	0.451	0.45	0.473	0.472	0.47	0.46	0.386	0.472
RMSE	0.748	0.747	0.747	0.731	0.732	0.733	0.74	0.785	0.731
MAE	0.601	0.601	0.601	0.59	0.59	0.59	0.595	0.614	0.59
MAPE	3.43	3.646	3.378	3.253	3.252	3.25	3.25	3.304	3.253
$p = 50$ $T = 200$									
R^2	0.475	0.472	0.468	0.492	0.493	0.494	0.496	0.504	0.491
AIC	-0.585	-0.549	-0.512	-0.665	-0.665	-0.664	-0.661	-0.654	-0.676
AICc	0.428	0.468	0.509	0.346	0.346	0.347	0.35	0.359	0.335
BIC	-0.47	-0.383	-0.297	-0.628	-0.625	-0.619	-0.604	-0.558	-0.659
R^2_{OS}	0.452	0.452	0.457	0.482	0.481	0.48	0.477	0.457	0.482
RMSE	0.742	0.743	0.739	0.722	0.722	0.723	0.725	0.739	0.722
MAE	0.591	0.592	0.589	0.576	0.577	0.577	0.579	0.586	0.576
MAPE	3.427	3.586	3.71	3.161	3.169	3.184	3.216	3.35	3.145
$p = 100$ $T = 100$									
R^2	0.491	0.488	0.484	0.498	0.499	0.501	0.506	0.521	0.495
AIC	-0.558	-0.491	-0.423	-0.666	-0.665	-0.663	-0.659	-0.644	-0.687
AICc	0.478	0.56	0.647	0.357	0.358	0.361	0.367	0.387	0.334
BIC	-0.375	-0.229	-0.082	-0.606	-0.601	-0.591	-0.566	-0.494	-0.661
R^2_{OS}	0.449	0.448	0.449	0.472	0.47	0.468	0.46	0.384	0.473
RMSE	0.739	0.739	0.738	0.723	0.724	0.726	0.731	0.779	0.722
MAE	0.591	0.593	0.59	0.58	0.581	0.582	0.586	0.607	0.58
MAPE	4.305	4.489	4.609	4.118	4.121	4.118	4.079	3.946	4.103
$p = 100$ $T = 500$									
R^2	0.475	0.475	0.471	0.493	0.493	0.494	0.494	0.498	0.492
AIC	-0.621	-0.609	-0.59	-0.675	-0.675	-0.675	-0.674	-0.67	-0.679
AICc	0.384	0.396	0.415	0.329	0.329	0.329	0.33	0.334	0.325
BIC	-0.562	-0.525	-0.481	-0.656	-0.655	-0.653	-0.646	-0.622	-0.671
R^2_{OS}	0.462	0.464	0.46	0.488	0.488	0.488	0.487	0.482	0.488
RMSE	0.735	0.734	0.736	0.717	0.717	0.717	0.718	0.721	0.717
MAE	0.587	0.586	0.588	0.573	0.573	0.573	0.574	0.576	0.573
MAPE	3.961	3.708	3.932	3.659	3.661	3.665	3.672	3.669	3.654
$p = 500$ $T = 100$									
R^2	0.49	0.487	0.487	0.498	0.499	0.501	0.507	0.521	0.495
AIC	-0.555	-0.487	-0.425	-0.664	-0.664	-0.662	-0.657	-0.642	-0.686
AICc	0.481	0.564	0.645	0.359	0.36	0.362	0.368	0.389	0.336
BIC	-0.371	-0.225	-0.085	-0.604	-0.599	-0.589	-0.565	-0.492	-0.659
R^2_{OS}	0.465	0.468	0.465	0.492	0.491	0.487	0.48	0.427	0.494
RMSE	0.736	0.734	0.736	0.717	0.718	0.72	0.725	0.76	0.715
MAE	0.589	0.59	0.59	0.575	0.576	0.578	0.581	0.598	0.574
MAPE	4.295	4.327	4.314	4.245	4.257	4.27	4.245	4.119	4.219
$p = 500$ $T = 500$									
R^2	0.488	0.486	0.486	0.503	0.504	0.504	0.505	0.508	0.503
AIC	-0.646	-0.63	-0.619	-0.696	-0.696	-0.695	-0.694	-0.691	-0.7
AICc	0.358	0.375	0.387	0.308	0.309	0.309	0.31	0.314	0.304
BIC	-0.587	-0.546	-0.509	-0.677	-0.676	-0.673	-0.667	-0.642	-0.691
R^2_{OS}	0.472	0.47	0.47	0.498	0.497	0.497	0.496	0.49	0.498
RMSE	0.724	0.726	0.725	0.707	0.707	0.707	0.708	0.712	0.706
MAE	0.577	0.578	0.578	0.563	0.563	0.563	0.564	0.566	0.563
MAPE	2.966	3.088	3.123	2.839	2.84	2.842	2.848	2.861	2.837

Note: ANN(1, m) denotes the artificial neural network with one hidden layer and m hidden neurons. LLR(h_f) is the local linear regression with bandwidth $h_f \times maxDist$ and OLS is the ordinary least squares. All the models use one single predictive index $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t$. The evaluation criteria are averaged over 100 replications.

Table C.3: In-sample and Out-of-sample evaluation criteria in the nonlinear case with factor interaction.

	ANN(1, 2)	ANN(1, 3)	ANN(1, 4)	LLR(0.5)	LLR(0.4)	LLR(0.3)	LLR(0.2)	LLR(0.1)	OLS
$p = 50$ $T = 100$									
R^2	0.428	0.447	0.444	0.326	0.347	0.383	0.441	0.536	0.278
AIC	-0.458	-0.413	-0.333	-0.361	-0.392	-0.445	-0.528	-0.59	-0.315
AICc	0.588	0.658	0.773	0.664	0.634	0.582	0.505	0.482	0.708
BIC	-0.222	-0.072	0.112	-0.271	-0.295	-0.333	-0.371	-0.244	-0.262
R^2_{OS}	0.281	0.278	0.273	0.23	0.248	0.276	0.303	0.204	0.185
RMSE	0.85	0.851	0.853	0.888	0.876	0.857	0.835	0.883	0.916
MAE	0.629	0.632	0.634	0.657	0.649	0.635	0.618	0.641	0.678
MAPE	4.372	3.382	3.675	3.987	3.81	3.468	3.733	5.046	4.349
$p = 50$ $T = 200$									
R^2	0.467	0.484	0.487	0.31	0.338	0.385	0.461	0.554	0.249
AIC	-0.591	-0.587	-0.556	-0.355	-0.396	-0.471	-0.603	-0.742	-0.28
AICc	0.425	0.434	0.473	0.656	0.615	0.541	0.41	0.281	0.731
BIC	-0.442	-0.372	-0.274	-0.299	-0.336	-0.402	-0.509	-0.519	-0.247
R^2_{OS}	0.391	0.396	0.41	0.269	0.298	0.344	0.41	0.429	0.203
RMSE	0.769	0.765	0.756	0.845	0.828	0.8	0.757	0.741	0.882
MAE	0.566	0.565	0.563	0.619	0.607	0.587	0.558	0.546	0.644
MAPE	2.909	3.124	3.135	3.302	3.208	3.055	2.837	2.875	3.496
$p = 100$ $T = 100$									
R^2	0.423	0.431	0.439	0.319	0.339	0.374	0.431	0.524	0.272
AIC	-0.452	-0.383	-0.325	-0.353	-0.383	-0.434	-0.515	-0.572	-0.308
AICc	0.594	0.688	0.782	0.673	0.643	0.593	0.518	0.501	0.714
BIC	-0.216	-0.042	0.121	-0.262	-0.285	-0.321	-0.358	-0.225	-0.256
R^2_{OS}	0.266	0.273	0.271	0.224	0.245	0.276	0.305	0.189	0.172
RMSE	0.845	0.84	0.839	0.871	0.858	0.839	0.821	0.881	0.9
MAE	0.629	0.626	0.627	0.649	0.64	0.627	0.613	0.64	0.67
MAPE	2.928	2.853	2.999	2.843	2.803	2.745	2.718	3.091	2.941
$p = 100$ $T = 500$									
R^2	0.575	0.636	0.666	0.336	0.377	0.448	0.568	0.698	0.246
AIC	-0.866	-1.005	-1.089	-0.404	-0.468	-0.591	-0.844	-1.204	-0.282
AICc	0.139	0.001	-0.082	0.6	0.536	0.413	0.16	-0.198	0.722
BIC	-0.79	-0.895	-0.946	-0.376	-0.439	-0.558	-0.799	-1.093	-0.265
R^2_{OS}	0.548	0.6	0.622	0.318	0.361	0.436	0.557	0.665	0.221
RMSE	0.669	0.629	0.61	0.828	0.801	0.752	0.665	0.574	0.884
MAE	0.497	0.473	0.463	0.6	0.583	0.552	0.496	0.437	0.637
MAPE	2.885	2.839	3.143	3.483	3.381	3.198	2.899	2.749	3.698
$p = 500$ $T = 100$									
R^2	0.428	0.438	0.442	0.317	0.34	0.377	0.436	0.531	0.267
AIC	-0.449	-0.39	-0.317	-0.347	-0.378	-0.431	-0.514	-0.569	-0.3
AICc	0.597	0.681	0.79	0.679	0.648	0.597	0.519	0.504	0.723
BIC	-0.213	-0.049	0.129	-0.256	-0.28	-0.318	-0.357	-0.222	-0.247
R^2_{OS}	0.274	0.282	0.274	0.239	0.259	0.288	0.316	0.231	0.188
RMSE	0.859	0.853	0.857	0.884	0.872	0.853	0.832	0.877	0.914
MAE	0.626	0.625	0.63	0.645	0.636	0.622	0.607	0.634	0.667
MAPE	7.326	8.639	8.325	6.565	6.788	7.072	7.324	7.665	5.968
$p = 500$ $T = 500$									
R^2	0.578	0.646	0.677	0.335	0.376	0.448	0.572	0.708	0.245
AIC	-0.878	-1.035	-1.123	-0.403	-0.468	-0.592	-0.853	-1.241	-0.28
AICc	0.127	-0.029	-0.116	0.601	0.536	0.412	0.152	-0.235	0.724
BIC	-0.802	-0.925	-0.979	-0.375	-0.438	-0.559	-0.808	-1.13	-0.263
R^2_{OS}	0.546	0.602	0.634	0.319	0.363	0.438	0.561	0.676	0.224
RMSE	0.671	0.627	0.601	0.828	0.801	0.752	0.663	0.564	0.885
MAE	0.494	0.47	0.455	0.596	0.579	0.548	0.491	0.429	0.632
MAPE	3.988	3.679	4.773	3.939	3.819	3.606	3.294	3.402	4.198

Note: All models include the first two predictive indices $\widehat{\boldsymbol{\phi}}_1'\widehat{\mathbf{f}}_t$, $\widehat{\boldsymbol{\phi}}_2'\widehat{\mathbf{f}}_t$. The evaluation criteria are averaged over 100 replications.

AppendixD. Eigenvalues of the empirical data of Stock and Watson (2012)

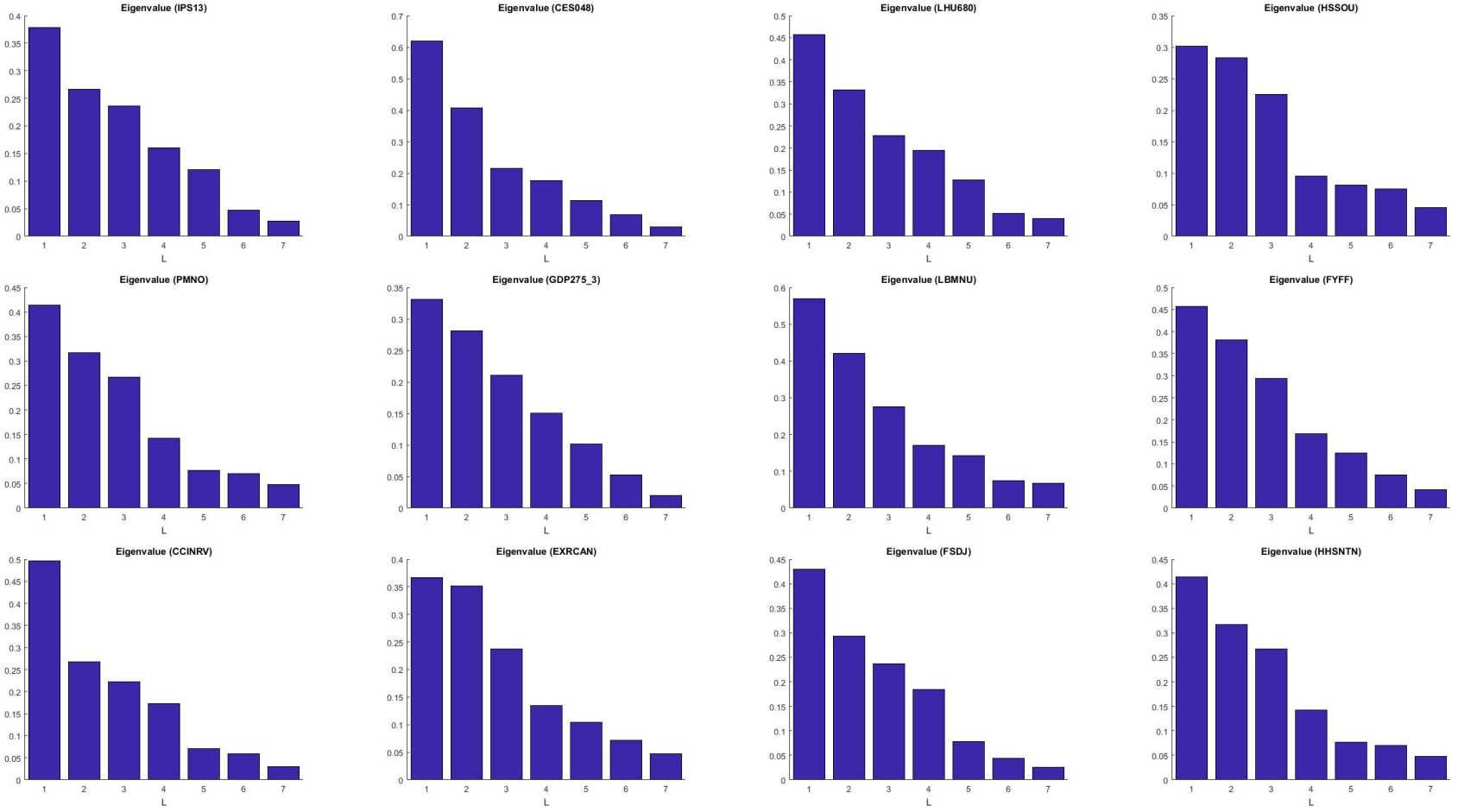

Figure D.5: The eigenvalues of $\widehat{\Sigma}_{fly}$ of each macroeconomic variables.

Ahn, S. C. and Horenstein, A. R. (2013). Eigenvalue ratio test for the number of factors. *Econometrica*, 81(3):1203–1227.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723.

Anderson, D. R., Burnham, K. P., and White, G. C. (1998). Comparison of akaike information criterion and consistent akaike information criterion for model selection and statistical inference from capture-recapture studies. *Journal of Applied Statistics*, 25(2):263–282.

Bai, J. and Ng, S. (2008). Large dimensional factor analysis. *Foundations and Trends in Econometrics*, 3(2):89–163.

Bai, J. and Ng, S. (2013). Principal components estimation and identification of static factors. *Journal of Econometrics*, 176(1):18–29.

Campbell, J. Y. and Thompson, S. B. (2008). Predicting excess stock returns out of sample: Can anything beat the historical average? *The Review of Financial Studies*, 21(4):1509–1531.

De Gooijer, J. G. and Kumar, K. (1992). Some recent developments in non-linear time series modelling, testing, and forecasting. *International Journal of Forecasting*, 8(2):135–156.

Doz, C., Giannone, D., and Reichlin, L. (2011). A two-step estimator for large approximate dynamic factor models based on kalman filtering. *Journal of Econometrics*, 164(1):188–205.

Efron, B. (1986). How biased is the apparent error rate of a prediction rule? *Journal of the American statistical Association*, 81(394):461–470.

Efron, B. (2004). The estimation of prediction error: covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467):619–632.

Fama, E. F. and French, K. R. (2015). A five-factor asset pricing model. *Journal of Financial Economics*, 116(1):1–22.

Fan, J. and Gijbels, I. (1996). *Local polynomial modelling and its applications: monographs on statistics and applied probability*. CRC Press, vol. 66 edition.

Fan, J., Xue, L., and Yao, J. (2017). Sufficient forecasting using factor models. *Journal of Econometrics*, 201(2):292–306.

Forni, M., Hallin, M., Lippi, M., and Reichlin, L. (2000). The generalized dynamic-factor model: Identification and estimation. *Review of Economics and statistics*, 82(4):540–554.

Haykin, S. S. (2009). *Neural networks and learning machines*. Upper Saddle River, NJ, USA:: Pearson, vol. 3 edition.

Hsieh, D. A. (1991). Chaos and nonlinear dynamics: application to financial markets. *The journal of finance*, 46(5):1839–1877.

Hurvich, C. M., Shumway, R., and Tsai, C. L. (1990). Improved estimators of kullback–leibler information for autoregressive model selection in small samples. *Biometrika*, 77(4):709–719.

Hurvich, C. M., Simonoff, J. S., and Tsai, C. L. (1998). Smoothing parameter selection in nonparametric regression using an improved akaike information criterion. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(2):271–293.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307.

Kaastra, I. and Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. *Neurocomputing*, 10(3):215–236.

Kapetanios, G. and Marcellino, M. (2009). A parametric estimation method for dynamic factor models of large dimensions. *Journal of Time Series Analysis*, 30(2):208–238.

Li, K. C. (1991). Sliced inverse regression for dimension reduction. *Journal of the American Statistical Association*, 86(414):316–327.

Li, Q. and Racine, J. (2004). Cross-validated local linear nonparametric regression. *Statistica Sinica*, pages 485–512.

Li, Q. and Racine, J. S. (2007). *Nonparametric econometrics: theory and practice*. Princeton University Press.

Ludvigson, S. C. and Ng, S. (2009). A factor analysis of bond risk premia (no. w15188). *National Bureau of Economic Research*.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems*. Pearson Education.

Qi, M. and Zhang, G. P. (2001). An investigation of model selection criteria for neural network time series forecasting. *European Journal of Operational Research*, 132(3):666–680.

Schumacher, C. and Breitung, J. (2008). Real-time forecasting of german gdp based on a large factor model with monthly and quarterly data. *International Journal of Forecasting*, 24(3):386–398.

Schwarz, G. (1978). Estimating the dimension of a model. the annals of statistics. *The annals of statistics*, 6(2):461–464.

Shamim, M. A., Hassan, M., A. S., and Zeeshan, M. (2016). A comparison of artificial neural networks (ann) and local linear regression (llr) techniques for predicting monthly reservoir levels. *KSCE Journal of Civil Engineering*, 20(2):971–977.

Stock, J. H. and Watson, M. W. (2002). Forecasting using principal components from a large number of predictors. *Journal of the American statistical association*, 97(460):1167–1179.

Stock, J. H. and Watson, M. W. (2012). Generalized shrinkage methods for forecasting using many predictors. *Journal of Business & Economic Statistics*, 30(4):481–493.

Watrous, R. L. (1987). Learning phonetic features using connectionist networks: An experiment in speech recognition. *1987 1st International Conference on Neural Networks*.

Yuan, J. L. and Fine, T. (1993). Forecasting demand for electric power. *In Advances in Neural Information Processing Systems*, pages 739–746.

Yuan, J. L. and Fine, T. L. (1998). Neural-network design for small training sets of high dimension. *IEEE Transactions on neural networks*, 9(2):266–280.

Zhang, G. P. (2003). Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175.

Zhang, G. P. and Berardi, V. L. (2001). Time series forecasting with neural network ensembles: an application for exchange rate prediction. *Journal of the Operational Research Society*, 52(6):652–664.

Zhang, G. P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514.