Evolution, Recurrency and Kernels in Learning to Model Inflation

By
JM Binner [1] B Jones [2] G Kendall [3] P Tino [4] J Tepper [5]

RP 0716

JM Binner [1] B Jones [2] G Kendall [3] P Tino [4] J Tepper [5]

[1] Aston University, Birmingham, UK
[2] State University of New York, Binghamton, USA
[3] University of Nottingham, Nottingham, UK
[4] University of Birmingham, Birmingham, UK
[5] Nottingham Trent University, Nottingham, UK

June 2007

ISBN No: 978-1-85449-706-2

JM Binner [1] B Jones [2] G Kendall [3] P Tino [4] J Tepper [5]

[1] Aston University, Birmingham, UK
[2] State University of New York, Binghamton, USA
[3] University of Nottingham, Nottingham, UK
[4] University of Birmingham, Birmingham, UK
[5] Nottingham Trent University, Nottingham, UK

Abstract

This paper provides the most fully comprehensive evidence to date on whether or not monetary aggregates are valuable for forecasting US inflation in the early to mid 2000s. We explore a wide range of different definitions of money, including different methods of aggregation and different collections of included monetary assets. We use non-linear, artificial intelligence techniques, namely, recurrent neural networks, evolution strategies and kernel methods in our forecasting experiment. In the experiment, these three methodologies compete to find the best fitting US inflation forecasting models and are then compared to forecasts from a naive random walk model. The best models were non-linear autoregressive models based on kernel methods. Our findings do not provide much support for the usefulness of monetary aggregates in forecasting inflation. There is evidence in the literature that evolutionary methods can be used to evolve kernels hence our future work should combine the evolutionary and kernel methods to get the benefits of both.

Keywords: Divisia, Inflation, Evolution Strategies, Recurrent Neural Networks, Kernel Methods.

1. Introduction

It is a widely held belief among macroeconomists that there is a long-run relationship between the rate of growth of the money supply and the rate of growth of prices: i.e. inflation. The objective of current monetary policy is to deliver low and stable inflation. In that regard, it is important to identify indicators of macroeconomic conditions that will alert policy makers to impending inflationary pressures sufficiently early to allow the necessary actions to be taken to control and remedy the problem. In this paper, we investigate a wide range of monetary aggregates for the United States and evaluate their usefulness as leading indicators for inflation in the early to mid 2000s. We derive inflation forecasts using sophisticated non-linear, artificial intelligence techniques including recurrent neural networks, evolution strategies and kernel methods.

Given the widely held belief in the existence of a long-run relationship between money and prices, monetary aggregates would seem to hold much promise as indicator variables for central banks. The European Central Bank (ECB), for example, employs a "two pillared" approach, which includes monetary analysis. Specifically, [1] states that "…the [President's Introductory] statement will [after identifying short to medium-term risks to price stability] proceed to monetary analysis to assess medium to long-term trends in inflation in view of the close relationship between money and prices over extended horizons."

Nevertheless, evidence to date has not tended to provide strong support for the use of monetary aggregates to forecast inflation for the United States; see, for example, Stock and Watson [2]. Moreover, as noted by the Chairman of the Federal Reserve Board Ben Bernanke, monetary aggregates have not played a central role in the formulation of US monetary policy, since 1982. He further states in [3] "why have monetary aggregates not been more influential in U.S. monetary policymaking, despite the strong theoretical presumption that money growth should be linked to growth in nominal aggregates and to inflation?". In practice, the difficulty has been that, in the United States, deregulation, financial innovation, and other factors have led to recurrent instability in the relationships between various monetary aggregates and other nominal variables." In the last two decades, other countries have also increasingly de-emphasized monetary aggregates in the conduct of monetary policy.

Recently, however, some economists have begun to issue cautionary notes on the importance of money. See, for examples, [4,5] and [6]. In particular, Carlstrom and Fuerst [7] state "…we think the current de-emphasis on the role of money may have gone too far. It is important to think seriously about the role of money and how money affects optimal policy." In a similar vein, the Governor of the Bank of England

Mervyn King stated in [8] "My own belief is that the absence of money in the standard models which economists use will cause problems in future, and that there will be profitable developments from future research into the way in which money affects risk premia and economic behaviour more generally. Money, I conjecture, will regain an important place in the conversation of economists."

A complicating factor in this debate is that money measurement is a complex problem. By considering a wide range of monetary aggregates, we attempt to assess the importance of two key issues affecting money measurement: First, what monetary assets are included in a particular aggregate? And, second, how are these assets aggregated together?

The first issue affecting money measurement is that there are a wide-range of possible monetary assets beyond currency, which may or may not be included in a particular monetary aggregate. At one end of the spectrum lies "narrow" aggregates, which include currency and various types of checking accounts. Beyond narrow money lies "zero-maturity" money, which includes, in addition to narrow money, assets that do not have a fixed maturity such as savings deposits and money market mutual fund assets. "Broader" monetary aggregates may also include assets with fixed terms to maturity such as small-denomination time deposits. Narrow aggregates include only assets that are primarily valued for their ability to facilitate transactions and, consequently, these assets earn relatively low interest rates or else they do not earn interest at all. Broader aggregates, in contrast, include a wide range of assets some of which earn competitive rates of return, such as money market mutual funds. These assets are valued both as a form of savings and for their ability to facilitate transactions (to a greater or lesser extent).

In practice, monetary aggregates have evolved over time in response to financial innovation and deregulation as well as other changes in the economic environment. For example, [3] discusses how problems with the narrow M1 monetary aggregate in the 1970s and 1980s led to increased interest by the Federal Reserve in the broader M2 monetary aggregate in the 1980s. In general, financial innovation can lead to shifts in demand between the various components of "money", which in turn can undermine earlier empirical regularities and make it more difficult to distinguish money which is held for transactions purposes from money which is held for savings purposes [9].

Recently, in the United States, two key issues have further complicated the choice between narrow and broad (and zero-maturity) aggregates. On the one hand, the growth of retail and commercial sweep programs has rendered the use of conventional narrow monetary aggregates, such as M1, inappropriate. The growth of retail sweep programs and the effects of such programs on M1 are discussed in [10] and [11] respectively. The effects of both commercial and retail sweep programs on US monetary data are discussed in detail by [12],[13], and [14]. Building on [11], Dutkowsky, Cynamon, and Jones [13] propose alternative narrow money measures that adjust the M1 aggregate for the effects of retail and commercial sweeping. They also note that the currently greater emphasis on either zero-maturity aggregates (such as M2M and MZM) or on the M2 aggregate is partially attributable to the fact that M1 has been distorted by sweeping.

On the other hand, the choice between zero-maturity aggregates and M2, which contains small time deposits, has been influenced by the so-called "missing M2" episode of the early 1990's. As documented by [15] and [16], M2 velocity began deviating from its long-run trend in the early 1990's. Duca and VanHoose [17] state that one response was to replace M2 with MZM, which was consistent with the view that it was heightened substitution between small time deposits and bond mutual fund assets, which largely accounted for the instability of velocity; see also [18]. But, [17] also go on to point out that the velocity of both M2 and MZM have encountered problems in the early 2000's. Thus, the choice between the various monetary aggregates is far from clear cut. As noted by [3], as a result of such experiences as the early 1990s, the FOMC discontinued setting target ranges for M2 after the requirement for doing so lapsed in 2000.

Beyond the problem of what assets to include within the definition of money, there lies a second key issue. Namely, how should these various monetary assets be aggregated together. Given that traditional monetary aggregates are constructed by simple summation, their use is highly questionable. This form of aggregation weights equally and linearly assets as different as cash and time and savings deposits, which requires the assumption that the included assets are all perfect substitutes in the provision of monetary services. The assumption of perfect substitutability has been refuted in numerous empirical studies; see, for example [19].

Barnett [20,21] advocates the use of Divisia monetary aggregates to measure the flow of monetary services, which do not require the assumption of perfect substitutability among the components of the aggregate. Divisia monetary aggregates build upon statistical index number theory and consumer demand theory and are, therefore, theoretically preferable to simple sum monetary aggregates. The growth rate of a Divisia monetary aggregate is a weighted sum of the growth rates of its components, where the weights depend on the expenditure shares of the assets. [21]

and [22] provide surveys of the relevant literature, whilst [23] reviews the construction of Divisia indices and associated problems.

We investigate the forecasting performance of both Divisia and simple sum monetary aggregates at various levels of aggregation ranging from narrow to broad. These include M1, M2M, M2, and M3. For comparison purposes, we also explore several interest rate variables. Our forecasting experiment builds on several recent papers including [24], [25], and [26], but is novel in several aspects.

As previously noted, Divisia monetary aggregates weight the growth rates of the component assets by their expenditure shares. In theory, the opportunity cost of a particular monetary asset is based on the spread between the rate of return on a benchmark asset, which does not provide any monetary services, and the rate of return on the monetary asset. In practice, however, a proxy must be chosen for the benchmark rate and this choice may affect the properties of the aggregate; see, for example, [27]. Drake and Mills [25, p. 153] suggest that the relatively poor performance of Divisia M2 in their study may be due to the choice of benchmark rate used by the Federal Reserve Bank of St. Louis in the construction of their Divisia indexes. A novel feature of our study is that we investigate how the choice of benchmark rate affects the forecasting performance of Divisia aggregates over a range of different levels of aggregation.

A second novel aspect of the paper lies in our use of sophisticated artificial intelligence techniques to examine the USA's recent experience of inflation. Our previous experience in inflation forecasting using state of the art approaches give us confidence to believe that significant advances in macroeconomic forecasting and policymaking are possible using techniques such as those employed in this paper. As in our earlier work, [28], results achieved using artificial intelligence techniques are compared with those using a baseline naïve predictor.

2. Data and Forecasting Model

Monthly seasonally adjusted CPI data for the US spanning the period 1960:01 to 2006:01 were used in this analysis. Inflation was constructed from CPI for each month as year on year growth rates of prices.

We employ a range of possible monetary aggregates for the US, which vary in terms of the range of monetary assets included in each aggregate and the method of aggregation. First, we explore monetary aggregates for four different groupings of monetary assets: M1, which consists of currency, demand deposits, and other checkable deposits; M2M, which consists of all M1 assets plus savings deposits including money market deposit accounts (MMDA), and non-institutional money market mutual funds; M2, which consists of M2M assets plus small-denomination time deposits; and M3, which consists of M2 assets plus institution only money market mutual funds, repurchase agreements, Eurodollar accounts, and large denomination time deposits.

Second, we explore different methods of aggregating the included monetary assets. We obtained simple sum and Divisia monetary aggregates for each of the four levels of aggregation (M1, M2M, M2, and M3) from online databases maintained by the Federal Reserve Bank of St Louis; see [29]. This provides us with eight monetary aggregates.

In addition to these publicly available Divisia indexes, we also used Divisia indexes from Elger, Jones and Nilsson [26]. The main difference between these alterative Divisia indexes and the ones maintained by the Federal Reserve Bank of St. Louis is how that benchmark rate of return is proxied. Elger, Jones, and Nilsson, [26], proxy the benchmark rate as the upper envelope of the returns in M3; *i.e.* as the maximum interest rate on all assets in M3. The St. Louis Fed's index is based on an upper envelope that also includes a long-term bond rate (the BAA rate). There are some other differences between the indexes as described by Elger, Jones, and Nilsson [26, pp. 432-3], an important one being that the alternative indexes do not include an implicit return on non-interest bearing demand deposits, but the St. Louis Fed's indexes do include an implicit return. We use these alternative Divisia indexes at all four levels of aggregation. In addition, we also constructed alternative indexes, which include the BAA rate in the upper envelope, but which in all other respects are identical to the alternative indexes from [26]. Thus, there are sixteen monetary aggregates in total.

Out of the sixteen available measures of money, we restricted the choice of monetary aggregate to just one for each of our model selections. We also experimented with the inflation forecasting potential of interest rates in our study. In that regard, we explored two different interest rate measures: a short interest rate, on three month Treasury bills, and the BAA rate. We allowed each modeller to add short and long rates alongside or instead of the chosen measures of money or to exclude interest rates. Lags of each variable and orders of differencing of each variable were permitted and left to the discretion of the individual modeller.

We investigate the evolutionary approach and evaluate it against two non-evolutionary state-of-the-art machine learning approaches to model the data. Of the 541 data points available, the first 433 were used for training, (January 1960 - February 1997), the next 50 data points were used for validation, (March 1997 –

April 2001) and the next 46 data points were used for forecasting (May 2001 – February 2005).

Individual models compete against one another with the top four being selected based on their performance on the validation set. The winning network models are subsequently evaluated individually and as an ensemble to ascertain performances across horizons τ of six months ($\tau = 6$). For each model class, we report the test set performance of the best four candidates selected on the validation set. For illustration purposes, we also plot the predictions of the best performing candidate on the test set.

3. Methodologies

In this section, we describe the four machine learning methodologies that we utilise.

3.1. Evolutionary Neural Networks

Evolutionary neural networks typically evolve both their structures and weights by creating a population of networks, which compete against each other for survival. They are unsupervised in that there does not necessarily have to be training data available. Instead, each network is evaluated to see how it performs on a certain problem. A good example is the evolution of a world class checkers player [39], where the current board position was used as an input to the networks and the networks that survived were those that were able to identify those board positions that were favourable for the automated player. This methodology has recently been applied to forecasting (see [37,38]), with some success and, it is for this reason that we utilise it here in order to compare with the other approaches proposed in this paper.

We utilise a population of neural evolutionary neural networks [30,31], evolving the weights by using evolution strategies. A weight, w, in a neural network is represented as a pair of real numbers, $v = (w,\sigma)$. w, is the current value of the weight. σ, represents a standard deviation.

Mutation is performed by replacing w by:

$$w(n+1) = w(n) + \varepsilon \qquad (1)$$

where ε is drawn from $N(0, \sigma)$, Gaussian distribution with zero mean and standard deviation of σ. This mimics the evolutionary process that small changes occur more often than larger ones. Good introductions to evolution strategies can be found in [32,33,34,35,36].

The algorithm is as follows:

1. Create a population of neural networks. In this work we have a population size of 20. The number of hidden neurons in each network is set to a random value between one and the number of input neurons. Initially, the number of input neurons is set to 4 (to reflect the time lag for inflation). We add an additional 4 neurons for the measure of money we are using (that is d1..d16). We randomly decide if we will use RTB3M (in which case the number of input neurons is increased by 4). We also randomly decide whether to use BAA (in which case the number of input neurons is increased by 4). The activation function is randomly chosen from {sigmoid, tanh}. Finally, we randomly decide if the network is a recurrent network and change the structure accordingly.
2. Each of the networks from 1, above, are evaluated, using a simple feed forward strategy, with the evaluation being given by how well it predicts inflation for each element of the times series. Each network is measured in terms of RMSE (see section 4.1).
3. The networks are sorted, based on RMSE.
4. The best 10 networks are copied to the worst 10 (effectively discarding the worst 10).
5. The 10 networks that were copied are mutated. This is done by randomly calling one of the following mutation operators:
   - 5.1. Changing the weights, m times, using equation (1). $m = 200$.
   - 5.2. Change the activation function.
   - 5.3. Add a hidden neuron.
   - 5.4. Delete a hidden neuron.
   - 5.5. Increase the number of inputs. This effectively increases the time lag we are considering for each of the data that is being used (i.e., inflation, measure of money, RTB3M and BAA).
   - 5.6. Decrease the number of inputs. This is similar to adding a neuron, except that neurons are deleted (and the time lag reduced).

   After 5.3, 5.4, 5.5 and 5.6 we execute 5.1. This is done as these four operators significantly change the network and we need to carry out some form of local optimisation, else the networks are likely to die out at the next iteration.
6. Repeat from 2 for 1,000 iterations.
7. Return the best network.

Some of our previous work has shown success with this approach (see, for example, [37,38], as well as others (e.g. [39])) but the dataset under consideration here is a lot larger and we have never compared the results against so many other methodologies as we do here.

3.2. Multi-recurrent Neural Networks

Recurrent neural networks (RNNs) are typically adaptations of the traditional feed-forward multi-layered perceptron (FF-MLP) trained with gradient-descent learning algorithms (see [40]) such as back-propagation [41]. This class of RNN extends the FF-MLP architecture to include recurrent connections that allow network activations to feedback as inputs to units within the same or preceding layer(s). Such internal memories enable the RNN to construct dynamic internal representations of temporal order and dependencies which may exist within the data. Units that receive feedback values are often referred to as *context* or *state* units. Also, assuming non-linear activation functions are used, the universal function approximation properties of FF-MLPs naturally extends to RNNs. These properties have led to wide appeal of RNNs for modelling time-series data [42,43,44,46,48].

We build upon our previous successes with RNNs [45,46] for modelling financial time-series data and extend our RNN models to a subset of RNNs referred to as *multi-recurrent networks* (MRNs) [44, 47]. The MRN architecture combines several types of feedback and delay to form a state-based model whose state transitions are modeled as an extended non-linear auto-regressive moving average (ARMA) process [43]. The architecture employs three levels of feedback allowing recurrent connections from: i) the output layer back to the input layer, as found in *Jordan networks* [48]; ii) the hidden layer back to the input layer, as found in Elman's *Simple Recurrent Networks (SRNs)* [49] and finally iii) from the context units within the input layer back to themselves (self-recurrent links). As described in Section 2, the external input vector $\mathbf{x}(t)$ consists of between one to four input variables representing combinations of previous inflation rates, one of the measures of money and zero, one or two of the interest rate measures. We do not permit recurrent or self-recurrent connections from external input units. We extend each allowable feedback by additional banks of context units (memory banks) on the input layer. The number of additional memory banks relates directly to the degree of granularity at which past and current information is integrated and stored. As per [40], we use φ memory banks where $\varphi = 4$ as past experience has shown that moving beyond this does not lead to enhanced performance. Rather, it is the number of units within each bank that is pivotal to the performance of the network and that is determined on the validation set. The MRN architecture is shown in figure 1 and can be expressed generally as:

$$\hat{y}(t+\tau) = g(f(\mathbf{c}(t), \mathbf{x}(t), \mathbf{W}_f(t)), \mathbf{W}_g(t)) \qquad (2)$$

where $\hat{y}(t+\tau)$ denotes the predicted value of inflation (the target for $\hat{y}(t+\tau)$ is $y(t+\tau)$ where τ=6 for our purposes and refers to the value of inflation 6 months ahead); $\mathbf{c}(t)$ (the context vector) is the concatenation of the: i) previous hidden state vector with four delays of varying strength and ii) summation of elements of previous output vector with four delays of varying strength; $\mathbf{x}(t)$ is the external vector of input variables; $\mathbf{W}_f(t)$ is the weight matrix connecting the input layer to the hidden layer, $\mathbf{W}_g(t)$ is the weight matrix connecting the hidden layer to the output layer and, finally, vector function *f* and function *g* return the activation vectors from the hidden and output layers, respectively. We apply the hyperbolic tangent function to the inner products performed for *f* and the binary sigmoid function to the inner products performed for *g*. We also use an *internal* output unit to provide an estimate, $\hat{\varepsilon}(t+\tau)$, for the noise term $\varepsilon(t+\tau)$ where $\varepsilon(t+\tau) = y(t+\tau) - \hat{y}(t+\tau)$.

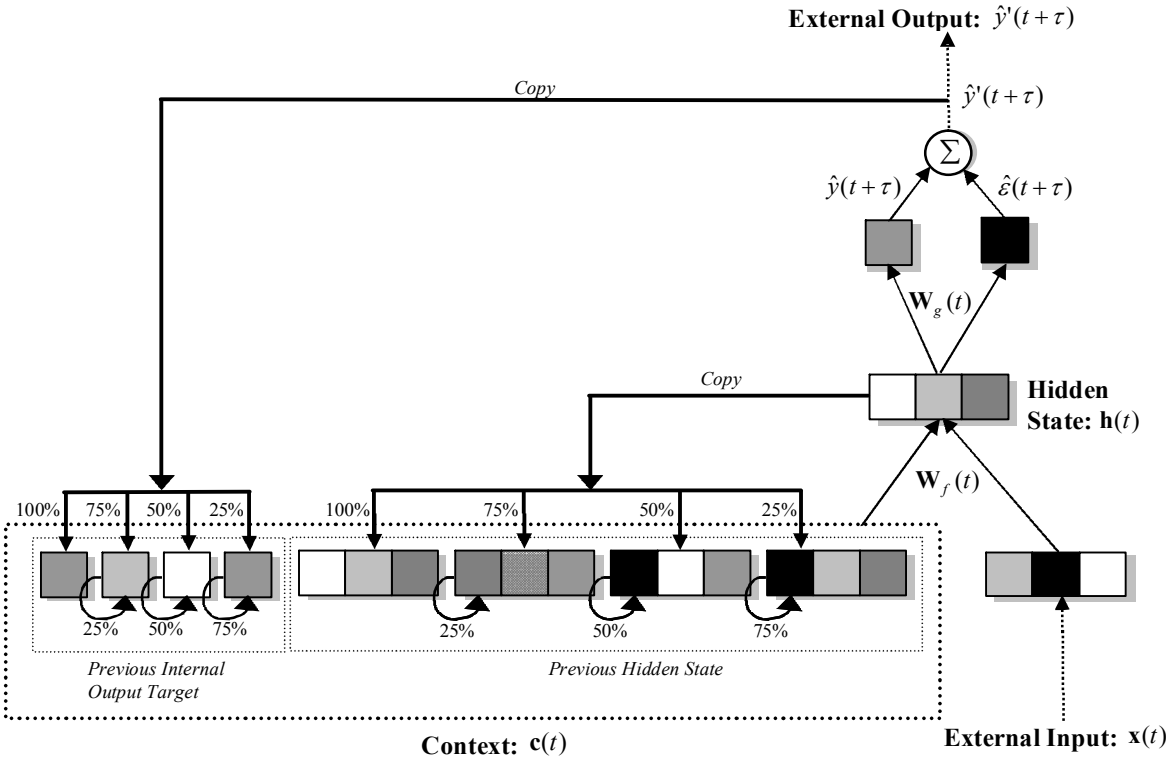

Figure 1. Architecture for the Multi-recurrent Network.

For the inner products associated with $\hat{\varepsilon}(t+\tau)$ we use the identity function for *g*.

After a number of preliminary experiments to determine the most useful information to feed back from the output layer, it was clearly evident that feeding back the target value, $y(t+\tau)$, produces superior models (with respect to performance on the validation set) compared to feeding back either the predicted forecast, $\hat{y}(t+\tau)$, or the estimated noise value, $\hat{\varepsilon}(t+\tau)$. This method, however, is unrealistic as, for the current prediction task, the target values would not be available for use during estimation. It was therefore decided to assess whether we could estimate the target, for feed back purposes, by simply summing the predicted forecast with the estimated noise value to form an *internal target*, $\hat{y}'(t+\tau)$. This consistently produced superior models (with respect to performance on the validation set) compared to those which fed back either the predicted forecast or the estimated noise term and was therefore selected as the information to feedback from the output layer. As shown in figure 1, we return $\hat{y}'(t+\tau)$ as the network's prediction value.

The context vector, $\mathbf{c}(t)$, represents an internal memory of varying rigidity – that is some context units will represent information from very recent time steps and thus change rapidly whilst others will represent information further back in time and change much more slowly. To achieve this, the unit values fed back from the hidden and output layers are combined, to varying degrees, with their respective context units, $c_i(t)$, in the input layer (as determined by the weighting values applied to their recurrent links). The influence of the previous context unit values are determined by the weighting values applied to the self-recurrent links. When the weightings on the recurrent links are greater than those on the self-recurrent links then more flexible memories are formed storing more recent information at the expense of historical information stored in the context units. Conversely, if the weighting values on the self-recurrent links are greater than those on the recurrent links then more rigid memories are formed since more historical information is preserved in the context units at the expense of the more recent information being fed back from subsequent layers. This can be generally expressed as follows:

$$c_i(t) = f(a_j(t-1)v_j + c_i(t-1)z_i) \qquad (3)$$

where $c_i(t)$ refers to context unit i at time t; $a_j(t-1)$ denotes either an internal output target value or a hidden unit activation value at time *t-1*; v_j refers to the connection strength of the recurrent link from a_j to c_i where $v_j = \frac{1}{\varphi} j$ where $j = 1,2,...,\varphi$; $c_i(t-1)$ refers to context unit i at time *t-1*; and finally z_i refers to the connection strength of the self-recurrent link for c_i where $z_i = \frac{1}{\varphi} i$ where $i = 1,2,...,\varphi$. The number of hidden units, and thus size of the recurrent layer, is a significant hyper-parameter for this model and is determined by performance on the validation set.

We use the backpropagation-through-time [50,51] an efficient gradient-descent learning algorithm for training RNNs. Input data were scaled to zero mean and unit variance. To facilitate convergent learning, the '*search and converge*' learning rate schedule as defined by [55] was used for all training experiments with an initial learning rate, η_0, of *0.0054*. This provides an effective time varying learning rate that guarantees convergence of stochastic approximation algorithms and has proven effective for temporal domains (for example see [52]). The momentum term constant is fixed at *0.95* due to the low learning rates. To identify the onset of over-fitting during training we use the decline in performance on the hold-out validation set as the basis for terminating the training process. We generate independent training sequences directly from the time-series using a time window, whose lag size is determined by performance on the validation set. The MRN context units are initialized to known values at the beginning of each sequence of data and each element within a sequence is processed sequentially by the network. Although this restricts the MRN to behaving as a finite memory model and also means that the MRN must first learn to ignore the first context values of each sequence, it allows for efficient randomized learning.

Although such RNNs and variants thereof have enjoyed some success for time-series problem domains we tread with cautious optimism. Due to the additional recurrency, such models inherently contain large degrees of freedom (weights) which may require many training cases to constrain. Also, gradient descent-based learning algorithms are notoriously difficult at finding the optimum solution the architecture is theoretically capable of representing [53,54].

3.3. Kernel-based Regression Techniques

Kernel methods (sometimes known as "kernel machines") are a popular class of learning algorithms based on a simple idea: it is possible to produce non-linear versions of traditional linear learning techniques by formulating them in a (so-called) feature space H (as opposed to working directly in the input space X where the inputs come from). Given a kernel function $K: X \times X \rightarrow R$ (satisfying Mercer conditions), there exists a Hilbert space H where K acts as a dot product. In kernel machines, the feature space associated with kernel K is such a Hilbert space H. Now, because many linear learning algorithms can be written purely in terms of dot products of inputs, it is easy to formulate their non-linear versions using the "kernel trick": simply substitute dot products of inputs by dot products of images of inputs in H, which can be calculated directly using the kernel function K. Kernel machines were shown in numerous studies to be able to achieve competitive state-of-art performances. There is one prominent problem with kernel machines though: they typically scale rather unfavorably with the number of training examples, as this is directly related to the number of degrees of freedom. Numerous approaches for achieving sparsity of kernel machines have been suggested. For more information on kernel machines we refer the interested reader to e.g. [55].

Kernel machines have shown great promise in financial forecasting [61][62]. In this paper, we use the "kernelized" version of the linear recursive least squares technique, termed *Kernel Recursive Least Squares* (KRLS) [55]. This particular approach is an example of a kernel machine with on-line constructive sparsification methodology. Loosely speaking, a training input is allowed to enter construction of the kernel machine only if its image in the feature space H cannot be "approximately" represented as a linear combination of images in H of previously admitted training inputs. This makes sense, since there is no nonlinearity in the model formulation once we transform the inputs into the feature space H. Of course, the resulting model will be non-linear, because the mapping of inputs to their images in H is nonlinear. The amount of tolerance in allowing the image in H of the currently considered input not to lie exactly in the linear subspace spanned by images in H of the previously admitted training inputs is quantified by a parameter $\nu > 0$. The greater the value of parameter, ν, the more sparse will be the resulting model. On top of this sparsification methodology, a kernel based version of recursive least squares algorithm operates, enabling to solve

efficiently in a recursive manner nonlinear least squares prediction tasks.

The input vector $\mathbf{x}(t)$ at time step t consists of lagged past inputs $(x_1, x_2, ..., x_L)$, where L is the input lag and $x_j, j = 1,2,...,L$ is the value of input variable at time $t - j + 1$. The resulting (trained) model returns for each input $\mathbf{x}(t)$ a real valued output $f(\mathbf{x}(t)) \in R$ (predicted inflation rate at time $t + \tau$) that takes the typical form of a kernel regression machine:

$$\hat{y}(t+\tau) = f(\mathbf{x}(t)) = \sum_{i=1}^{N} w_i K(\mathbf{x}^{j_i}, \mathbf{x}(t)) \quad (4)$$

where N is the number of admitted training inputs $\mathbf{x}^{j_i}$, $i = 1,2,...N$, and $K(.,.)$ is the kernel function. Given the set of training input-output pairs $(\mathbf{x}^j, y_j), j = 1,2,...,M$, due to the model sparsity, N is typically (much) smaller than the number of training inputs M. We used spherical Gaussian kernels, defined as:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\kappa^2}\right) \quad (5)$$

where $\kappa > 0$ determines the kernel width (KW) and $\|.\|^2$ is the squared Euclidean norm in the input space X. Small kernel widths κ yield more "flexible" models (that are more prone to possible overfitting of training data).

Given the training data, the collinearity threshold ν and the kernel width, κ, the KRLS returns a (potentially) sparse model that can be tested on the hold-out validation set. The most appropriate values for ν and κ are given by the best performing model on the validation set.

We also considered Kernel Partial Least Squares (KPLS) [56]. This is a kernelized version of the Partial Least Squares technique. Again, spherical Gaussian kernels were used. Kernel width as well as the number of latent factors were set on the validation set.

3.4. Naïve Model - Random walk hypothesis

Our baseline Naïve predictor operates as follows: if the prediction horizon is τ months (in our case τ=6), predict that in τ months we will observe the current inflation rate $y(t)$. In other words, we predict that the inflation rate at time $t + \tau$ will be $y(t)$, i.e. $\hat{y}(t+\tau) = y(t)$. This model corresponds to the random walk (RW) hypothesis with moves governed by a symmetrical zero-mean density function, and thus measures "the degree to which the efficient market hypothesis applies".

4. Results

Before reporting results of individual prediction techniques, we briefly describe performance measures that form the basis of our model evaluation and comparison.

4.1. Performance measures

The actual and predicted inflation rates at time t are denoted $y(t)$ and $\hat{y}(t)$ respectively. Given a sample of T test times $t_1, t_2, ..., t_T$, we calculate the root mean square error (RMSE) of the model predictions

$$RMSE = \sqrt{\frac{1}{T}\sum_{n=1}^{T}(y(t_n) - \hat{y}(t_n))^2} \quad (6)$$

However, it may be difficult to appreciate the value of RMSE for model M on its own. Therefore, we also report the rate (in percent) of improvement in RMSE of the tested model M with respect to the baseline random walk (RW) assumption:

$$IORW(M) = \frac{RMSE(RW) - RMSE(M)}{RMSE(M)} \quad (7)$$

where $RMSE(M)$ and $RMSE(RW)$ are the RMSE of the model M and RW baseline, respectively. Negative values of $IORW(M)$ indicate the case where model predictions are worse than baseline prediction capabilities.

4.2. Evolutionary Neural Networks

Table 1 and 2 show the results from the evolved neural network approach. In comparison with the results of the other two approaches (see sections 4.3 and 4.4), these results are not as competitive. However, it is noticeable that the best performing model apart from the ensemble (model 1) does not use any measure of money as input. This agrees with

the findings of both the recurrent neural network and kernel based approach. This model has few parameters, in that we can simply present it with the data and a population of networks and it evolves both the structure of the network and the inputs that it requires. Table 2 shows the structure of the networks that were evolved. All the networks were fully connected. It is noticeable that two of the networks have a large number of input neurons. This is due to the time lag being increased. Note that we did not explicitly encourage evolutionary pressure towards smaller networks.

The advantage of the methodology is that for future development there is the potential to provide even greater coverage of the search space. For example, add additional mutation operators, increase the population size, increase the time we give to the local search when we make major change to the network etc. Whilst this would require additional experimentation, this investment in time could be worthwhile.

Evol NN	Money Measure	TB Intr. Rate	BAA Intr. Rate	Past Infl. Rate
1	No	No	No	Yes
2	Simple Sum - MZM	No	No	Yes
3	Divisia MZM from St Louis Fed	No	No	Yes
4	Simple Sum M1	No	No	Yes

Table 1: Four best evolved models and input variables used by each model.

An illustration of best evolved network predictions over the test period is presented in figure 2. The solid and dashed lines correspond to the real and predicted inflation rates, respectively.

Evol NN	Input Neurons	Hidden Neurons	RMSE
1	5	6	0.0090
2	76	1	0.0134
3	12	2	0.0135
4	66	1	0.0270
Ensble	n/a	n/a	0.0082

Table 2: Number of neurons and RMSE on test set for the four best evolved models.

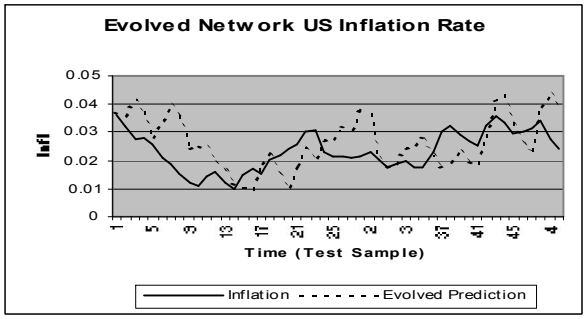

Figure 2: The best predicted inflation rates by the evolved neural network

4.3. Multi-recurrent Neural Networks

To determine the combination of allowable input variables which have the most influence on predicted inflation values, a simple selection scheme was followed. Firstly, experiments with only past inflation rates were performed. Past inflation rates are naturally used in all other input variable combinations. Secondly, experiments with one or both of the interest rate measures (TB, BAA) were performed. Finally, to determine the influence of money measures, a measure of money was included as an input variable with zero, one or both of the interest rates. For each of the above variable combinations, time-lags of 6, 12, 18, 24 and 32 months were examined. The MRN architecture and training regime described in Section 3 was used in all experiments. The number of allowable free parameters (and thus hidden nodes) was limited to a percentage of the number of training patterns.

After training the MRNs with the above variable combinations, the four best models obtained are shown in table 3 and were all achieved with a time-lag of 24 months which generated a training set of 9,864 patterns (411 sequences). MRN models were selected based on its performance on the hold-out validation set. It can be seen that in two of the best performing models, models 1 and 3, measures of money and an interest rate did appear to influence predicted inflation values. However, the level of influence is unclear as superior performance is evident without a measure of money, namely model 4, which only used both interest rates and past inflation rates.

MRN	Money Measure	TB Intr. Rate	BAA Intr. Rate	Past Infl. Rate
1	Divisia	Yes	No	Yes

	M1 (Elger, Jones & Nilsson)			
2	No	No	Yes	Yes
3	Simple Sum M3	Yes	No	Yes
4	No	Yes	Yes	Yes

Table 3: Four best MRN models and input variables used by each model.

After repeating experiments with model 4 configuration, it was clear that over the 6-month forecast horizon, superior performance is consistently obtained when using no measure of money, both interest rates and past inflation rates.

Table 4 shows the number of free parameters and resulting performance on the test set for each winning model. Model 1 required substantially more free parameters than other 'winning' models. Although this model performed best on the validation set, it performed worst on the test set. Generally, however, very good performance can be obtained on the test set with MRN models that have as low as 5% of the number of free parameters of model 1.

We also averaged predicted inflation values across all four MRN models to obtain ensemble MRN performance. The resulting RMSE for the validation set proved superior to each of the individual MRN models.

MRN	#Hid	#Pmtrs	RMSE
1	25	2,752	0.0072
2	5	147	0.0077
3	10	502	0.0070
4	10	502	0.0064
Ensble	n/a	n/a	0.0065

Table 4: Number of parameters and RMSE on test set for the four best RNN models.

An illustration of model 4 MRN predictions over the test period is presented in figure 3. The solid and dashed line correspond to the real and predicted inflation rates respectively.

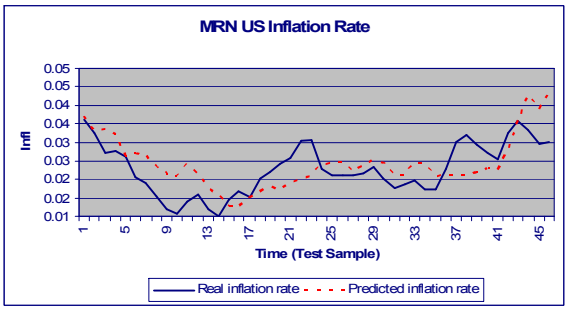

Figure 3: The best predicted inflation rates by the MRN.

4.4. Kernel machines

In general, over the 6-month forecast horizon performances of kernel machines KRLS and KPLS were comparable; hence just results for KRLS are reported. The best performing KRLS model candidates on validation set were consistently (non-linear) autoregressive models on inflation rates alone (see table 5). Input data was normalized to zero mean and unit standard deviation (the mean and variance are determined on the training set alone). As table 6 shows, the best 4 candidates from the KRLS model class used lags of 10 and 12 days. The kernel width (KW) κ varied between 1.2 and 1.5 and the collinearity threshold ν ranged between 0.21 and 0.28.

KRLS	Money Measure	TB Intr. Rate	BAA Intr. Rate	Past Infl. Rate
1	No	No	No	Yes
2	No	No	No	Yes
3	No	No	No	Yes
4	No	No	No	Yes

Table 5: Four best Kernel models and input variables used by each model.

KRLS	In Lag	KW	ν	RMSE
1	12	1.2	0.27	0.0050
2	10	1.2	0.27	0.0053
3	10	1.2	0.28	0.0057
4	10	1.5	0.21	0.0082
Ensble	n/a	n/a	n/a	0.0055

Table 6: Model structure and test set performance of the best four Kernel recursive least squares models.

The rather limited range of values of structural parameters for the KRLS class indicates that we have found a stable regime within which the KRLS class

yields a satisfying performance on the given problem. It seems that having historical monthly inflation rates roughly a year into the past and a spherical kernel of width slightly larger than standard deviation of input data, with modest model sparsity induced by $\nu \cong 0.27$ yields the best performances. An illustration of KRLS predictions over the test period is presented in figure 4. The solid and dashed lines correspond to the real and predicted inflation rates, respectively.

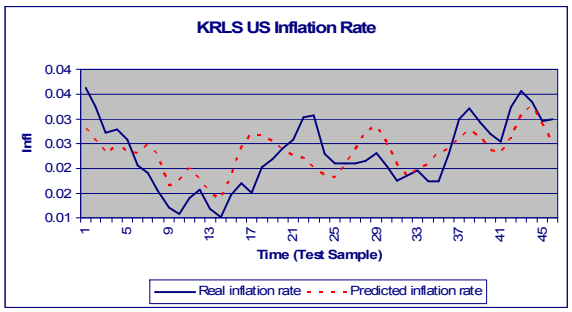

Figure 4: The best predicted inflation rates by the KRLS.

4.5. Model Comparison

A summary of results appears below in Table 7. This shows the best fitting model from each of the above approaches. The kernel based regression provides the biggest improvement over the random walk of 43.4% with the recurrent neural network in second place.

	Evol NN	MRN	KRLS
Model	Ensemble	4	1
RMSE	0.0082	0.0064	0.0050
IORW	N/A	27.5	43.4
Money?	No	No	No

Table 7: Results summary

Our early indications suggest that the kernel method appears to implement a more efficient feature extraction method that is well suited to the problem. The recursive least squares can be viewed as a complexity controlling mechanism, such that you only include things that are completely necessary. Also, the width of the Kernel and input lag have to be set appropriately. Everything is compact and the minimum model needed does the modeling. It appears that using the kernelised version of the linear least squares technique provides an effective mechanism for constructing efficient feature domains well suited to the task of prediction. One key difference between the three techniques employed in this paper is the treatment of the input variables. The kernel based regression and the evolved neural networks treat the inputs as inherently static, whilst the recurrent neural networks treat the inputs as dynamic due to the internal state memory being formed and fed back at each time step.

Since the validation and test sets are only samples, the performance of the best model on the test set may overestimate the typical performance one would expect from the corresponding model class. The average performance of the few best performing models, however, gives an indication of the expected performance, if sound model selection techniques are used. With the exception of the evolved neural network model class, the average IORW measures of the best four models were 19.9% and 29.7% for the MRN and KRLS model classes respectively. This gives a strong positive indication of the predictive capacity of the model classes considered in this inflation prediction task.

With respect to the question 'does money matter?' simple sum money and a Divisia money constructed using the envelope approach are among the best performing models in the MRN model class. Nevertheless, the best model in that class did not contain any money but did contain interest rates. The best models overall are the nonlinear autoregressive models based on kernel methods. It is felt that further improvements in the construction of the money supply might help to obtain further improvements in the inflation forecasting performance of the monetary aggregates.

5. Conclusions and Future Work

Our results corroborate the recent findings of [26] that little evidence is found that either aggregation method or level of aggregation has a significant impact on the performance of our model, in contrast to the results reached by [24]. To the contrary, our results lend more support to the conclusion that the use of monetary aggregates lead, at most, to marginal improvements in forecasting inflation and are, therefore, of limited value in that context. With continual innovation in financial markets, the impact on the measurement of monetary aggregates will continue to present problems in the form of monetary control. Aggregates that have been adjusted to take account of financial innovations and the riskiness of the assets, along the lines proposed recently by [57], have the potential to make a valuable contribution to the future development of monetary policy and this issue is likely to be a fruitful area for future research.

How should we interpret our finding that monetary aggregates are not helpful in forecasting US inflation? There are several possible explanations for these findings. As Stock and Watson point out [58], beginning in the mid-1980s inflation has become more difficult to forecast in the sense that it has become more difficult for an inflation forecaster to provide value added relative to a univariate model. This is not unique to monetary aggregates, but is a more general phenomenon; see, for example, [59]. Another explanation, more specific to money, is that monetary aggregates may be less informative when inflation is low and stable than during periods of high and volatile inflation. The reason is that one can think of money demand shocks as noise that obscures the signal from the monetary aggregates. Thus, in periods when inflation and money growth are both subdued, the signal to noise ratio of the monetary aggregates is likely to be low; see [60]."

The "market is efficient" in the sense that it is difficult to beat the Naïve RW model. The IORW measure is best suited for our purposes, because rather than worrying about the precise RMSE values, we should be concerned about by how much we can beat the rather obvious RW strategy. In our experiments (not reported here) we found that the longer the prediction horizon, the harder it is to accurately predict the inflation rate. It seems that inclusion of historical inflation rates alone works quite well for the KRLS model class. We may speculate that the value of money is implicitly represented in the inflation rates and thus their explicit inclusion as input variables does not help, rather, it can actually make things worse because of the under sampling problems.

The validation set is just a sample therefore we cannot rely solely on picking the best model on the validation set. Slightly worse models on the validation set may be slightly better on the test set. Therefore we constructed flat ensembles of several best performing models on the validation set. In one case the performance of the ensemble on the test set was indeed better than that of the single winner picked on the validation set.

We also noted an interesting fact that the IORW measures did not change drastically with increasing prediction horizon. That indicates that for longer prediction horizons, it is more difficult to predict the inflation rates, but the naive model is less suited as well, so things cancel out.

The class of recurrent neural networks used for this problem domain appeared to have offered some advantages over the evolved FFMLPs with time windows and simple recurrent networks. Although the additional feedback connections used increases the complexity of the model, the improvement in performance appears to outweigh the negative effect of increased number of parameters. Also, the success of the MRNs is largely dependent on the hyper-parameters used, most of which have to be empirically established. There is no universal rule to determine the level and nature of recurrent and self-recurrent connections. A key problem with the approach taken is the use of gradient-descent based learning algorithms. The problems with back-propagation through time, for example, are well documented [53,61]. Bengio et al. [53] in particular has shown that the error gradient vanishes exponentially for dependent observations that are separated by intervening non-related observations, where greater number of intervening observations yields poorer generalization performance. Although the MRN offers modifications to the architecture which may alleviate some of the problems posed by gradient-descent learning methods (e.g. through the formation of sluggish state-spaces), we will also investigate other learning algorithms that are better able to find solutions the architecture is theoretically capable of representing [49]. In addition to investigating other learning algorithms, it is envisaged that improvements in MRN performance could also be obtained as follows:

- Train the MRN linearly on the time-series without resetting the context layer to reduce existing limitations of finite memory MRNs and to eliminate the need for such nets to learn to ignore reset values;
- Balance bias and variance more effectively by using weight regularisation techniques;
- Investigate and evaluate further standardization and scaling techniques to better facilitate non-linear estimation of the data.

In the case of the kernel based regression method, further investigation is required to understand and improve the stability of the model with respect to the addition of further input variables (e.g. such as measures of money). It may be that other compound measures of money may be more useful however this is likely to be model dependent. Further investigation into controlling model complexity (e.g. such as determining the number of kernels) may additionally provide further improvement and insight into the efficacy of this approach to inflation forecasting. This will require more sophisticated versions of cross-validation than the one employed in this paper.

Future work will need to address the problem of why exactly do the kernel models break down when measures of money are added. It seems that this issue is related to overfitting. When we tried using money alone, we could (after some work) get an excellent fit on the training set, but of course the

models were inferior on the validation (and test) set. In other words, the learnt relationships by KRLS between past measures of money and future inflation were not representing anything real.

We noted that, using KRLS, a good fit on the training set was obtained more naturally and easily when only past inflation rates were considered. This would suggest that, conditional on the kernel machine class used,, there is little mutual information between past bulk measures of money and future inflation. Of course, future work will need to validate or refute this hypothesis.

Finally, there is huge scope for deeper involvement of evolutionary methods in our future work. In general, it may be worthwhile to combine the evolutionary side with machine learning methods to get the benefits of both. For example, evolutionary methods can be used to evolve specialized architectures of neural networks. Also, there is evidence in the literature that evolutionary methods can be used to evolve kernels and support vector machines [62,63].

References

[1] European Central Bank (2003). The outcome of the ECB's evaluation of its monetary policy strategy. *ECB Monthly Bulletin*, June, 87-102.

[2] Stock, J., and Watson, M. (1999). Forecasting inflation. *Journal of Monetary Economics*, 44, 293–335.

[3] Bernanke, B. (2006**).** Monetary Aggregates and Monetary Policy at the Federal Reserve: A Historical Perspective: Remarks at the Fourth ECB Central Banking Conference, Frankfurt, Germany, November 10, 2006, *Federal Reserve Board* http://www.federalreserve.gov/boarddocs/speeches/2006/20061110/default.htm.

[4] Nelson, E. (2002) "Direct Effects of Base Money on Aggregate Demand: Theory and Evidence*" Journal of Monetary Economics*, 49(4), 687–708.

[5] Nelson, E. (2003) "The Future of Monetary Aggregates in Monetary Policy Analysis." *Journal of Monetary Economics*, 50(5), 1029–59.

[6] Leeper, E. and J. Rouch (2003) "Putting 'M' back in Monetary Policy" *Journal of Money, Credit and Banking* 35, 1217-1256.

[7] Carlstrom, C. and T. Fuerst (2004) "Thinking about Monetary Policy without Money" *International Finance* 7:2, 325-347.

[8] King, M. (2002). No money, no inflation – the role of money in the economy. *Bank of England Quarterly Bulletin*, Summer, 162-177.

[9] Mullineux A.W. (1996). Financial Innovation, Banking and Monetary Aggregates, Chapter 1, pp. 1–12. Cheltenham, UK: (Eds) Edward Elgar.

[10] Anderson, R., & Rasche, R. (2001, January/February). Retail sweep programs and bank reserves: 1994–1999. *Federal Reserve Bank of St. Louis Review*, *83*, 51–72.

[11] Dutkowsky, D., & Cynamon, B. (2003). Sweep programs: The fall of M1 and the rebirth of the medium of exchange. *Journal of Money, Credit, and Banking*, *35*, 263–280.

[12] Jones, B., Dutkowsky, D., & Elger, T. (2005). Sweep programs and optimal monetary aggregation. *Journal of Banking and Finance*, *29*, 483–508.

[13] Dutkowsky, D., Cynamon, B., & Jones, B. (2006). US narrow money for the twenty-first century. *Economic Inquiry*, *44*, 142–152.

[14] Cynamon, B. Z., D. H., and B. E. Jones (2006). Redefining the Monetary Aggregates: A Clean Sweep. *Eastern Economic Journal*, 32, 661-672.

[15] Duca, J. V. "Financial Technology Shocks and the Case of the Missing M2." *Journal of Money, Credit,and Banking*, 32(4), 2000, 820–39.

[16] Hafer, R. W., and D. C. Wheelock. ''The Rise and Fall of a Policy Rule: Monetarism at the St. Louis Fed, 1968–1986. '' Federal Reserve Bank of St. Louis Review, 83(1), 2001, 1–24.

[17] Duca, J., & VanHoose, D. (2004). Recent developments in understanding the demand for money. *Journal of Economics and Business*, *56*, 247–272.

[18] Carlson, J., Hoffman, D., Keen, B., & Rasche, R. (2000). Results of a study of the stability of cointegrating relations comprised of broad monetary aggregates. *Journal of Monetary Economics*, *46*, 345–383.

[19] Fisher, D. and A. F. Fleissig (1997) "Monetary Aggregation and the Demand for Assets", *Journal of Money, Credit, and Banking*, 29, 458-475.

[20] W.A. Barnett, (1980) "Economic monetary aggregates; an application of index number and aggregation theory". *Journal of Econometrics* 14(1), 11–48, Reprinted in W.A. Barnett and A. Serletis (Eds.) The Theory of Monetary Aggregation, North-Holland, Amsterdam, Ch. 2, 2000, 11–48.

[21] W.A. Barnett, D. Fisher, and A. Serletis, (1992) "Consumer theory and the demand for money", *Journal of Economic Literature,* 30, 2086-119. Reprinted in The Theory of Monetary Aggregation (Ed.) W.A. Barnett and A. Serletis, North Holland, Amsterdam, pp. 389-427.

[22] Anderson, R., Jones, B. and Nesmith, T. (1997). Monetary Aggregation Theory and Statistical Index Numbers. Federal Reserve Bank of St. Louis Review, 79, 31-51.
[23] L. Drake, A.W. Mullineux, and J. Agung (1997) "One Divisia money for Europe". *Applied Economics* 29: 775-786.
[24] Schunk, D. (2001). The relative forecasting performance of the Divisia and simple sum monetary aggregates. *Journal of Money, Credit, and Banking*, *33*, 272–283.
[25] Drake, L., & Mills, T. (2005). A new empirically weighted monetary aggregate for the United States. *Economic Inquiry*, *43*, 138-157.
[26] Elger, T., B. Jones, and B. Nilsson (2006) "Forecasting with Monetary Aggregates: Recent Evidence for the United States" *Journal of Business and Economics* 15, 428-446.
[27] Hancock, M. (2005). Divisia Money. *Bank of England Quarterly Bulletin*, Spring, 39-46.
[28] Binner, J.M., Kendall, G. and Gazely, A.M. (2005) "Evaluating the performance of a EuroDivisia index using artificial intelligence techniques", 4th International Conference on Computational Intelligence in Economics and Finance, 871-874.
[29] Anderson, R., Jones, B. and Nesmith, T. (1997). Building new monetary services indexes: Concepts, data, and methods. Federal Reserve Bank of St. Louis Review, 79, 53–82.
[30] Yao. Y. (1999). Evolving artificial neural networks. Proceedings of the IEEE, 87(9):1423-1447
[31] Stanley, K. and Miikkulainen, R. (2002). Evolution of Neural Networks through Augmenting Topologies. Evolutionary Computation, 10:99–127
[32] Bäck, T., Fogel, D. B. and Michalewicz. (1997). Handbook of evolutionary computation. Oxford University Press. ISBN 0 7503 0392 1
[33] Fogel, D.B. (1998) Evolutionary computation the fossil record, IEEE Press, ISBN 0-7803-3481-7
[34] Fogel, D.B. (2000) Evolutionary computation: toward a new philosophy of machine intelligence, 2nd Ed., IEEE Press Marketing, ISBN 0-7803-5379-X
[35] Michalewicz, Z. (1996). Genetic algorithms + data structures = evolution programs (3rd rev. and extended ed.). Springer-Verlag, Berlin
[36] Michalewicz, Z and Fogel, D.B. (2000). How to solve it. Springer-Verlag. ISBN 3-540-66061-5
[37] Binner J.M., Kendall G. and Gazely A.M. (2004). Evolving Neural Networks with Evolutionary Strategies: A New Application to Divisia Money. Advances in Econometrics, 19:127-143.
[38] Binner J.M. and Kendall G., (2002). Co-Evolving Neural Networks with Evolutionary Strategies : A New Application to Divisia Money. Proc International Conference on Artificial Intelligence (IC-AI'02), CSREA Press. Arabnia H.R and Youngsong M. (eds), pp 884-889.
[39] Fogel D.B. (2002). Blondie24: Playing at the Edge of AI. Morgan Kaufmann.
[40] Pearlmutter, B. A. (1995): Gradient calculations for dynamic recurrent neural networks: a survey. IEEE Transactions on Neural Networks, 6(5), 1212-1228.
[41] Rumelhart D., Hinton G. and Williams R. (1986). Learning Internal Representations by Error Propagation, in: Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol. 1. Princeton University Press, pp. 318-362.
[42] Moshiri S., Cameron N. and Scuse D. (1999). Static, dynamic and hybrid ANN models in forecasting inflation. Computational Economics, 14, 219-235.
[43] Tenti P. (1996). Forecasting foreign exchange rates using recurrent neural networks. Applied Artificial Intelligence 10, 567-581.
[44] Ulbricht C. (1994). Multi-Recurrent Networks for Traffic Forecasting. Proceedings of the Twelfth National Conference in Artificial Intelligence, AAAI Press/MIT Press, Cambridge, MA 883-888.
[45] Binner J.M., Elger T., Nilsson B. and Tepper, J.A. (2006). Predictable non-linearities in U.S. inflation. Economic Letters, Vol. (93), [3], 323 – 328.
[46] Binner J.M., Elger T., Nilsson B. and Tepper J. (2004). Tools for non-linear time series forecasting in Economics: an empirical comparison of regime switching vector autoregressive models and recurrent neural networks. Advances in Econometrics: Applications of Artificial Intelligence in Finance and Economics, 19: 71-91, Elsevier Science Ltd.
[47] Dorffner, G. (1996). Neural networks for time series processing. Neural Network World, 6(4):447–468, 1996
[48] Jordan M. (1986). Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. Proc of the 8th Annual Conference of the Cognitive Science Society, 531-545.
[49] Elman J. (1990). Finding Structure in Time. Cognitive Science 14, 179-211.

[50] Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. In Proceedings of IEEE Special Issue on Neural Networks, vol 78, pp. 1550-1560.
[51] Williams, R. J. and Peng J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. Neural Computation, 2, 490-501.
[52] Lawrence, S., C. L. Giles, and S. Fong (2000): Natural language grammatical inference with recurrent neural networks. IEEE Transactions on Knowledge and Data Engineering, 12(1), 126-140.
[53] Bengio, Y., Simard P., and Frasconi P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5, 1550-1560.
[54] Sharkey, N., Sharkey, A. and Jackson, S. (2000). Are SRN's sufficient for modelling language acquisition? In: P. Broeder and J. Murre, Models of Language Acquisition: Inductive and Deductive Approaches. Oxford University Press. p. 33-54.
[55] Engel Y., Mannor S. and Meir R. (2004). The Kernel Recursive Least-Squares Algorithm. IEEE Transactions on Signal Processing, 52, (8) 2275-2285.
[56] Rosipal R. and Trejo L.J. (2001). Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space. Journal of Machine Learning Research, 2, 97-123.
[57] Barnett, W.A. and S. Wu (2005), "On User Costs Of Risky Monetary Assets", *Annals of Finance,* 1(1), pp. 35-50.
[58] Stock, J. H., and M. W. Watson (2007). Why has U.S. Inflation Become Harder to Forecast? *Journal of Money, Credit, and Banking*, 39, 3-33.
[59] Atkeson, A. and L. E. Ohanian. (2001). Are Phillips Curves Useful for Forecasting Inflation? Federal Reserve Bank of Minneapolis Quarterly Review, 25:1, 2-11.
[60] Estrella, A., & Mishkin, F. (1997). Is there a role for monetary aggregates in the conduct of monetary policy? *Journal of Monetary Economics*, 40, 279-304.
[61] Haykin, S. (1999) Neural Networks: A Comprehensive Foundation. 2nd edition, Prentice Hall, Upper Saddle River, NJ.
[62] Igel, C., Wiegand, S., and Friedrichs, F. (2005). Evolutionary optimization of neural systems: the use of strategy adaptation. Trends and Applications in Constructive Approximation (M. G. de Bruin, D. H. Macho and J. Szabados, eds), International Series of Numerical Mathematics 151:103-123.
[63] Howley, T., and Madden, M. G. (2004). The genetic evolution of kernels for support vector machine classifiers. In Proceedings of 15th Irish Conference on Artificial Intelligence and Cognitive Science, September 2004