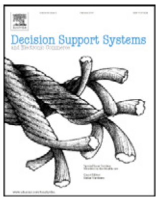
Contents lists available at SciVerse ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss



Forecasting and trading the EUR/USD exchange rate with stochastic Neural Network combination and time-varying leverage

Georgios Sermpinis [a,*], Christian Dunis [b], Jason Laws [c], Charalampos Stasinakis [a]

[a] University of Glasgow Business School, University of Glasgow, Gilbert Scott Building, Glasgow, G12 8QQ, United Kingdom
[b] Liverpool Business School, JMU, John Foster Building, 98 Mount Pleasant, Liverpool L3 5UZ, United Kingdom
[c] University of Liverpool Management School, The University of Liverpool, Chatham Street, Liverpool, L69 7ZH, United Kingdom

ARTICLE INFO

Article history:
Received 18 September 2011
Received in revised form 1 May 2012
Accepted 21 May 2012
Available online 30 May 2012

Keywords:
Psi Sigma network
Recurrent Network
Forecast combinations
Kalman Filter
LASSO
Leverage

ABSTRACT

The motivation of this paper is to investigate the use of a Neural Network (NN) architecture, the Psi Sigma Neural Network (PSN), when applied to the task of forecasting and trading the Euro/Dollar (EUR/USD) exchange rate using the European Central Bank (ECB) fixing series and to explore the utility of Kalman Filters in combining NN forecasts. This is done by benchmarking the statistical and trading performance of PSN with a Naive Strategy, an Autoregressive Moving Average (ARMA) model and two different NN architectures, a Multi-Layer Perceptron (MLP) and a Recurrent Network (RNN). We combine our NN forecasts with Kalman Filter, a traditional Simple Average, the Bayesian Average, the Granger–Ramanathan's Regression Approach (GRR) and the Least Absolute Shrinkage and Selection Operator (LASSO). Finally, we apply a time-varying leverage strategy based on RiskMetrics volatility forecasts in order to further improve the forecasting performance of our models and combinations. The statistical and trading performance of our models is estimated throughout the period of 2002–2010, using the last two years for out-of-sample testing. In terms of our results, the PSN outperforms all models' individual performances in terms of statistical accuracy and trading performance. The forecast combinations also present improved empirical evidence, with Kalman Filters outperforming by far its benchmarks. We also note that after the application of the time varying leverage, all models except ARMA show a substantial increase in their trading performance.

1. Introduction

The term of Neural Network (NN) originates from the biological neuron connections of human brain. The artificial NNs are computation models that embody data-adaptive learning and clustering abilities, deriving from parallel processing procedures [34]. The NNs are considered a relatively new technology in Finance, but with high potential and an increasing number of applications. However, their practical limitations and contradictory empirical evidence lead to skepticism on whether they can outperform existing traditional models.

The motivation for this paper is to investigate the trading performance of a novel Neural Network (NN) architecture, the Psi Sigma Neural Network (PSN), and explore the utility of Kalman Filters in combining NN forecasts. Firstly, we apply the EUR/USD European Central Bank (ECB) fixing series to a Naive Strategy, an Autoregressive Moving Average (ARMA) model and three NNs, namely a Multi-Layer Perceptron (MLP), a Recurrent Network (RNN) and a PSN. Secondly, we compare the Kalman Filter with four forecast combination methods. That is the traditional Simple Average, the Bayesian Average, Granger–Ramanathan's Regression Approach (GRR) and the Least Absolute Shrinkage and Selection Operator (LASSO). The models' performance is estimated using the EUR/USD ECB fixing series of the period of 2002–2010, using the last two years for out-of-sample testing. We also introduce a time-varying leverage strategy based on RiskMetrics volatility forecasts.

Our results show, PSN outperforms its NN and statistical benchmarks in terms of annualized returns and information ratios. The NN forecast combinations, excluding the Bayesian Average model, present improved annualized returns and information ratios and in almost all cases outperform every individual NN performance. More specifically, the Kalman Filter outperforms all individual models and combination forecasts. We also note that the Kalman Filter forecasts are statistically different from their benchmarks under the Diebold–Marino test [11]. Finally we note that, after applying the time-varying leverage strategy, all models except ARMA show substantial increase in their trading performance.

Section 2 is a literature review of previous research on PSNs and combination forecasts, especially Kalman Filters. Section 3 follows the detailed description of the EUR/USD ECB fixing series, used as our dataset. Section 4 gives an overview of the benchmark models

* Corresponding author.
E-mail addresses: georgios.sermpinis@glasgow.ac.uk (G. Sermpinis), cdunis@tiscali.co.uk (C. Dunis), J.Laws@liverpool.ac.uk (J. Laws), c.stasinakis.1@research.gla.ac.uk (C. Stasinakis).

0167-9236/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2012.05.039

and the architectures of the NNs selected, while Section 5 describes the forecast combination methods we implemented. The statistical and trading performance of our models is presented in Sections 6 and 7. Finally, some concluding remarks are summarized in Section 8.

2. Literature review

The most common NN architecture is the MLP and seems to perform well at time-series financial forecasting [38], although the empirical evidence can be contradictory in many cases. For example, Tsaih et al. [53] attempt to forecast the S&P 500 stock index futures and in their application Reasoning Neural Networks perform better than MLPs. Lam [35] also examines the financial forecasting performance of feed-forward NNs and concludes that they fail to outperform the maximum benchmarks in all cases. Ince and Trafalis [31] forecast the EUR/USD, GBP/USD, JPY/USD and AUD/USD exchange rates with MLP and Support Vector Regression and their results show that MLP achieves less accurate forecasts. Finally, according to Alfaro et al. [2] the AdaBoost algorithm is superior to MPLs, when applied to the task of forecasting bankruptcy of European firms. On the other hand, Tenti [49] and Dunis and Huang [15] achieved encouraging results also by using RNNs to forecast the exchange rates. But the PSN architecture presents remarkable empirical evidence compared to both MLP and RNN. PSNs were first introduced by Ghosh and Shin [23] as architectures able to capture high-order correlations. Ghosh and Shin [23,24] also present results on their forecasting superiority in function approximation, when compared with a MLP network and a Higher Order Neural Network (HONN). Ghazali et al. [22] compare PSN with HONN and MLP in terms of forecasting and trading the IBM common stock closing price and the US 10-year government bond series. PSN presented improved statistical accuracy and annualized return compared with both benchmarks. Satisfactory forecasting results of PSN were presented by Hussain et al. [30] on the EUR/USD, the EUR/GBP and the EUR/JPY exchange rates using univariate series as inputs in their networks. On the other hand, Dunis et al. [16] also study the EUR/USD series with PSN and fail to outperform MLP, RNN and HONN in a simple trading application.

Bates and Granger [4] and Newbold and Granger [39] suggested combining rules based on variances–covariances of the individual forecasts, while Granger and Ramanathan [26] presented a regression combination forecast framework with encouraging results. According to Palm and Zellner [40], it is sensible to use Simple Average for combination forecasting, while Deutsch et al. [10] achieved substantially smaller squared forecast errors combining forecasts with changing weights. The regression framework, presented in the 90s, performs poorly though in many cases, which leads the research to turn to more sophisticated methods. For example, Chan et al. [8] suggested the use of Ridge Regression, while Swanson and Zeng [48] use Bayesian Information Criteria. However, in real applications there are also contradictory results regarding both these models (see Stock and Watson [46] and Rapach and Strauss [42]). Finally, Leigh et al. [36] presented novel experiments of combining pattern recognition, NNs and genetic algorithms, in order to forecast price changes for the NYSE Composite Index. From their approach stock market purchasing opportunities are identified and encouraging decision-making results are achieved.

Time-series analysis is often based on the assumption that the parameters are fixed. However, in reality financial data and the correlation structure between financial variables are time-varying. Harvey [28] and Hamilton [27] both suggest using state space modeling, such as Kalman Filter, for representing dynamic systems where unobserved variables (so-called 'state' variables) can be integrated within an 'observable' model. According to Goh and Mandic [25] the recursive Kalman Filter is suitable for processing complex-valued nonlinear, non-stationary signals and bivariate signals with strong component correlations. Kalman Filter is also considered an optimal time-varying financial forecast for financial markets [19]. Anandalingam and Chen [3] compare Kalman Filter with Bayesian combination forecast model, while Sessions and Chatterjee [44] conclude that recursive methods are found to be very effective. LeSage and Magura [37] extend the Granger–Ramanathan combination method by allowing time-varying weights and their methodology outperforms traditional and other forecast combinations. Terui and van Dijk [50] also suggest that the combined forecasts perform well, especially with time varying coefficients. Finally, Stock and Watson [46] try to forecast the output growth of seven countries and note that time-varying combination forecasts can lack in robustness, despite performing well in many cases.

3. The EUR/USD exchange rate and related financial data

The European Central Bank (ECB) publishes a daily fixing for selected EUR exchange rates: these reference mid-rates are based on a daily concentration procedure between central banks within and outside the European System of Central Banks, which normally takes place at 2.15 p.m. ECB time. The reference exchange rates are published both by electronic market information providers and on the ECB's website shortly after the concentration procedure has been completed. Although only a reference rate, many financial institutions are ready to trade at the EUR fixing and it is therefore possible to leave orders with a bank for business to be transacted at this level.

In this paper, we examine the EUR/USD over period 2002–2010, using the last two years for out-of-sample. In order to train our Neural Networks we further divided our in-sample dataset in two sub-periods (see more in Section 4.2) (Table 1).

The graph below shows the total dataset for the EUR/USD and its volatile trend since early 2008 (Fig. 1).

The EUR/USD time series, shown above, is non-normal and non-stationary. Jarque–Bera statistics confirm its non-normality at the 99% confidence interval with slight skewness and low kurtosis. To overcome the non-stationary issue, the EUR/USD series is transformed into a daily series of rate returns. So given the price level P_1, P_2,…, P_t, the return at time t is calculated as:

$$R_t = \left(\frac{P_t}{P_{t-1}}\right) - 1. \tag{1}$$

The stationary property of the EUR/USD return series is confirmed at the 1% significance level (ADF and PP test statistics) and its summary statistics are shown in Fig. 2. From those it is obvious that the slight skewness and low kurtosis remain. The Jarque–Bera statistic confirms again that the EUR/USD series is non-normal at the 99% confidence interval. For more details on Jarque–Bera statistics see Jarque and Bera [32].

In the absence of any formal theory behind the selection of the inputs of a Neural Network, we conduct some Neural Network experiments and a sensitivity analysis on a pool of potential inputs in the training dataset in order to help our decision. Our aim is to select the set of inputs for each network which is the more likely to lead to the best trading performance in the out-of-sample dataset. In our application, we select as inputs the set of variables that provide the higher trading performance for each network in the test sub-period. To our surprise this set of inputs is identical for all Neural Network models. These sets of inputs for each network are presented in Table 2 below.[1]

[1] We also explored as inputs autoregressive terms of other exchange rates (e.g. the USD/JPY and GBP/JPY exchange rates), commodities prices (e.g. Gold Bullion and Brent Oil) and stock market prices (e.g. FTSE100 and DJIA). However, the set of inputs presented in Table 2 gave our NNs the highest trading performance in the training period and were thus retained.

Table 1
The EUR/USD dataset–Neural Networks' training datasets.

Periods	Trading days	Start date	End date
Total dataset	2295	3/01/2002	31/12/2010
Training dataset (*in-sample*)	1270	3/01/2002	29/12/2006
Test dataset (*in-sample*)	511	02/01/2007	31/12/2008
Validation dataset (*out-of-sample*)	514	02/01/2009	31/12/2010

4. Forecasting models

4.1. Benchmark forecasting models

In this paper we use two traditional forecasting strategies, the Naive Strategy and the Auto-Regressive Moving Average (ARMA) model, in order to benchmark the efficiency of the NNs' trading performance.

4.1.1. Naive Strategy

The Naive Strategy is considered to be the simplest strategy to predict the future. That is to accept as a forecast for time $t+1$, the value of time t, assuming that the best prediction is the most recent period change. Thus, the model takes the form:

$$\hat{Y}_{t+1} = Y_t \tag{2}$$

where Y_t is the actual rate of return at time t and $\hat{Y}_{t+1}$is the forecast rate of return at time $t+1$. In order to evaluate the Naive trading performance, a simulated strategy is used.

4.1.2. Auto-Regressive Moving Average Model (ARMA)

The ARMA model is based on the assumption that the current value of a time-series is a linear combination of its previous values plus a combination of current and previous values of the residuals [6]. Thus, the ARMA model embodies autoregressive and moving average components and can be specified as below:

$$\begin{aligned} Y_t &= \varphi_0 + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \ldots + \varphi_p Y_{t-p} \\ &\quad + \varepsilon_t - w_1 \varepsilon_{t-1} - w_2 \varepsilon_{t-2} - \ldots - w_q \varepsilon_{t-q} \end{aligned} \tag{3}$$

Where:

- Y_t is the dependent variable at time t
- $Y_{t-1}, Y_{t-2}, \ldots Y_{t-p}$ are the lagged dependent variables
- $\varphi_0, \varphi_1, \ldots, \varphi_p$ are the regression coefficients
- ε_t is the residual term
- $\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$ are the previous values of the residual terms
- w_1, $w_2, \ldots, w_q$ are the residual weights.

Based on the In-Sample correlogram (Training and Test subsets), a restricted ARMA (13,13) model was chosen as the best for an out-of-sample estimation (See Appendix A). The ARMA model, used in this paper, can be specified as follows:

$$\begin{aligned} Y_t &= 0.0288 - 0.2689 Y_{t-3} + 0.6028 Y_{t-4} - 0.3921 Y_{t-6} - 0.6884 Y_{t-9} \\ &\quad + 0.3641 Y_{t-13} + 0.2638 \varepsilon_{t-3} - 0.59 \varepsilon_{t-4} + 0.3916 \varepsilon_{t-6} \\ &\quad + 0.6227 \varepsilon_{t-9} - 0.3165 \varepsilon_{t-13}. \end{aligned} \tag{4}$$

The evaluation of the ARMA model selected comes in terms of trading performance.

4.2. Neural Networks (NNs)

Neural Networks exist in several forms in the literature. The most popular architecture is the Multi-Layer Perceptron (MLP). A standard Neural Network has at least three layers. The first layer is called the input layer (the number of its nodes corresponds to the number of explanatory variables). The last layer is called the output layer (the number of its nodes corresponds to the number of response variables). An intermediary layer of nodes, the hidden layer, separates the input from the output layer. Its number of nodes defines the amount of complexity the model is capable of fitting. In addition, the input and hidden layers contain an extra node called the bias node. This node has a fixed value of one and has the same function as the intercept in traditional regression models. Normally, each node of one layer has connections to all the other nodes of the next layer.

The network processes information as follows: the input nodes contain the value of the explanatory variables. Since each node connection represents a weight factor, the information reaches a single hidden layer node as the weighted sum of its inputs. Each node of the hidden layer passes the information through a nonlinear activation function and passes it on to the output layer if the calculated value is above a threshold.

The training of the network (which is the adjustment of its weights in the way that the network maps the input value of the training data to the corresponding output value) starts with randomly chosen weights and proceeds by applying a learning algorithm called backpropagation of errors[2] [45]. The learning algorithm simply tries to find those weights which minimize an Error Function (normally the sum of all squared differences between target and actual values). Since networks with sufficient hidden nodes are able to learn the training data (as well as their outliers and their noise) by heart, it is crucial to stop the training procedure at the right time to prevent overfitting (this is called 'early stopping'). This can be achieved by dividing the dataset into 3 subsets respectively called the training and test sets used for simulating the data currently available to fit and tune the model and the validation set used for simulating future values. The training of a network is stopped when the mean squared forecasted error is at minimum in the test-sub period. The network parameters are then estimated by fitting the training data using the above mentioned iterative procedure (backpropagation of errors). The iteration length is optimized by maximizing the forecasting accuracy for the test dataset. Then the predictive value of the model is evaluated applying it to the validation dataset (out-of-sample dataset).

Since the starting point for each network is a set of random weights, forecasts can differ between networks. In order to eliminate any variance between our NN forecasts and to add robustness to our results, we used the Simple Average of a committee of 10 NNs, which presented the highest profit in the training sub-period. This was necessary in order to eliminate any outlier network that could jeopardize our conclusions. The characteristics of the NNs used in this paper are presented in Appendix B.

4.2.1. The Multi-Layer Perceptron Model (MLP)

MLPs are feed-forward layered NN, trained with a back-propagation algorithm. According to Kaastra and Boyd [34], they are the most commonly used types of artificial networks in financial time-series forecasting. The training of the MLP network is processed on a three-layered architecture, as described above. A typical MLP model is shown in Fig. 3.

Where:

- $x_t^{[n]} (n = 1, 2, \cdots, k+1)$Z are the inputs (including the input bias node) at time t
- $h_t^{[m]} (m = 1, 2, \ldots, j+1)$Z are the hidden nodes outputs (including the hidden bias node) at time t
- $\hat{Y}_t$Z is the MLP output
- u_{jk}, w_j are the network weights

[2] Backpropagation networks are the most common multi-layer networks and are the most commonly used type in financial time series forecasting [33].

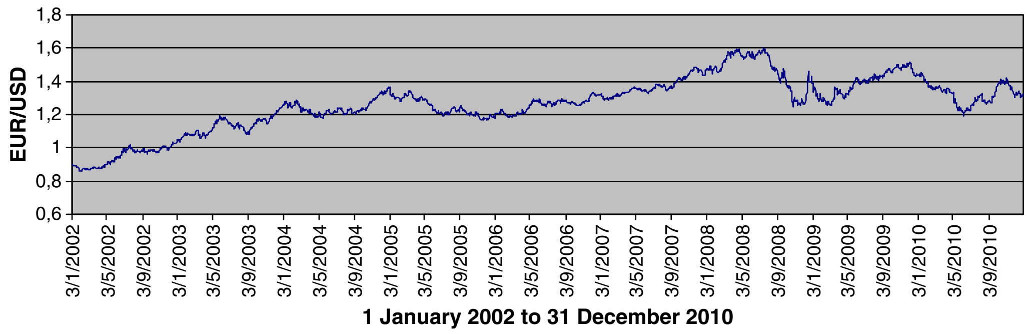


Fig. 1. EUR/USD Frankfurt daily fixing prices.

- is the transfer sigmoid function

$$S(x) = \frac{1}{1+e^{-x}} \tag{5}$$

- is a linear function

$$F(x) = \sum_i x_i. \tag{6}$$

The Error Function to be minimized is

$$E\left(c, w_j\right) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(w_k, c)\right)^2 \tag{7}$$

with y_t being the target value. The evaluation of the MLP model selected comes in terms of trading performance.

4.2.2. The Recurrent Neural Network (RNN)

The next model is the recurrent Neural Network. While a complete explanation of RNN models is beyond the scope of this paper, we present below a brief explanation of the significant differences between RNN and MLP architectures. For an exact specification of Recurrent Networks, see Elman [21].

A simple Recurrent Network has an activation feedback which embodies short-term memory. The advantages of using Recurrent Networks over feed-forward networks for modeling non-linear time series have been well documented in the past. However, as mentioned by Tenti [49], "the main disadvantage of RNNs is that they require substantially more connections, and more memory in simulation than the standard back-propagation networks" (p. 569), thus resulting in a substantial increase in computational time. However, having said this, RNNs can yield better results in comparison with simple MLPs due to the additional memory inputs. A simple illustration of the architecture of an Elman RNN is presented below (Fig. 4).

Where:

- $x_t^{[n]}(n=1,2,\ldots,k+1), u_t^{[1]}, u_t^{[2]}$Z are the RNN inputs at time t (including bias node)
- $\tilde{y}_t$ is the output of the RNN
- $d_t^{[f]}(f=1,2)$ and $w_t^{[n]}(n=1,2,\ldots,k+1)$are the weights of the network
- $U_t^{[f]}, f=(1,2)$ is the output of the hidden nodes at time t
- is the transfer sigmoid function: $S(x) = \frac{1}{1+e^{-x}}$
- is a linear function: $F(x) = \sum_i x_i$.

The Error Function to be minimized is

$$E(d_t, w_t) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(d_t, w_t)\right)^2 \tag{8}$$

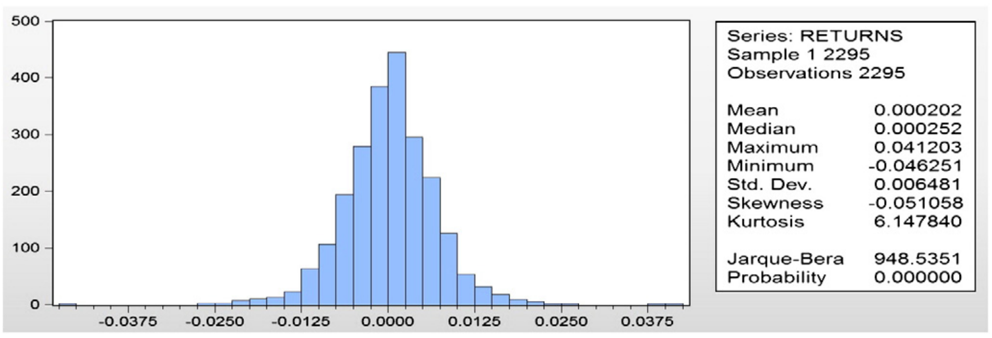
In short, the RNN architecture can provide more accurate outputs because the inputs are (potentially) taken from all previous values (see inputs $U_{j-1}^{[1]}$ and $U_{j-1}^{[2]}$in the figure above). The evaluation of the RNN model selected comes in terms of trading performance.

4.2.3. The Psi-Sigma Neural Network (PSN)

The PSNs are a class of Higher Order Neural Networks with a fully connected feed-forward structure. Ghosh and Shin [23] were the first to introduce the PSN, trying to reduce the numbers of weights and connections of a Higher Order Neural Network. Their goal was to combine the fast learning property of single-layer networks with the mapping ability of Higher Order Neural Networks and avoid increasing the required number of weights. The training process is again three-layered. The PSN architecture of a one-output layer is shown in Fig. 5.

Where:

- x_t $(n=1,2,\ldots,k+1)$ are the model inputs (including the input bias node)



Series: RETURNS
Sample 1 2295
Observations 2295

Mean	0.000202
Median	0.000252
Maximum	0.041203
Minimum	-0.046251
Std. Dev.	0.006481
Skewness	-0.051058
Kurtosis	6.147840
Jarque-Bera	948.5351
Probability	0.000000

Fig. 2. EUR/USD returns summary statistics.

Table 2
Explanatory variables.

Number	Explanatory variables	Lag[a]
1	EUR/USD exchange rate return	1
2	EUR/USD exchange rate return	2
3	EUR/USD exchange rate return	4
4	EUR/USD exchange rate return	5
5	EUR/USD exchange rate return	8
6	EUR/USD exchange rate return	10
7	EUR/GBP exchange rate return	1
8	EUR/GBP exchange rate return	2
9	EUR/JPY exchange rate return	1

[a] In our application the term 'Lag 1' means that today's closing price is used to forecast the tomorrow's one.

- $\tilde{y}_t Z$ is the PSN output
- w_j ($j=1,2..,k$) are the adjustable weights (K is the desired order of the network)
- $h(x) = \sum_i x_i$ is the hidden layer activation function (9)
- $\sigma(x) = \dfrac{1}{1+e^{-xc}}$ is the output sigmoid activation function (10)

(c the adjustable term)

The Error Function minimized is

$$E\left(c, w_j\right) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \tilde{y}_t(w_k, c)\right)^2 \quad (11)$$

with y_t being the target value. The training of the PSN is achieved also with the backpropagation and the 'early-stopping' procedure, as described in Section 4.2. The structure of the PSN and the sigmoid output function require the normalization of the inputs and the de-normalization of the outputs. Based on Ghazali et.al [22], our inputs are normalized between the values of 0.2 and 0.8 and at the end the outputs of the network are de-normalized back.

For example let us consider a Psi Sigma network which is fed with a N+1 dimensional input vector $x=(1,x_1,\ldots,x_N)^T$. These inputs are weighted by K weight factors$w_j=(w_{0j},w_{1j},\ldots,w_{Nj})^T j=1,2,..K$ and summed by a layer of K summing units, where K is the desired order of the network. So the output of the j-th summing unit,h_j in the hidden layer, is given by: $h_j = w_j^T x = \sum_{k=1}^{N} w_{kj}x_k + w_{oj}$,j=1,2,…, K while the output $\tilde{y}$ of the network is given by $\tilde{y} = \sigma\left(\prod_{j=1}^{K} h_j\right)$. Note that by using products in the output layer we directly incorporate the capabilities of higher order networks with a smaller number of weights and processing units. For example, a k-th degree higher order Neural Network with d inputs needs $\sum_{i=0}^{k} \frac{(d+i-1)!}{i!(d+1)!}$ weights if all products of up to k components are to be incorporated while a similar Psi Sigma network needs only $(d+1)*k$ weights. Also note that the sigmoid function is neuron adaptive. As the network is trained not only the weights but also c in Eq. (10) is adjusted. This strategy seems to provide better fitting properties and increases the approximation capability of a Neural Network by introducing an extra variable in the estimation, compared to classical architectures with sigmoidal neurons [54].

The price for the flexibility and speed of Psi Sigma networks is that they are not universal approximators. We need to choose a suitable order of approximation (or else the number of hidden units) by considering the estimated function complexity, amount of data and amount of noise present. To overcome this, our code runs simulations for orders two to six and then it presents the best network. The evaluation of the PSN model selected comes in terms of trading performance.

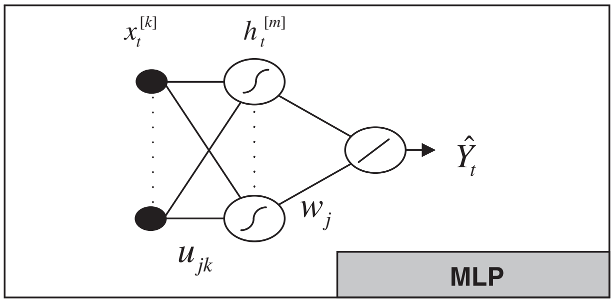


Fig. 3. A single output, fully connected MLP model (bias nodes are not shown for simplicity).

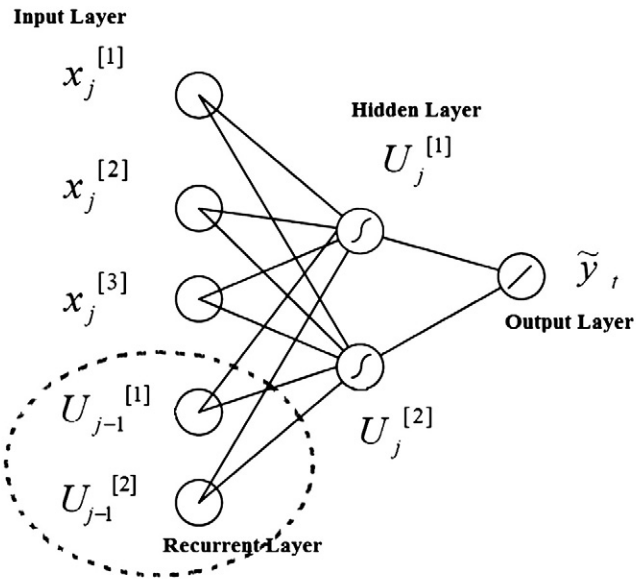


Fig. 4. Elman RNN with two nodes in the hidden layer.

5. Forecasting combination techniques

In this section we present the five techniques that we used to combine our NN forecasts. It is important to outline that a forecast combination targets either to follow the trend of the best individual forecast ('*combining for adaptation*') or to significantly outperform each one of them ('*combining for improvement*') [57]. Consequently, we decided to exclude the ARMA and the naive strategy from our combination techniques. Both strategies present a considerably worse trading performance than their NNs' counterparts both in-sample and out-of-sample. Therefore, their inclusion in our combination techniques will deteriorate their performance rather than improve it.

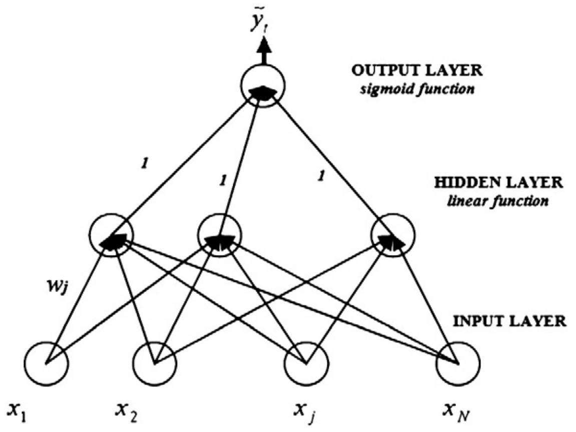


Fig. 5. A PSN with one output layer.

Table 3
Summary of in-sample statistical performance.

	Traditional techniques		Neural Networks			Forecast combinations				
	NAIVE	ARMA	MLP	RNN	PSN	Simple Average	Bayesian Average	GRR	LASSO	Kalman Filter
MAE	0.0065	0.0045	0.0044	0.0042	0.0039	0.0037	0.0037	0.0035	0.0038	0.0033
MAPE	399.44%	122.20%	97.13%	93.35%	89.43%	84.98%	85.13%	82.78%	87.63%	71.51%
RMSE	0.0086	0.0060	0.0053	0.0050	0.0041	0.0036	0.0036	0.0032	0.0037	0.0023
Theil-U	0.7021	0.6948	0.6686	0.5087	0.4292	0.4522	0.4625	0.4245	0.4613	0.2713

5.1. Simple Average

The first forecasting combination technique used in this paper is Simple Average, which can be considered a benchmark forecast combination model. Given the three NNs' forecasts f^t_{MLP},f^t_{RNN},f^t_{PSN} at time t, the combination forecast at time t is calculated as:

$$f^t_{c_{NNs}} = \left(f^t_{MLP} + f^t_{RNN} + f^t_{PSN}\right)/3. \tag{12}$$

5.2. Bayesian averaging

A Bayesian Average model specifies optimal weights for the combination forecast based on the Akaike Information Criterion (AIC) and Schwarz Bayesian Information Criterion (SIC). According to Buckland et al. [7] the Bayesian weights using AIC, can be estimated as:

$$w_{AIC,i} = \frac{e^{-0.5\Delta AIC_i}}{\sum_{j=1}^{3} e^{-0.5\Delta AIC_j}} \tag{13}$$

Where:

- $i = 1,2,3$ for f_{MLP},f_{RNN},f_{PSN}respectively
-

$$\Delta AIC_i = AIC_i – AIC_{i,min}. \tag{14}$$

Based on the above, the combination forecast at time t is $f^t_{c_{NNs}} = \left(\sum_{i=1}^{3} w_{AIC,i} f^t_i\right)/3$ and in our case the AIC Bayesian models take the following form:

$$f^t_{c_{AIC}} = \left(0.334209988 f^t_{MLP} + 0.330831059 f^t_{RNN} + 0.334958953 f^t_{PSN}\right)/3. \tag{15}$$

The Bayesian Average weights for SIC are defined similarly and in our case the SIC Bayesian model is specified as follows:

$$f^t_{c_{SIC}} = \left(0.334210009 f^t_{MLP} + 0.330831081 f^t_{RNN} + 0.33495891 f^t_{PSN}\right)/3. \tag{16}$$

Eqs. (15) and (16) are similar as the AIC and SIC criteria for our NNs in the in-sample period are very close. For that reason we will present only the Bayesian Average based on the AIC criterion, the one that presented a marginally better trading performance in-sample. Nonetheless, the weights are in favor (maximized) of PSN, namely the model with the minimum AIC and SIC respectively. For details on the exact calculation of the AIC and SIC and their Bayesian Average weights see Appendix C.

5.3. Granger and Ramanathan Regression Approach (GRR)

According to Bates and Granger [4], a combining set of forecasts outperforms the individual forecasts that the set consists of. Taking this basic idea one step further, Granger and Ramanathan [26] suggested three regression models as follows:

$$f_{c1} = a_0 + \sum_{i=1}^{n} a_i f_i + \varepsilon_1 \tag{GRR – 1}$$

$$f_{c2} = \sum_{i=1}^{n} a_i f_i + \varepsilon_2 \tag{GRR – 2}$$

$$f_{c3} = \sum_{i=1}^{n} a_i f_i + \varepsilon_3, \quad where \quad \sum_{i=1}^{n} a_i = 1 \tag{GRR – 3}$$

Where

- f_i, $i = 1,…,n$ are the individual one-step-ahead forecasts,
- f_{c1}, f_{c2}, f_{c3} are the combination forecast of each model,
- α_0 is the constant term of the regression
- α_i are the regression coefficients of each model
- ε_1, ε_2, ε_3 are the error terms of each regression model.

The GRR-1 model, which was selected for our case, is usually preferred in order to avoid forecast errors correlated with the individual forecasts f_i [48]. Thus, the GRR model at time t used in this paper is specified as shown below:

$$f^t_{c_{NNs}} = 0.0422 + 35.023 f^t_{MLP} + 13.461 f^t_{RNN} + 56.132 f^t_{PSN} + \varepsilon_t. \tag{17}$$

However, the variety of data and the biased and correlated forecasts raise questions on GRR model selection or modification, which are further discussed in the literature [9,12].

5.4. Least Absolute Shrinkage and Selection Operator (LASSO)

The LASSO Regression is a class of Shrinkage or Regularization Regressions, which applies when multicollinearity exists among the regressors [47]. The main difference between this technique and the Ordinary Least Squares (OLS) Regression is that LASSO method also

Table 4
Summary of out-of-sample statistical performance.

	Traditional techniques		Neural Networks			Forecast combinations				
	NAIVE	ARMA	MLP	RNN	PSN	Simple Average	Bayesian Average	GRR	LASSO	Kalman Filter
MAE	0.0084	0.0059	0.0058	0.0056	0.0048	0.0048	0.0048	0.0047	0.0046	0.0044
MAPE	405.62%	131.20%	112.37%	105.97%	97.88%	94.07%	93.76%	92.83%	92.05%	88.37%
RMSE	0.0107	0.0077	0.0061	0.0060	0.0054	0.0053	0.0051	0.0049	0.0053	0.0043
Theil-U	0.7958	0.8749	0.7301	0.6001	0.4770	0.5672	0.5598	0.5297	0.6142	0.5212

Table 5
Summary results of Diebold–Mariano statistic for MSE and MAS loss functions.

	NAIVE	ARMA	MLP	RNN	PSN	Simple Average	Bayesian Average	GRR	LASSO
s_{MSE}	−9.307	−9.321	−6.244	−5.698	−5.184	−4.869	−4.896	−4.351	−4.112
s_{MAE}	−9.845	−9.832	−9.189	−8.881	−8.159	−7.851	−7.873	−7.679	−7.352

minimizes the residual squared error, by adding a coefficient constraint (similarly to Ridge Regression [8]).

Compared to Ridge Regression, LASSO best applies in samples of few variables with medium/large effect such in our case [29]. For more details on the mathematical specifications of LASSO see Wang et al. [55]. Given the vectors of independent and dependent variables:

$$\begin{pmatrix} X_1^T \\ \vdots \\ X_N^T \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1N} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NN} \end{pmatrix}, \quad Y = (y_1, \ldots, y_N)^T \tag{18}$$

and the training data $\{(X_1,y_1),\ldots,(X_N,y_N)\}$, the LASSO coefficients are estimated based on the following argument:

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{d} \beta_i x_{ij} \right)^2 \right\} \quad subject\ to \quad \sum_{j=1}^{d} \left| \beta_j \right| \leq k, k > 0. \tag{19}$$

The argument (19) is based on Breiman's non-negative garrote minimization process [58]. Here k stands for the '*tuning parameter*', because it controls the amount of shrinkage applied to the coefficients [52]. In our case, we experimented with various values of k in the in-sample period and we concluded that the best results in terms of trading performance are acquired when the constraint takes the following form:

$$|\beta_{MLP}| + |\beta_{RNN}| + |\beta_{PSN}| \leq 10.6. \tag{20}$$

Subject to this constraint our model takes the form:

$$f^t_{c_{NNs}} = 3.284 f^t_{MLP} + 1.591 f^t_{RNN} + 5.623 f^t_{PSN} + \varepsilon_t. \tag{21}$$

This LASSO constraint makes the model adaptive, since it creates a penalization balance on each estimate, by leading some coefficients to zero or close to zero (see the unconstrained regression of GGR (17) compared to LASSO (21)).

5.5. Kalman Filter

Kalman Filter is an efficient recursive filter that estimates the state of a dynamic system from a series of incomplete and noisy measurements. The time-varying coefficient combination forecast suggested in this paper is shown below:

Measurement equation:

$$f^t_{c_{NNs}} = \sum_{i=1}^{3} a_i^t f_i^t + \varepsilon_t, \quad \varepsilon_t \sim NID\left(0, \sigma_\varepsilon^2\right) \tag{22}$$

State equation:

$$a_i^t = a_i^{t-1} + n_t, \quad n_t \sim NID\left(0, \sigma_n^2\right) \tag{23}$$

Where:

- f^t_{cNNs} is the dependent variable (combination forecast) at time t
- $f_i^t (i = 1,2,3)$ are the independent variables (individual forecasts) at time t
- $a_i^t (i = 1,2,3)$ are the time-varying coefficients at time t for each NN
- ε_t, n_t are the uncorrelated error terms (noise).

When Kalman Filter is applied, all $a_i^t$are estimated in time, along with the log-likelihood of the model based on the observations up to time t. Then the likelihood function is maximized with a numerical optimization algorithm, based onσ_n^2. The updated alphas for the state equation are estimated at time t based on the new observations at time t and then the state estimates are propagated in time $t+1$. Thus, the Kalman Filter update can be considered as the best unbiased linear estimate of the individual forecastsf_i^t, given $f_{cNNs}{}^t$ and the prior information. After Kalman Filter and the numerical optimization algorithm, a Kalman smoothing algorithm should be applied, because the accuracy is increased to the end of the sample. This algorithm 'smoothes' the estimates by running backwards in time and using information acquired after time t and allows our model to compute forecasts, which use all available measurement data over the forecast sample.

Following Welch and Bishop [53] and Dunis et al. [14], in our study the alphas are calculated by a simple random walk and we initialized $\varepsilon_1 = 0$. Based on the above, our Kalman Filter model has as a final state the following:

$$f^t_{c_{NNs}} = 5.80 f^t_{MLP} + 1.16 f^t_{RNN} + 75.89 f^t_{PSN} + \varepsilon_t \tag{24}$$

Table 6
Summary of in-sample trading performance.

	Traditional techniques		Neural Networks			Forecast combinations				
	NAIVE	ARMA	MLP	RNN	PSN	Simple Average	Bayesian Average	GRR	LASSO	Kalman Filter
Annualized return (excluding costs)	1.49%	13.87%	23.19%	26.14%	28.10%	32.74%	32.39%	33.99%	30.57%	42.63%
Annualized volatility	9.68%	9.70%	9.38%	9.59%	9.23%	9.51%	9.52%	9.49%	9.54%	9.35%
Information ratio (excluding costs)	0.15	1.43	2.47	2.73	3.05	3.44	3.4	3.58	3.21	4.56
Maximum drawdown	−8.59%	−6.52%	−5.91%	−6.55%	−6.55%	−6.55%	−6.55%	−6.55%	−6.55%	−6.66%
Annualized transactions	130	100	121	136	74	107	106	104	106	121
Transaction costs	0.91%	0.70%	0.85%	0.95%	0.52%	0.75%	0.74%	0.73%	0.74%	0.85%
Annualized return (including costs)	0.58%	13.17%	22.34%	25.19%	27.58%	31.99%	31.65%	33.26%	29.83%	41.78%
Information ratio (including costs)	0.06	1.36	2.38	2.63	2.99	3.36	3.32	3.50	3.13	4.47

Table 7
Summary of out-of-sample trading performance.

	Traditional techniques		Neural Networks			Forecast combinations				
	NAIVE	ARMA	MLP	RNN	PSN	Simple Average	Bayesian Average	GRR	LASSO	Kalman Filter
Annualized return (excluding costs)	−4.80%	10.60%	14.80%	16.07%	18.37%	16.37%	16.59%	16.99%	20.23%	28.79%
Annualized volatility	12.03%	11.07%	11.83%	11.02%	10.89%	10.85%	10.85%	11.02%	10.99%	10.92%
Information ratio (excluding costs)	−0.4	0.96	1.25	1.46	1.69	1.51	1.53	1.54	1.84	2.64
Maximum drawdown	−6.41%	−6.23%	−6.23%	−6.23%	−6.31%	−6.31%	−6.31%	−6.31%	−6.31%	−6.31%
Annualized transactions	77	54	71	71	76	70	71	63	69	73
Transaction costs	0.54%	0.38%	0.50%	0.50%	0.53%	0.49%	0.50%	0.44%	0.48%	0.51%
Annualized return (including costs)	−5.34%	10.22%	14.30%	15.57%	17.84%	15.88%	16.09%	16.55%	19.75%	28.28%
Information ratio (including costs)	−0.44	0.92	1.21	1.41	1.64	1.46	1.48	1.50	1.80	2.59

From the above equation we note that the Kalman filtering process favors the PSN model. This is what one would expect, since it is the model that performs best individually.

In order to achieve optimal Kalman Filter estimation, it is important though to introduce a noise ratio.

$$n_r = \sigma_\varepsilon^2 / \sigma_n^2 \tag{25}$$

The results are becoming more adaptive when the noise ratio rises [14]. When$\sigma_n^2 = 0$, the model transforms to the typical OLS model. Appendix D describes the Kalman filtering and smoothing process.

6. Statistical performance

As it is standard in the literature, in order to evaluate statistically our forecasts, the RMSE, the MAE, the MAPE and the Theil-U statistics are computed (see among others Dunis and Williams [20] and Dunis and Chen [13]). The statistical analysis will provide some information regarding the accuracy of our forecasts and strengthen our conclusions. The RMSE and MAE statistics are scale-dependent measures but give a basis to compare volatility forecasts with the realized volatility while the MAPE and the Theil-U statistics are independent of the scale of the variables. In particular, the Theil-U statistic is constructed in such a way that it necessarily lies between zero and one, with zero indicating a perfect fit. A more detailed description of these measures can be found on Pindyck and Rubinfeld [41] and Theil [51], while their mathematical formulas are presented in Appendix E. For all four of the error statistics retained (RMSE, MAE, MAPE and Theil-U) the lower the output, the better the forecasting accuracy of the model concerned. In Tables 3 and 4 we present the in-sample period and out-of-sample periods respectively.

We note that from our individual forecasts, the PSN outperformed all other models in both the in-sample and out-of-sample periods. Similarly, for our forecast combination methodologies the Kalman Filter beat its benchmarks for the four statistical criteria retained in both estimation periods. Adding to the above statistical performance of the Kalman Filter, the Diebold–Mariano [11] statistic for predictive accuracy is also computed for both MSE and MAE loss functions (for more details on the Diebold–Mariano statistic see Appendix F). The results of the Diebold–Mariano statistic, comparing Kalman filter with each other method, are summarized in Table 5.

Table 8
Classification of leverage in sub-periods.

	Extremely low vol.	Medium low vol.	Lower high vol.	Upper high vol.	Medium high vol.	Extremely high vol.
Leverage	2.5	2	1.5	1	0.5	0

From the above table we note that the null hypothesis of equal predictive accuracy is rejected for all comparisons and for both loss functions at 5% confidence interval, since the test results $|s_{MSE}| > 1.96$ and $|s_{MAE}| > 1.96$. Moreover, the statistical superiority of the Kalman Filter forecasts is confirmed as for both loss functions the realizations of the Diebold–Mariano [11] statistic are negative.[3] We also note that our second best model in statistical terms, the LASSO regression, has the closest forecasts with Kalman Filter.

7. Trading performance

7.1. Trading strategy and transaction costs

The trading strategy applied in this paper is to go or stay ‘long’ when the forecast return is above zero and go or stay ‘short’ when the forecast return is below zero. The ‘long’ and ‘short’ EUR/USD positions are defined as buying and selling Euros at the current price respectively. The transaction costs for a tradable amount, say USD 5–10 million, are about 1 pip (0.0001 EUR/USD) per trade (one way) between market makers. But since we consider the EUR/USD time series as a series of middle rates, the transaction costs are one spread per round trip. With an average exchange rate of EUR/USD of 1.369 for the out-of-sample period, a cost of 1 pip is equivalent to an average cost of 0.007% per position.

7.2. Trading performance before leverage

The trading performance measures and their calculation description are presented in Appendix E. In Table 6 we present the in-sample trading performance of our models and forecast combinations before and after transaction cost.

We note that all our models present a positive trading performance after transaction costs. From our single forecasts the PSN outperforms each NN and statistical benchmark in terms of annualized return and information ratio. Our other two artificial intelligence models, the RNN and the MLP, present the second and third best trading performance respectively. Concerning our forecast combinations we observe that the Kalman Filter presents the best trading performance with an annualized return of 41.78% and an information ratio of 4.47 after transaction costs. It is also worth noting that all our forecast combinations outperform our best single forecast, the PSN in terms of trading performance. In Table 7 below we present the out-of-sample performance of our models before and after transaction costs.

[3] In our exercise we apply the Diebold–Mariano test to couples of forecasts (Kalman Filter vs. another forecasting model). A negative realization of the Diebold–Mariano test statistic indicates that the first forecast (Kalman Filter) is more accurate than the second forecast. The lower the negative value, the more accurate are the Kalman Filter forecasts.

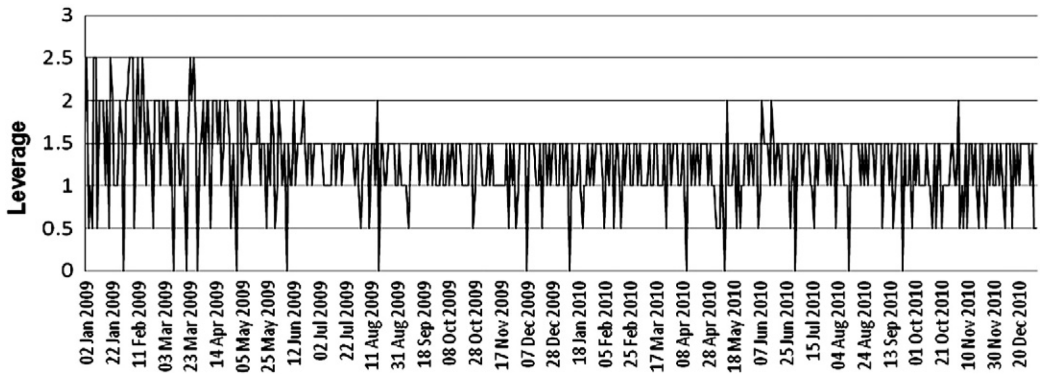


Fig. 6. Leverages assigned in the out-of-sample period.

From the last two rows of Table 7, we note that the PSN continues to outperform all other single forecasts in terms of trading performance. From our forecast combinations, only the Kalman Filter and the LASSO methods seem to beat our best single forecast. The Simple Average, Bayesian Average and GRR methods who demonstrated a better performance in the in-sample period seem unable to maintain this superiority in the out-of-sample period. Moreover, we note that the trading performance of the Bayesian Average and Simple Average strategies is very close. This was expected as the AIC and the BIC information criteria for our 3 NNs are very close in the in-sample period. On the other hand, the GRR strategy still outperforms the MLP and the RNN models in terms of annualized return and information ratio. That could be thought as a trend to adapt to the best individual performance (‘*combining for adaptation*’ [57]). We also note that the Kalman Filter achieves a 10% higher annualized return than our second best methodology, the LASSO regression. It seems that the ability of Kalman Filter to provide efficient computational recursive means to estimate the state of our process gives it a considerable advantage compared to our fixed parameters combination models.

7.3. Leverage to exploit high information ratios

In order to further improve the trading performance of our models we introduce a leverage based on RiskMetrics one day ahead volatility forecasts[4] (for more details on RiskMetrics model see Appendix G). The intuition of the strategy is to avoid trading when volatility is very high while at the same time exploiting days when the volatility is relatively low. As mentioned by Bertolini [5], there are few papers on market-timing techniques for foreign exchange, with the notable exception of Dunis and Miao [17,18]. The opposition between market-timing techniques and time-varying leverage is only apparent as time-varying leverage can also be easily achieved by scaling position sizes inversely to recent risk measures behavior.

Firstly, we forecast with RiskMetrics the one day ahead realized volatility of the EUR/USD exchange rate in the test and validation sub-periods. Then, following Dunis and Miao [17,18] we split these two periods into six sub-periods, ranging from periods with extremely low volatility to periods experiencing extremely high volatility. Periods with different volatility levels are classified in the following way: first the average (μ) difference between the actual volatility in day t and the forecasted for day t + 1 and its ‘volatility’ (measured in terms of standard deviation σ) are calculated; those periods where the difference is between μ plus one σ are classified as ‘Lower high vol. periods’. Similarly, ‘Medium high vol.’ (between $\mu+\sigma$ and $\mu+2\sigma$) and ‘Extremely high vol.’ (above $\mu+2\sigma$) periods can be defined. Periods with low volatility are also defined following the same 1σ and 2σ approach, but with a minus sign.

For each sub-period a leverage is assigned starting with 0 for periods of extremely high volatility to a leverage of 2.5 for periods of extremely low volatility (see for leverage factors [17,18]). Table 8 below presents the sub-periods and their relevant leverages.

The parameters of our strategy (μ and σ) are updated every three months by rolling forward the estimation period. So for example, for the first three months of our validation period, μ and σ are computed based on the eighteen months of the test sub-period. For the following three months, the two parameters are computed based on the last fifteen months of our test sub-period and the first three of the validation sub-period. The leverages assigned in the days of the out-of-sample period, based on the above strategy are summarized in the following figure (Fig. 6).

The cost of leverage (interest payments for the additional capital) is calculated at 1.75% p.a. (that is 0.0069% per trading day[5]). Our final results are presented in Table 9 below.

The most striking performance achieved by the time-varying leverage strategy is the significant reduction in the maximum drawdown, the essence of risk for an investor in financial markets. Not only do all models, except ARMA, experience a higher performance in terms of return or risk-adjusted return, but maximum drawdowns are reduced by as much as 50%, from 6.31% to 3.38% in the case of the Kalman Filter combination! Even the naive strategy seems to try to invert its previous discouraging performance (see Table 9). The PSN still outperforms every NN and increases its annualized profit over 3%. Similarly our Bayesian Average and Simple Average combination methods present a 3% increase of annualized return, but they still cannot outperform the PSN and RNN individual performance. Our other two forecast combination techniques, the GGR and the LASSO, also present an increased annualized return and information ratio. Finally, the Kalman Filter continues to present a remarkable trading performance with the highest information ratio and a 5.67% increase in terms of annualized return. When transaction and leverage costs are included, the profit decreases, but the trend of the results is not affected. That allows us to conclude, that in all cases the Kalman Filter

[4] We also explored a GJR (1,1) model in forecasting volatility. Its statistical accuracy in the test sub-period in terms of the MAE, MAPE, RMSE and the Theil-U statistics is only slightly better compared with RiskMetrics. However, when we measure the utility of GJR in terms of trading efficiency for our models within the context of our strategy in the test sub-period, our results in terms of annualized returns are slightly better with RiskMetrics for most of our models. Moreover, RiskMetrics is simpler to implement than the more complicated GJR. Therefore, we choose to present in this paper the results obtained with RiskMetrics. The results obtained with GJR, which are very close to the ones presented here, are available upon request. It is also worth noting that the ranking of our models in terms on information ratio and annualized return is the same whether we use GJR or RiskMetrics.

[5] The interest costs are calculated by considering a 1.75% interest rate p.a. (the Euribor rate at the time of calculation) divided by 252 trading days. In reality, leverage costs are also applied during non-trading days so that we should calculate the interest costs using 360 days per year. But for the sake of simplicity, we use the approximation of 252 trading days to spread the leverage costs of non-trading days equally over the trading days. This approximation prevents us from keeping track of how many non-trading days we hold a position.

Table 9
Summary of out-of-sample trading performance — final results[6]

[6] Not taken into account the interest that could be earned during times where the capital is not traded (non-trading days) or not fully invested and could therefore be invested.

	Traditional techniques		Neural Networks			Forecast combinations				
	NAIVE	ARMA	MLP	RNN	PSN	Simple Average	Bayesian Average	GRR	LASSO	Kalman Filter
Annualized return (excluding costs)	−2.34%	7.28%	18.13%	19.44%	22.28%	19.12%	19.36%	22.37%	25.08%	34.46%
Annualized volatility	10.14%	10.44%	9.90%	9.04%	9.85%	9.09%	9.13%	9.38%	9.20%	9.32%
Information ratio (excluding costs)	−0.23	0.7	1.83	2.15	2.26	2.1	2.12	2.38	2.73	3.7
Maximum drawdown	−3.50%	−3.20%	−3.66%	−3.14%	−3.66%	−2.98%	−3.21%	−2.83%	−2.94%	−3.38%
Annualized transactions	122	90	115	117	122	111	113	97	110	114
Average leverage factor (ex post)[a]	n.a.	n.a.	1.13	1.19	1.12	1.09	1.09	1.26	1.18	1.15
Transaction and leverage costs	1.79%	1.57%	1.74%	1.75%	1.79%	1.72%	1.73%	1.62%	1.71%	1.73%
Annualized return (including costs)	−4.13%	5.71%	16.39%	17.69%	20.49%	17.40%	17.63%	20.75%	23.37%	32.73%
Information ratio (including costs)	−0.41	0.55	1.66	1.96	2.08	1.91	1.93	2.21	2.54	3.51

[a] The average leverage factor *ex post* is computed as the ratio of the annualized returns after costs of Tables 7 and 9 for those models which achieved an in-sample information ratio of at least 2 and, as such, would have been candidates for leveraging out-of-sample.

can be considered by far the optimal forecast combination for our dataset and the models under study.

8. Concluding remarks

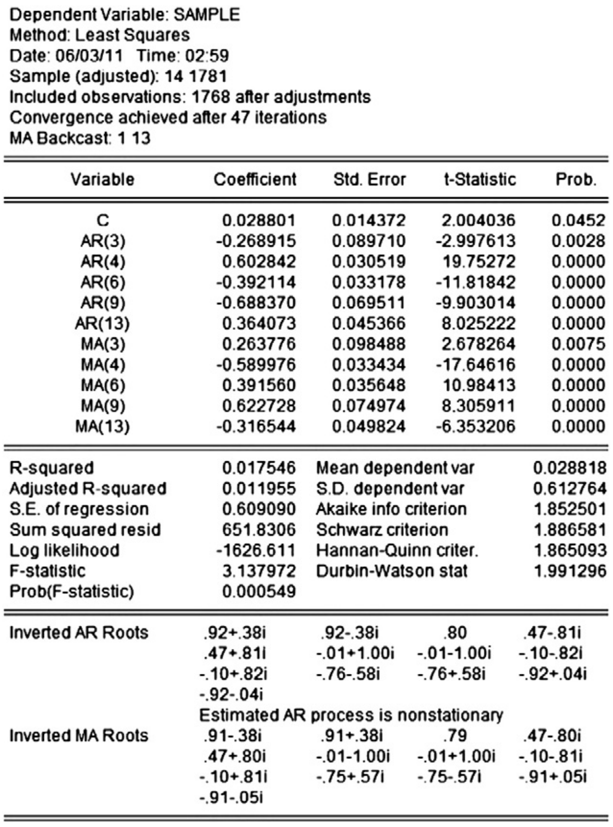
In this paper we investigate the trading and statistical performance of a Neural Network (NN) architecture, the Psi Sigma Neural Network (PSN), and explore the utility of Kalman Filters in combining NN forecasts. Firstly, we apply the EUR/USD European Central Bank (ECB) fixing series to a Naive Strategy, an Autoregressive Moving Average (ARMA) model and three NNs, namely a Multi-Layer Perceptron (MLP), a Recurrent Network (RNN) and a PSN. Secondly, we compare a Kalman filter-based combination with four other forecast combination methods. That is the traditional Simple Average, the Bayesian Average, Granger–Ramanathan's Regression Approach (GRR) and the Least Absolute Shrinkage and Selection Operator (LASSO). The models' performance is estimated through the EUR/USD ECB fixing series of the period of 2002–2010, using the last two years for out-of-sample testing. We also introduce a time-varying leverage strategy based on RiskMetrics volatility forecasts.

As it turns out, the PSN outperforms its benchmark models in terms of statistical accuracy and trading performance. It is also shown that all the forecast combinations, outperform out-of-sample all our single models except the PSN for the statistical and trading terms retained. It is interesting that the 'combining for improvement' pattern that all combination forecasts showed in the in-sample period pattern, changes regarding the out-of-sample combination forecasts. Simple Average, Bayesian Average and GRR do not continue to outperform PSNs' best individual performance but are better than MLP and RNN, while LASSO and Kalman Filter present the best results. It seems that the ability of Kalman Filter to provide efficient computational recursive means to estimate the state of our process gives it a considerable advantage compared to our fixed parameters combination models. Finally, all models except ARMA show a substantial increase in their trading performance and a striking reduction in maximum drawdowns after applying time-varying leverage with Kalman Filter still being the best approach. The remarkable trading performance of Kalman Filter allows us to conclude that it can be considered as an optimal forecast combination for the models and time-series under study.

Our results should go some way towards convincing a growing number of quantitative fund managers to experiment beyond the bounds of the more traditional models and trading strategies. The results in Table 9, with an information ratio in excess of 3, should also provide motivation for the use of Kalman Filter in combining model based forecasts.

Appendix A. The ARMA model

Fig. A.1 shows the output of the ARMA model selected. The null hypothesis that all the coefficients are not significantly different from zero is rejected at 95% confidence interval.



```
Dependent Variable: SAMPLE
Method: Least Squares
Date: 06/03/11   Time: 02:59
Sample (adjusted): 14 1781
Included observations: 1768 after adjustments
Convergence achieved after 47 iterations
MA Backcast: 1 13

      Variable      Coefficient    Std. Error    t-Statistic     Prob.

          C          0.028801      0.014372      2.004036       0.0452
        AR(3)       -0.268915      0.089710     -2.997613       0.0028
        AR(4)        0.602842      0.030519      19.75272       0.0000
        AR(6)       -0.392114      0.033178     -11.81842       0.0000
        AR(9)       -0.688370      0.069511     -9.903014       0.0000
        AR(13)       0.364073      0.045366      8.025222       0.0000
        MA(3)        0.263776      0.098488      2.678264       0.0075
        MA(4)       -0.589976      0.033434     -17.64616       0.0000
        MA(6)        0.391560      0.035648      10.98413       0.0000
        MA(9)        0.622728      0.074974      8.305911       0.0000
        MA(13)      -0.316544      0.049824     -6.353206       0.0000

R-squared             0.017546    Mean dependent var       0.028818
Adjusted R-squared    0.011955    S.D. dependent var       0.612764
S.E. of regression    0.609090    Akaike info criterion    1.852501
Sum squared resid     651.8306    Schwarz criterion        1.886581
Log likelihood       -1626.611    Hannan-Quinn criter.     1.865093
F-statistic           3.137972    Durbin-Watson stat       1.991296
Prob(F-statistic)     0.000549

Inverted AR Roots     .92+.38i      .92-.38i       .80        .47-.81i
                      .47+.81i     -.01+1.00i    -.01-1.00i   -.10-.82i
                     -.10+.82i     -.76-.58i     -.76+.58i    -.92+.04i
                     -.92-.04i
                      Estimated AR process is nonstationary
Inverted MA Roots     .91-.38i      .91+.38i       .79        .47-.80i
                      .47+.80i     -.01-1.00i    -.01+1.00i   -.10-.81i
                     -.10+.81i     -.75+.57i     -.75-.57i    -.91+.05i
                     -.91-.05i
```

Fig. A.1. The ARMA model detailed output.

Appendix B. NNs' training characteristics

In Table B.1 we present the characteristics of the Neural Networks with the best trading performance in the test sub-period which we used in our committees. The choice of these parameters is based on an extensive experimentation in the in-sample sub-period and on the relevant literature [13,22,49]. For example for the number of iterations, we started our experimentation from 10.000 iterations and we stopped at the 200.000 iterations, increasing in each experiment the number of iterations by 5.000.

Table B.1
The NNs' training characteristics.

Parameters	MLP	RNN	PSN
Learning algorithm	Gradient descent	Gradient descent	Gradient descent
Learning rate	0.001	0.001	0.5
Momentum	0.003	0.004	0.5
Iteration steps	100,000	60,000	40,000
Initialisation of weights	N(0,1)	N(0,1)	N(0,1)
Input nodes	9	9	9
Hidden nodes	7	5	4
Output node	1	1	1

Appendix C. Bayesian Information Criteria

AIC measures the relative goodness of fit of a statistical model, as introduced by Akaike [1]. On the other hand, SIC (also known as BIC or SBIC [43]) is considered a criterion to select the best model among models with different numbers of parameters. If N is the sample size of the dataset, k the total number of parameters in the equation of interest and s^2 the maximum likelihood estimate of the error variance, then AIC and BIC are calculated as shown below:

$$AIC = N\log\left(s^2\right) + 2k \quad , \quad SIC = N\log\left(s^2\right) + k\log(N) \tag{C.1}$$

Table C.1
Calculation of weights for the AIC and SIC Bayesian Averaging model.

	AIC	SIC	Δ_{AIC}	Δ_{SIC}	w_{AIC}	w_{SIC}
MLP	1.825879871	1.832039254	0.004476988	0.004476604	0.334209988	0.334210009
RNN	1.846203174	1.852362557	0.024800291	0.024799907	0.330831059	0.330831081
PSN	1.821402883	1.827562265	0	0	0.334958953	0.33495891

Table C.1 describes the estimation of the Bayesian Information Criteria for the cases of MLP, RNN and PSN forecasts, based on Eq. (13).

Appendix D. Kalman Filter and smoothing process

A generalized linear state space model of the $nx1$ vector y_t is defined as:

$$y_t = c_t + Z_t a_t + \varepsilon_t, \varepsilon_t \sim NID\left(0, \sigma_\varepsilon^2\right) \; and \; a_{t+1} = d_t + T_t a_t + n_t, n_t \sim NID\left(0, \sigma_n^2\right) \tag{D.1}$$

where α_t is a $mx1$ vector of possible state variables and c_t, Z_t, d_t and T_t are conformable vectors and matrixes.

The ε_t and n_t vectors are assumed to be serially independent, with contemporaneous variance structure:

$$\Omega_t = \operatorname{var}_t \begin{bmatrix} \varepsilon_t \\ n_t \end{bmatrix} = \begin{bmatrix} H_t & G_t \\ G_t{}' & Q_t \end{bmatrix} \tag{D.2}$$

where H_t is a nxn symmetric variance matrix, Q_t is a mxm symmetric variance matrix and G_t is a nxm matrix of covariances [56].

If now we consider the conditional distribution of the state vector α_t, given information available at time t-1, we can define with the Kalman Filter the mean and variance matrix of the conditional distribution as:

$$a_{t|t-1} = E_{t-1}(a_t) \tag{D.3}$$

$$P_{t|t-1} = E_{t-1}\left[\left(a_t - a_{t|t-1}\right)\left(a_t - a_{t|t-1}\right)'\right]. \tag{D.4}$$

Thus, the recursive algorithm of the Kalman filter calculates the following three:

1. The one-step ahead mean $\alpha_{t|t-1}$ and one-step ahead variance $P_{t|t-1}$ of the states. Under the Gaussian error assumption, $\alpha_{t|t-1}$ is the minimum mean square error estimator of α_t and $P_{t|t-1}$ is the mean square error (MSE) of $\alpha_{t|t\text{-}1}$.
2. The one-step ahead estimate of y_t as:

$$\hat{y}_t = y_{t|t-1} = E_{t-1}(y_t) = E(y_t|a_{t|t-1}) = c_t + Z_t a_{t|t-1}. \tag{D.5}$$

3. The one-step ahead prediction errors and their variances respectively as:

$$\hat{\varepsilon}_t = \varepsilon_{t|t-1} = y_t - \hat{y}_{t|t-1}, \quad \hat{F}_t = F_{t|t-1} = \mathrm{var}\left(\varepsilon_{t|t-1}\right) = Z_t P_{t|t-1} Z_t{}' + H_t. \tag{D.6}$$

In our case, we set $\hat{y}_0 = 0$ and $P_0 = 1$. If P_0 was also set equal to zero, that would mean that there is no noise, so all the estimates would be equal to the initial state. Then, the next step is to embody a smoothing algorithm to our process. The smoothing algorithm, which uses all the information observed, in other words the whole sample T, to form expectations at any period until T, is known as fixed-interval smoothing. In this way it is possible to estimate the smooth estimates of the states and the variances:

$$\hat{\alpha}_t = a_{t|T = E_T(a_t) and V_t = \mathrm{var}_T(a_t)} \tag{D.7}$$

Additionally, not only the smoothed estimates of y_t and their variances can be calculated based on Eqs. (D.5) and (D.6) respectively, but also the smoothed estimates of the ε_t and n_t vectors and their corresponding smoothed variance matrix:

$$\hat{\varepsilon}_t = \varepsilon_{t|T} = E_T(\varepsilon_t), \hat{n}_t = n_{t|T} = E_T(n_t) \ and \ \hat{\Omega}_t = \mathrm{var}_t \begin{bmatrix} \hat{\varepsilon}_t \\ \hat{n}_t \end{bmatrix} = \begin{bmatrix} \hat{H}_t & \hat{G}_t \\ \hat{G}_t{}' & \hat{Q}_t \end{bmatrix}. \tag{D.8}$$

Appendix E. The statistical and trading performance measures

The statistical and trading performance measures are calculated as shown in Tables E.1 and TE.2 respectively:

Table E.1
The statistical performance measures and their calculation description.

Performance measures	Description
Mean absolute error	$MAE = \left(\frac{1}{n}\right) \sum_{\tau=t+1}^{t+n} \lvert\hat{\sigma}_\tau - \sigma_\tau\rvert$
	with σ_τ being the actual volatility and $\hat{\sigma}_\tau$ the forecasted value
Mean absolute percentage error	$MAPE = \frac{1}{n} \sum_{\tau=t+1}^{t+n} \left\lvert\frac{\sigma_\tau - \hat{\sigma}_\tau}{\sigma_\tau}\right\rvert$
Root mean squared error	$RMSE = \sqrt{\frac{1}{n} \sum_{\tau=t+1}^{t+n} (\hat{\sigma}_\tau - \sigma_\tau)^2}$
Theil-U	$Theil-U = \sqrt{\left(\frac{\frac{1}{n}\sum_{\tau=t+1}^{t+n}(\hat{\sigma}_\tau-\sigma_\tau)^2}{\sqrt{\frac{1}{n}\sum_{\tau=t+1}^{t+n}\hat{\sigma}_\tau^2}+\sqrt{\frac{1}{n}\sum_{\tau=t+1}^{t+n}\sigma_\tau^2}}\right.}$

Table E.2
The trading performance measures and their calculation description.

Performance measures	Description
Annualized return	$R^A = 252 * \frac{1}{N} * \left(\sum_{t=1}^{N} R_t\right)$ where $R_t$the daily return
Cumulative return	$R^C = \sum_{t=1}^{N} R_t$
Annualized volatility	$\sigma^A = \sqrt{252} * \sqrt{\frac{1}{N-1} * \sum_{t=1}^{N} (R_t - \bar{R})^2}$
information ratio	$SR = \frac{R^A}{\sigma^A}$
Maximum drawdown	Maximum negative value of $\sum(R_t)$ over the period $MD = Min_{i=1,\cdots,t;t=1,\cdots,N} \left(\sum_{j=i}^{t} R_j\right)$

Appendix F. Diebold–Mariano statistic for predictive accuracy

The Diebold–Mariano [11] statistic tests the null hypothesis of equal predictive accuracy. If n is the sample size and e_i^1, e_i^2 (i = 1,2…n) are the forecast errors of the two competing forecasts, then the loss functions are estimated as:

$$L_1^{MSE}\left(e_i^1\right) = \left(e_i^1\right)^2,\ L_2^{MSE}\left(e_i^2\right) = \left(e_i^2\right)^2 \tag{F.1}$$

$$L_1^{MAE}\left(e_i^1\right) = \left|e_i^1\right|,\quad L_2^{MAE}\left(e_i^2\right) = \left|e_i^2\right|. \tag{F.2}$$

The Diebold–Mariano statistic is based on the loss differentials:

$$d_i^{MSE} = L_1^{MSE}\left(e_i^1\right) - L_2^{MSE}\left(e_i^2\right) \tag{F.3}$$

$$d_i^{MAE} = L_1^{MAE}\left(e_i^1\right) - L_2^{MAE}\left(e_i^2\right). \tag{F.4}$$

The null hypotheses tested based on the s_{MSE} and s_{MAE} are:

- $H_0 : E(d_i^{MSE}) = 0$Z against the alternative $H_1 : E(d_i^{MSE}) \neq 0$
- $H_0 : E(d_i^{MAE}) = 0$Z against the alternative $H_1 : E(d_i^{MAE}) \neq 0$.

The Diebold–Mariano test statistic s is estimated as:

$$s = \frac{\bar{d}_i}{\sqrt{\hat{V}\left(\bar{d}_i\right)}} \quad \overset{d}{\rightarrow} \quad N(0,1) \tag{F.5}$$

where

$$V\left(\bar{d}_i\right) = n^{-1}\left[\hat{\gamma}_0 + 2\sum_{k=1}^{n-1}\hat{\gamma}_k\right] and\, \gamma_k = n^{-1}\sum_{i=k+1}^{n}\left(d_i - \bar{d}_i\right)\left(d_{i-k} - \bar{d}_i\right) \tag{F.6}$$

Appendix G. RiskMetrics Volatility Model

The RiskMetrics Volatility Model is a special case of the general Exponential Weighted Moving Average Model (EWMA). The EWMA suggests that the variance of a financial asset can be calculated using the formula:

$$\sigma_t^2 = \lambda\sigma_{t-1}^2 + (1-\lambda)r_{t-1}^2 \tag{G.1}$$

where $\sigma_{t-1}^2$is the EWMA variance at time $t-1$, r_{t-1}^2 the squared returns at time $t-1$ and λ a weight between 0 and 1. The RiskMetrics Volatility Model assumes that the weight $\lambda = 0.94$. So in our case, we estimate the daily volatility with the formula below:

$$RiskMetricsVol = \sqrt{0.94\sigma_{t-1}^2 + 0.06r_{t-1}^2}. \tag{G.2}$$

References

[1] H. Akaike, A new look at the statistical model identification, IEEE Transactions on Automatic Control 19 (6) (1974) 716–723.
[2] E. Alfaro, N. García, M. Gamez, D. Elizondo, Bankruptcy forecasting: an empirical comparison of AdaBoost and neural networks, Decision Support Systems 45 (1) (2008) 110–122.
[3] G. Anandalingam, L. Chen, Linear combination of forecasts: a general Bayesian model, Journal of Forecasting 8 (3) (1989) 199–214.
[4] J.M. Bates, C.W.J. Granger, The combination of forecasts, Operational Research Society 20 (4) (1969) 451–468.
[5] L. Bertolini, Trading Foreign Exchange Carry Portfolios, PhD Thesis, Cass Business School, City University London, 2010.
[6] C. Brooks, Introductory Econometrics for Finance, second revised ed, Cambridge University Press, Cambridge, 2008.
[7] S.T. Buckland, K.P. Burnham, N.H. Augustin, Model selection: an integral part of inference, Biometrics 53 (2) (1997) 603–618.
[8] Y.L. Chan, J.H. Stock, M.W. Watson, A dynamic factor model framework for forecast combination, Spanish Economic Review 1 (2) (1999) 91–121.
[9] N.E. Coulson, R.P. Robins, Forecast combination in a dynamic setting, Journal of Forecasting 12 (1) (1993) 63–67.
[10] M. Deutsch, C.W.J. Granger, T. Teräsvirta, The combination of forecasts using changing weights, International Journal of Forecasting 10 (1) (1994) 47–57.
[11] F.X. Diebold, R.S. Mariano, Comparing predictive accuracy, Journal of Business and Economic Statistics 13 (3) (1995) 253–263.
[12] F.X. Diebold, P. Pauly, Structural change and the combination of forecasts, Journal of Forecasting 6 (1) (1987) 21–40.
[13] C.L. Dunis, Y.X. Chen, Alternative volatility models for risk management and trading: application to the EUR/USD and USD/JPY rates, Derivatives Use, Trading and Regulation, 11, 2005, pp. 126–156.
[14] C.L. Dunis, G. Giorgioni, J. Laws, J. Rudy, Statistical Arbitrage and High-Frequency Data with an Application to Eurostoxx 50 Equities, Working Paper, Liverpool Business School, 2010.
[15] C.L. Dunis, X. Huang, Forecasting and trading currency volatility: an application of recurrent neural regression and model combination, Journal of Forecasting 21 (5) (2002) 317–354.
[16] C.L. Dunis, J. Laws, G. Sermpinis, Higher order and recurrent neural architectures for trading the EUR/USD exchange rate, Quantitative Finance 11 (4) (2011) 615–629.
[17] C.L. Dunis, J. Miao, Optimal trading frequency for active asset management: evidence from technical trading rules, Journal of Asset Management 5 (5) (2005) 305–326.
[18] C.L. Dunis, J. Miao, Volatility filters for asset management: an application to managed futures, Journal of Asset Management 7 (3) (2006) 179–189.
[19] C.L. Dunis, G. Shannon, Emerging markets of South-East and Central Asia: do they still offer a diversification benefit? Journal of Asset Management 6 (3) (2005) 168–190.
[20] C.L. Dunis, M. Williams, Modelling and trading the EUR/USD exchange rate: do neural network models perform better? Derivatives Use, Trading and Regulation 8 (2002) 211–239.
[21] J.L. Elman, Finding structure in time, Cognitive Science 14 (2) (1990) 179–211.
[22] R. Ghazali, A.J. Hussain, M. Merabti, Higher order neural networks for financial time series prediction, The 10th IASTED International Conference on Artificial Intelligence and Soft Computing, Palma de Mallorca, Spain, (2006), 2006, pp. 119–124.
[23] J. Ghosh, Y. Shin, The Pi-sigma network: an efficient higher-order neural networks for pattern classification and function approximation, Proceedings of International Joint Conference of Neural Networks 1 (1991) 13–18.
[24] J. Ghosh, Y. Shin, Efficient higher-order neural networks for classification and function approximation, International Journal of Neural Systems 3 (4) (1992) 323–350.
[25] S.L. Goh, D.P. Mandic, An augmented extended Kalman Filter algorithm for complex-valued recurrent neural networks, Neural Computation 19 (4) (2007) 1039–1055.
[26] C.W.J. Granger, R. Ramanathan, Improved methods of combining forecasts, Journal of Forecasting 3 (2) (1984) 197–204.
[27] J.D. Hamilton, Time Series Analysis, Princeton University Press, Princeton, N.J., 1994
[28] A.C. Harvey, Forecasting, Structural Time Series Models and the Kalman Filter, Cambridge University Press, Cambridge, U.K., 1990
[29] T. Hastie, R. Tibshirani, J.H. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, second ed. Springer, New York, 2009.
[30] A.J. Hussain, R. Ghazali, D. Al-Jumeily, M. Merabti, Dynamic ridge polynomial neural network for financial time series prediction, IEEE International conference on Innovation in Information Technology, Dubai 2006 (2006) 1–5.
[31] H. Ince, T.B. Trafalis, A hybrid model for exchange rate prediction, Decision Support Systems 42 (2) (2006) 1054–1062.
[32] C.M. Jarque, A.K. Bera, Efficient tests for normality, homoscedasticity and serial independence of regression residuals, Economics Letters 6 (3) (1980) 255–259.
[33] I. Kaastra, M. Boyd, Designing a neural network for forecasting financial and economic time series, Neurocomputing 10 (3) (1996) 215–236.
[34] B. Krose, P.V.D. Smagt, An Introduction to Neural Networks, eighth ed, University of Amsterdam, 1996.
[35] M. Lam, Neural network techniques for financial performance prediction: integrating fundamental and technical analysis, Decision Support Systems 37 (4) (2004) 567–581.
[36] W. Leigh, R. Purvis, J.M. Ragusa, Forecasting the NYSE composite index with technical analysis, pattern recognizer, neural network, and genetic algorithm: a case study in romantic decision support, Decision Support Systems 32 (4) (2002) 361–377.
[37] J. Lesage, M. Magura, A Mixture-Model Approach to Combining Forecasts, Journal of Business & Economic Statistics 10 (4) (1992) 445–452.
[38] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, R. Winkler, The accuracy of extrapolation (time series) methods: results of a forecasting competition, Journal of Forecasting 1 (2) (1982) 111–153.
[39] P. Newbold, C.W.J. Granger, Experience with forecasting univariate time series and the combination of forecasts, Journal of the Royal Statistical Society 137 (2) (1974) 131–165.
[40] F.C. Palm, A. Zellner, To combine or not to combine? Issues of combining forecasts, Journal of Forecasting 11 (8) (1992) 687–701.
[41] R.S. Pindyck, D.L. Rubinfeld, Econometric Models and Economic Forecasts, forth ed. Irwin/McGraw-Hill, Boston, 1998.
[42] D.E. Rapach, J.K. Strauss, Forecasting US employment growth using forecast combining methods, Journal of Forecasting 27 (1) (2008) 75–93.

[43] G. Schwarz, Estimating the dimension of a model, The Annals of Statistics 6 (2) (1978) 461–464.
[44] D.N. Sessions, S. Chatterjee, The combining of forecasts using recursive techniques with non-stationary weights, Journal of Forecasting 8 (3) (1989) 239–251.
[45] A.F. Shapiro, A Hitchhiker's guide to the techniques of adaptive nonlinear models, Insurance: Mathematics and Economics 26 (2–3) (2000) 119–132.
[46] J.H. Stock, M.W. Watson, Combination forecasts of output growth in a seven-country data set, Journal of Forecasting 23 (6) (2004) 405–430.
[47] R. Sundberg, Shrinkage regression, in: A.H. El-Shaarawi, W.W. Piegorsch (Eds.), Encyclopedia of Environmetrics, John Wiley & Sons, Ltd., Chichester, 2002, pp. 1994–1998.
[48] N.R. Swanson, T. Zeng, Choosing among competing econometric forecasts: regression-based forecast combination using model selection, Journal of Forecasting 20 (6) (2001) 425–440.
[49] P. Tenti, Forecasting foreign exchange rates using recurrent neural networks, Applied Artificial Intelligence 10 (6) (1996) 567–581.
[50] N. Terui, H.K. Van Dijk, Combined forecasts from linear and nonlinear time series models, International Journal of Forecasting 18 (3) (2002) 421–438.
[51] H. Theil, Applied Economic Forecasting, North-Holland Pub. Co., Amsterdam and Rand McNally, Chicago, 1966.
[52] R. Tibshirani, Regression shrinkage and selection via the lasso: a retrospective, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73 (3) (2011) 273–282.
[53] R. Tsaih, Y. Hsu, C.C. Lai, Forecasting S&P 500 stock index futures with a hybrid AI system, Decision Support Systems 23 (2) (1998) 161–174.
[54] L. Vecci, F. Piazza, A. Uncini, Learning and approximation capabilities of adaptive spline activation function neural networks, Neural Networks 11 (2) (1998) 259–270.
[55] H. Wang, G. Li, G. Jiang, Robust regression shrinkage and consistent variable selection through the LAD–Lasso, Journal of Business and Economic Statistics 25 (3) (2007) 347–355.
[56] G. Welch, G. Bishop, An introduction to the Kalman Filter, Design 7 (1) (2001) 1–16.
[57] Y. Yang, Combining forecasting procedures: some theoretical results, Econometric Theory 20 (1) (2004) 176–222.
[58] M. Yuan, Y. Lin, On the non-negative garrotte estimator, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 69 (2) (2007) 143–161.

Dr Georgios Sermpinis is lecturer in Economics at the University of Glasgow. His research interests lie in financial forecasting and the utility of decision support systems in financial problems.

Professor Christian Dunis is Emeritus Professor of Banking and Finance at Liverpool Business School, JMU. Currently he is Joint General Manager, Horus Partners Wealth Management Group.

Dr Jason Laws is senior lecturer in Finance at the University of Liverpool. His research interests lie in financial risk management and financial derivatives.

Mr Charalampos Stasinakis is a PhD student at the University of Glasgow.