WORKING PAPERS

N° 1666

May 2025

"Affine Feedforward Stochastic (AFS) Neural Network"

Christian Gouriéroux and Alain Monfort

Toulouse School of Economics

Affine Feedforward Stochastic (AFS) Neural Network

Gourieroux, C.,[(1)], and A., Monfort [(2)]

May, 2025

We thank E., Sentana for useful remarks.

Acknowledgment : Gourieroux gratefully acknowledge financial support of the ACPR Chair : "Regulation and Systemic Risks", the ERC DYSMOIA, and the ANR Project "Form Machine Learning to Structural Econometrics with Discrete Variables".

[1]University of Toronto, Toulouse School of Economics and CREST.
[2]CREST.

Affine Feedforward Stochastic (AFS) Neural Network

Abstract

The aim of this paper is to link the machine learning method of multilayer perceptron (MLP) neural network with the classical analysis of stochastic state space models. We consider a special class of state space models with multiple layers based on affine conditional Laplace transforms. This new class of Affine Feedforward Stochastic (AFS) neural network provides closed form recursive formulas for recursive filtering of the state variables of different layers. This approach is suitable for online inference by stochastic gradient ascent optimization and for recursive computation of scores such as backpropagation. The approach is extended to recurrent neural networks and identification issues are discussed.

Keywords : Stochastic Neural Network, Perceptron, State Space Model, Laplace Transform, Nonlinear Prediction, Curse of Dimensionality, Deep Learning, Stochastic Gradient Descent, Indirect Inference.

1 Introduction

Over the last two decades statistics of the algorithmic modelling, i.e. the machine learning has advanced significantly, especially for prediction problems [see Breiman (2001) and the comments by Cox and Efron for the debate between the algorithmic culture and the approaches based on stochastic models]. The algorithmic methods have been successful in recommendation systems, speech recognition, or image analysis, but appeared less efficient and robust in other fields such as credit scoring, or nonlinear time series. One of their weaknesses is the lack of interpretation, that is the black-box effect. Typically, the data are seen as "generated by a black-box in which a vector of input variables X go in one side and on the other side the response variables Y come out" [Breiman (2001), p 199]. In practice, this black-box is replaced by a prediction algorithm, which often is not sufficiently transparent.
The aim of this paper is to introduce a stochastic structure replicating the standard scheme of Figure 1.

Figure 1 : Standard Scheme

$$X \to [BlackBox] \to Y$$

We will focus on black-boxes with feedforward stochastic network structures. Then we will distinguish three components:
i) The entry (input) process X that describes how the input enters the black-box (system).
ii) The network structure within the black-box. It has J (hidden) layers and for each layer j registers the impact of X on several components of a vector Z_j, sometimes called the features. The number of components (or nodes, neurons, modulus) can depend on the layer.
iii) The exit (output) process Y that describes how the output depends on Z_J.

The input X and output Y can differ from the observed underlying variables X^*, Y^* with economic, physical, or structural interpretations.[3] In practice they are defined from X^*, Y^* by applying some known nonlinear transformations. For instance, we could have $Y = [Y^*, (Y^*)^2]'$ and $X = [X^*, X^* \mathbb{1}_{X^*>0}]'$.
We focus our attention on the strict feedforward scheme (see Figure 2).

[3] In some applications the input-output are called sensor-receptor.

Figure 2 : Strict Feedforward Scheme

$$\underbrace{X \to}_{\text{entry process}} \quad \underbrace{Z_1 \to Z_2 \to}_{\text{network}} \quad \ldots \underbrace{\to Z_J \to Y}_{\text{exit process}}$$

The variables X, Z, Y and the transitions are potentially random. This scheme has to be distinguished from the standard algorithmic scheme often used in practice where both the variables and transitions are deterministic, as if the variables were replaced by their pointwise predictions $\hat{Z}_j's$ and $\hat{Y}$, say (see Figure 3).

Figure 3 : Predictions Feedforward Scheme

$$\underbrace{X \to}\underbrace{\hat{Z}_1 \to \hat{Z}_2 \to \ldots \to}\underbrace{\hat{Z}_J \to \hat{Y}}.$$

In the classical probabilistic models, the variables X, Z_j, Y are random and the assumption of strict feedforward scheme implies that the sequence of $Z_j's$ satisfies the Markov property. To contrast, under machine learning the $\hat{Z}_j's$ and $\hat{Y}$ are deterministic functions of X. In particular, this scheme assumes a flexible functional form $\hat{Y}(X)$ for the pointwise prediction of X. This interpretation is often given when such machine learning approaches are applied [see Kuan and White (1994) and also Lee et al. (2017) for an attempt to manage jointly deterministic and stochastic neural networks].

Example 1 : The Multilayer Perceptron (MLP) Neural Network

The MLP network introduced in Rosenblatt (1957), (1958) is a typical example of feedforward scheme for prediction [see Haykin (1998), Yu and Deng (2015), Chapter 4 for general presentations].[4] In general the variable X and the predictions are valued in different domains. The vector X has dimension p and its univariate components take real values. The components of the $\hat{Z}_j, j = 1, \ldots, J$, take values in [0,1], and $\hat{Y}$ has real value. Then a deterministic chain is, for instance, a system with two layers and two nodes (neurons) per layer :

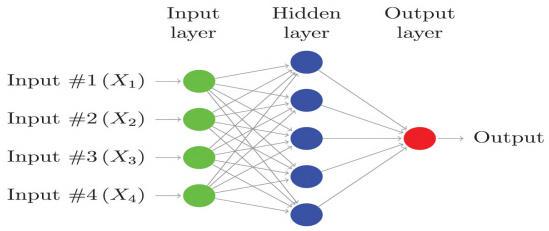
[4]"Although inspired by the way information is processed in the brain, it is far from a realistic description of how brains actually work" [Kuan and White (1984)]. Nevertheless this explains terminologies as neuron, neural, synoptic appeared in the machine learning literature and the introduction of the word "artificial" in Artificial Neural Network (ANN) or Artificial Intelligence (AI).

$$
\begin{aligned}
\text{entry process}: \quad \hat{Z}_{11} &= G(c'_{10}X + d_{10}), \hat{Z}_{12} = G(c'_{20}X + d_{20}), \\
\text{network}: \quad \hat{Z}_{21} &= G(c'_{11}\hat{Z}_1 + d_{11}), \hat{Z}_{22} = G(c'_{21}\hat{Z}_1 + d_{21}), \text{with } \hat{Z}_1 = (\hat{Z}_{11}, \hat{Z}_{12}), \\
\text{exit process}: \quad \hat{Y} &= \gamma'\hat{Z}_2 + \delta, \text{ and } \hat{Z}_2 = (\hat{Z}_{21}, \hat{Z}_{22})',
\end{aligned}
$$

where $c_{10}, c_{20}, \ldots, \delta$ are the parameters, called synoptic weights (or connection strengths) for the c parameters and biases for the d parameters with the appropriate dimensions, and G is often the logistic function $G(y) = 1/[1 + \exp(-y)]$.[5] When the components of X are explicitly mentioned, the scheme is detailed in Figure 4 for $p = 4$.

[5]Other activation functions G have also been suggested, as the Threshold Logic Unit (TLU) : $G(y) = \mathbb{1}_{y>0}$, or the popular Rectified Linear Unit (RELU) : $G(y) = \max(0, y)$ in the machine learning jargon.

Figure 4 : The MLP Network



The transitions of the feedforward scheme in Figures 3 and 4 are usually parametric and deterministic. The only implicitly introduced probabilistic assumption concerns the pair of variables (X, Y) assumed random. The single prediction of interest is $\hat{Y} = E(Y|X)$. In particular the approach does not explain how to predict in a coherent way transformations of Y as Y, Y^2, or $|Y|$, and how to account for the monotonicity or convexity restrictions between the pointwise predictions of such nonlinear transforms. [6].

Example 1 (follows) : Identification issues

[6] as needed in financial applications when Y is a return

The use of the MLPNN is often justified by mentioning the superposition theorems that state that every continuous function of p continuous variables can be represented by an appropriate composition of functions of one variable (see online Appendix 1 on the history of superposition theorems). Then, for continuous inputs X, the MLPNN is presented as a nonparametric estimation method of the nonlinear pointwise prediction $E(Y|X)$. This approach is reversing the results of the representation theorems : the function of interest, i.e. the prediction, is now unknown, and is estimated by means of the estimation of a representation. This implies two different identification issues :
i) A (strict) identification issues, since the representation is not unique in general (see online appendix 1).
ii) A weak identification issue by the curse of dimensionality (CoD) of any nonparametric estimator that is supposed to capture all cross effects of the components of X. Practically, with at most thousands of observations available in the main frequent applications in Economic and Finance[7], we cannot expect to get accurate results with more than three continuous conditioning variables. In fact in the MLPNN what really matters is the number L_1 of neurons in the first layer that summarize the p continuous input variables. Then, we get the three following situations :

i) If $L_1 > p$, we have a (strict) identification issue.
ii) If $3 < L_1 \leq p$, we encounter the CoD and a weak identification issue.
iii) If $L_1 \leq 3 < p$, we have no identification issue, but we do not have a pure nonparametric approximation, but an approximation based on a constrained nonparametric model with L_1 linear indexes. [see Geenens (2011), Conn and Li (2019) for the discussion of the curse of dimensionality in nonparametric inference][8].

The strict feedforward stochastic neural networks (SNN) in Figure 4 is also considered in probabilistic state-space models under an alternative terminology. Layer is a terminology common to network and state-space models. The nodes (neurons) components of Z_j are the state variables, the transitions from Z_j to Z_{j+1} are the transition equations, the exit (output) process is the measurement equation. Each transition, for instance from Z_j to Z_{j+1},

[7]This can correspond to the number of firms in an industrial sector or to daily observations on asset return over 20 years.

[8]These remarks are also valid for other neural networks as the Kolmogorov-Arnold Networks (KAN) recently introduced [see Liu et al. (2024)]

is characterized by the conditional distribution of Z_{j+1} given Z_j (instead of being defined by the conditional expectation only).

The aim of our paper is fourfold :
i) Render compatible the two cultures, that are the probabilistic state-space model of Figure 2 and the algorithmic model that automatically updates the predictions in Figure 3[9].
ii) Introduce flexible families of stochastic networks that do not implicitly assume that the Z_j variables are binary, i.e. valued in $\{0, 1\}$, with their predictions $\hat{Z}_j$ valued in $[0, 1]$.
iii) Select appropriate learning criteria develop the statistical inference in order to estimate the predictive distribution, measure the errors in nonlinear pointwise predictions, filter out the latent features, or analyse the nonlinear Impulse Response Functions (IRF).
iv) Discuss the interpretation of algorithmic online estimation under strong or weak identification issues.

There already exists a literature trying to relate the two cultures, mainly concerned about the nonparametric prediction [see Gallant (1981), Gallant and White (1981), Athey and Imbens (2019)]. See, however, Example 1 (follows) on identification issues concerning this literature. The comparison of the MLP neural network models and the stochastic state space models is less discussed and considered with specific applications in mind [Yu and Deng (2015) for speech recognition, Twumasi and Twumasi (2022) for the analysis of a (blood) supply chain or Lee et al (2017) already mentioned].

The plan of the paper is the following. Section 2 introduces the Affine Feedforward Stochastic (AFS) network. We discuss its analogies with the MLP neural network, and the activation functions corresponding to this class of model. This stochastic model allows for the modelling of predictive distributions, then for coherent predictions of various transformations of Y and the determination of prediction intervals. It is also applicable to variables such as count or positive variables. Section 3 shows the backward and forward prediction algorithms that extend the Kalman filter algorithms existing for the Gaussian linear state space model. The statistical inference/learning

[9]This updating is for given parameters $c_{01}, c_{02}, \ldots, \delta$. Other algorithms can update the parameter values when the number of observations of (X, Y) increases (see Section 4.1.4 for online inference).

is discussed in Section 4. This section is at the core of the paper. Indeed "training large scale feedforward SNN is notoriously hard since backpropagation is not directly applicable in particular for binary or discrete hidden units". We show how backpropagation algorithm works in the AFSNN framework. We first review the method of moments and the quasi-maximum likelihood approach. The associated objective functions have no simple closed-form expressions in stochastic state (SNN) space models and the optimisation has to be done by the algorithms such as the Stochastic Gradient Descent (SGD) algorithm, possibly adjusted for online use. We also show its version with alternating optimisations. Section 5 extends the modelling and state-space analysis to a stochastic dynamic framework, leading to a new class of Recurrent Neural Networks (RNN). In this framework we discuss the problems of controlled variables and the conditional versus unconditional approaches. The identification issues are examined in Section 6 in both static and dynamic frameworks. Section 7 considers AFS neural networks with constraints between layers. We explain how they can be estimated by indirect inference with the unconstrained AFSNN as the auxiliary model and appropriate nested stochastic approximation algorithms. Section 8 concludes. Additional results are provided in appendices and online appendices. Appendix 1 provides the prediction formulas for AFS neural network with intercepts and possibly with multivariate activation functions. Appendix 2 provides the algorithm for the first and second-order derivatives of the score. Online Appendix 1 reviews the literature on the superposition theorems, that is the possibility to represent every continuous function of several variables by an appropriate composition of functions of one variable. The chain rule and the associated backpropagation algorithm are presented in online Appendix 2 in a general framework. The architecture of the Long Short Term Model (LSTM) is described in online Appendix 3.

2 The Affine State-Space Model

We consider a model with $J+2$ layers including one input layer, one output layer and J hidden layers. The input variables are $X = Z_0$, the hidden state variables are $Z_j, j = 1, \ldots, J$, and the output variables $Y = Z_{J+1}$.
For expository purpose, we assume that the layers have the same numbers of neurons, that is : $\dim Z_j = L, \forall j$ (The extension to different numbers of neurons on the layers is presented in Appendix 1).

2.1 The Affine Feedforward Stochastic (AFS) Network

Let us now define the generative model, that is specify recursively the joint distribution of $Z_j, j = 1, \ldots, J+1$ given $X = Z_0$. This is done by means of the conditional Laplace Transforms (LT), also called conditional Moment Generating Functions (MGF) . In our framework this is done by considering the transitions from one layer to the next one, which are J transition equations for $j = 1, \ldots, J$, and one measurement equation for $j = J+1$.

Definition 1: An AFS network is such that :

i) $E[\exp(u'Z_j)|Z_{j-1}, \ldots, Z_0] = E(\exp(u'Z_j)|Z_{j-1}], j = 1, \ldots, J+1,$

ii) $E[\exp(u'Z_j)|Z_{j-1}] = \exp[a_{j-1}(u)'C_{j-1}Z_{j-1}], j = 1, \ldots, J+1,$

where u is the L-dimensional argument of the Laplace transform, a_j is a L-dimensional function of u and C_j a square (L, L) matrix of parameters.

$Z = (Z_j, j = 0, \ldots, J+1)$ can be considered as a process indexed by layer j, instead of being indexed by time as it is done in the time series literature (see Section 5 for the dynamic framework and the recurrent neural network). Condition i) assumes that this process is Markovian. It is not homogeneous since the transition distributions from Z_{j-1} to Z_j depend on the layer through matrix C_{j-1}, that depends on j. Each matrix C_{j-1} can be seen as characterizing a network. It is a Compound Autoregressive (CaR) or affine process since the conditional log-Laplace transform is linear in the conditioning variable.[10].

Each transition involves two transformations :
i) The state variables $Z_{l,j-1}, l = 1, \ldots, L$ are first combined to construct L combinations $c_{l,j-1}Z_{j-1}, l = 1, \ldots, L$, where $c_{l,j-1}$ is the l^{th} row of matrix C_{j-1}. These latent indexes (or latent scores) do not depend on u that is on the type of (exponential) moments we are interested in.
ii) Then a second combination of indexes, that now depends on u, is applied through the so-called activation function $a_{j-1}(u)$.

[10]CaR processes, also called affine processes when they are defined in continuous time, are widely used in financial applications [Darolles, Gourieroux, and Jasiak (2000), Duffie et al. (2003)].

The definition 1 of the AFS network requires the existence of the conditional Laplace transform and the fact that it characterizes the associated conditional distribution. In the one-dimensional framework and with nonnegative variables Z_j, it exists for a negative argument u. Moreover its knowledge for $u \in (-\infty, 0]$ characterizes the distribution [Feller (1968), (1971)].
These properties can also be extended to the multivariate framework with appropriate support and/or multivariate positivity conditions on Z_j.[11]

2.2 The Conditionally Independent AFS Network

To facilitate the comparison with the deterministic MLP networks and with the deep Restricted Boltzmann Machine (RBM), we can constrain the neurons $Z_{lj}, l = 1 \ldots, L$ to be independent conditional on Z_{j-1}. Thus we do not allow for intralayer connections between the hidden units (states). This constraint is often introduced to facilitate parallel computations.

Proposition 1 : The AFS network satisfies the conditional independence at each layer j, if and only if:

$$a_j(u) = (\alpha_{j1}(u_1), \ldots, \alpha_{jL}(u_L)]',$$

where $u = (u_1, \ldots, u_L)'$ and the functions $\alpha_{jl}(.)$ depend on a single argument.

Then the conditional LT for layer j becomes :

$$E[\exp(u'Z_j)|Z_{j-1}] = \exp\{\sum_{l=1}^{L} \alpha_{j-1,l}(u_l) c_{l,j-1} Z_{j-1}\}.$$

Remark 1 : The set of activation functions is even more constrained in the standard machine learning literature, where the basic functions are equal in each layer, i.e. $\alpha_{j,l} \equiv \alpha_j$ independent of $l = 1, \ldots L$, and the basic functions are equal in all hidden layers : $\alpha_j = \alpha$, independent of $j = 1, \ldots, J$. However, different activation functions can be encountered in some building blocks as in the Long Short Term Model (LSTM), that mixes a logistic activation function, a hyperbolic tangent activation function and an identity activation function on the output layer (see online Appendix 3).

[11] They can also be extended to any type of variable by replacing the conditional moment generating functions by characteristic functions.

Remark 2 : The model is easily extended to include intercept terms within the exponentials (see Appendix 1).

2.3 Log-Laplace Transforms as Activation Functions

The log-Laplace transform interpretations of the quantities $a_j(u)'C_jZ_j$ (resp. $\alpha_{jl}(u_l)c_{lj}Z_j$) imply restrictions on functions a_j (resp. α_{jl}). Let us consider nonnegative variables Z_{jl} and negative arguments u_l. Then the Laplace Transform (LT) characterizes the distribution. Typically the LT are increasing, convex and even totally monotonous, that is infinitely differentiable with respect to $u, u \leq 0$, with positive derivatives of any order.
We will see later on that the functions $\alpha(.)$ are the analogues of the activation functions introduced in MLP and Boltzmann neural networks. Let us provide some examples of such activation functions α standardized by $\alpha(0) = 0$, $\frac{d\alpha(0)}{du} = 1$, and of the associated distributions. They correspond to different neurons with constrained support, as nonnegative, or discrete support.

The activation functions are obtained by considering the moment generating functions of specific parametric families of distributions of the form :

$$E_\lambda[\exp(uY)] = \exp[\lambda_1\alpha(u, \lambda_2)],$$

where the subparameter λ_1, is either in $I\!R = (-\infty, +\infty)$, or in $I\!R^+ = (0, \infty)$. These activation functions are log-Laplace transforms of infinitely divisible distributions (see Appendix 1.1). These distributions are the following :

i) Exponential distribution : $\alpha(u) = -\log(1-u), u \leq 0$;
ii) Poisson distribution $\mathcal{P}(1) : \alpha(u) = \exp(u) - 1$.
iii) Inverse Gaussian or Wald distribution :

$$\alpha(u) = 1 - \sqrt{1-2u}.$$

iv) Normal distribution : $\alpha(u) = u + \nu u^2$.
v) Compound Poisson distribution, i.e. distribution of a Poisson sum of i.i.d. variables : $\alpha(u) = \exp\psi(u) - 1$; where ψ is any log-Laplace transform of a distribution with unitary mean. When the underlying i.i.d. variables are discrete such a compounding allows to reach any infinitely divisible discrete distribution [Feller (1968), Steuter and Van Harn (1979), Lemma 1.2]. For

instance, if the underlying family is negative binomial the activation function is :

$$\alpha(u) = \log p - \log[1 - (1-p)\exp u],$$

where $p \in (0,1)$ and $u < -\log(1-p)$. This activation function associated with an AFS count neural network is an alternative to the deep Boltzmann machine architecture [Salakhutdinov and Hinton (2006) and Hinton, Osindero and Teh (2006) for a structured version applied to the recognition of hand written digits], where the activation function is the logistic function associated with binary variables.

These sets of distributions can be extended, if the basic affine model allows for a nonzero intercept term (see Appendix 1). All the activation functions α are negative on the set of negative argument u. for graphical representations, it can be more convenient to consider the nonnegative functions $u \to \alpha^*(u) = -\alpha(-u)$ defined on nonnegative arguments.

Remark 3 : Depending of the selected family of distributions, the underlying parameter λ_1 can take values in $\mathbb{R}$ or $\mathbb{R}^+$. When its domain is $\mathbb{R}$, the AFS is obtained with λ_1 replaced by a linear function of Z_j, say. When this domain is $\mathbb{R}^+$, it can be replaced by a linear function $C'Z_j$, say, that will imply positivity constrained parameters. Alternatively it can be applied to a positive nonlinear transform such as $\exp(C'Z_j)$, say.

3 Forecasting

The multilayer stochastic state-space model introduced in Section 2 provides closed form forecasting formulas leading to forward and backward deterministic recursive equations.

3.1 Recursive Equations

Let us denote the network parameters by $C = (C_0, \ldots, C_J)$. We are interested in closed form expressions for the nonlinear prediction formulas of all future variables given the input $X = Z_0$, and also for the prediction of the output variable $Y = Z_{J+1}$ given some latent ones :

Proposition 2 :
i) The conditional Laplace Transforms are exponential linear in the conditioning variables :

$$E[\exp(u'Y)|Z_j] = E[\exp(u'Z_{J+1})|Z_j] = \exp[B'_j(u,C)Z_j], \forall j,$$

$$E[\exp(u'Z_j)|X] = E[\exp(u'Z_j)|Z_0] = \exp[A'_j(u,C)Z_0], \forall j.$$

ii) The functions $B_j(u,C), A_j(u,C)$ satisfy the backward and forward recursive equations :

$$B_{j-1}(u,C) = C'_{j-1}a_{j-1}[B_j(u,C)], \forall j,$$

$$A_j(u,C) = A_{j-1}[C'_{j-1}a_{j-1}(u), C], \forall j,$$

with terminal conditions :

$$B_{J+1}(u,C) = u, A_0(u,C) = u.$$

Proof : We have just to show the result by induction and an application of the iterated expectation theorem. Let us consider the backward recursion for B. We have :

$$\begin{aligned} & E[\exp(u'Y)|Z_{j-1}] \\ = \; & E\{E(\exp(u'Y)|Z_j)|Z_{j-1}\} \\ = \; & E\{\exp[B'_j(u,C)Z_j]|Z_{j-1}\} \\ = \; & \exp\{a'_{j-1}[B_j(u,C)]C_{j-1}Z_{j-1}\}. \end{aligned}$$

The backward recursion follows.

Similarly, for the forward recursion, we get :

$$\begin{aligned} & E\{\exp(u'Z_j)|Z_0\} \\ = \; & E\{E[\exp(u'Z_j)|Z_{j-1}]|Z_0\} \\ = \; & E[\exp(a_{j-1}(u)'C_{j-1}Z_{j-1}]|Z_0] \\ = \; & \exp[A_{j-1}[C'_{j-1}a_{j-1}(u), C]'Z_0. \end{aligned}$$

The result follows.

QED

Corollary 1 : The Laplace transform of the predictive distribution of $Y = Z_{J+1}$ given $X = Z_0$ has a closed form expression :

$$\begin{aligned} & E[\exp(u'Y)|X] \\ = \; & E\exp(u'Z_{J+1})|Z_0] = \exp[B_0'(u,C)X] \\ = \; & \exp[A_{J+1}'(u,C)X], \end{aligned}$$

where the functions B_0 (resp. A_{J+1}) are obtained by backward (resp. forward) recursions.

Thus, in the AFS network, the different forecasts can be derived without a numerical algorithm of Monte-Carlo Markov Chain (MCMC), or Gibbs sampling as it is commonly done in restricted Boltzmann networks.

Corollary 2 : i) $A_j(u,C)$ depends on C through $C_0,\ldots,C_{j-1}$ only.
ii) $B_j(u,C)$ depends on C through $C_j,\ldots,C_J$ only.

Proof : This is a direct consequence of the recursive equations in Proposition 2.

QED

As seen later on this property facilitates the backward/forward (backpropagation) algorithms, especially for computing the first and second-order derivatives of the functions with respect to the parameters (see Appendix 2 and online Appendix 2).

3.2 AFS Network and MLP Network

We have shown in Proposition 2 that the AFS network implies deterministic backward and forward sequences of predictions, based on the sequence of $A_j's$ and $B_j's$. Let us now show that the backward deterministic algorithm is of a MLP network type.

Corollary 3 :
Let us consider a conditionally independent AFS network. Then the associated deterministic algorithm for the $B'_j s$ is similar to the deterministic algorithm used in a MLP network for computing the pointwise prediction of $E[\exp(u'Y)|X]$, with activation functions $\alpha_{j,l}, j, l$ varying, and synoptic parameters $C_j, j = 0, \ldots, J$.

Proof : This is just a reinterpretation of the recursive equation for sequence B_j compared to the MLP recursion in Example 1. For instance let us choose a common activation function $a_j \equiv G$, independent of j. Then the backward recursion for B_j becomes :

$$\begin{aligned} B_j &= C'_j G(B_{j+1}) = C'_j G[C'_{j+1} G(B_{j+2})] \\ &= C'_j G[C'_{j+1} G(C'_{j+2} G(B_{j+3}))] \ldots \end{aligned}$$

This is a nonlinear recursive equation alternating the linear transformation C_j and the effect of the nonlinear activation function G.

QED

In order to approximate the predictive distribution of Y given X, the AFS network leads to the AFS pointwise predictions $E[\exp(u'Y)|X = x]$. It is important to note that the AFS framework provides coherent predictions for the different values of argument u. They are not performed separately as usual with the MLPNN. This standard practice does not ensure that the approximations satisfy the structural restrictions on the pointwise predictions such as the monotonicity and convexity of the conditional Laplace transform, for instance.

The standard activation functions of the MLP network are replaced by the basic log Laplace transforms α_{jl}. The choice of a logistic function implicitly assumes predicted binary state-variables. In the AFS network the state variables can be discrete as well as continuous (see Section 2.3). The choice of the logistic function G has been justified theoretically by the possibility to approximate any function of continuous inputs X by a MLP network with a logistic activation function and a sufficiently large number of layers and/or neurons (see online appendix 1 for a review). However, similar results exist for other activation functions and networks .

In practice the NN are implemented with limited numbers of layers and neurons. It is interesting to discuss the type of conditional distributions of Y given X that can be generated by a special NN depending on the number of layers, neurons, and the choice of the activation function. As an illustration, we use an AFS neural network for the core of the neural network, which is completed by a conditionally Gaussian output layer. There are two inputs, two neurons on each intermediate layer and one (conditionally Gaussian) output. When the intercepts are introduced (see Appendix 1), the conditionally independent AFS network has the following structure which is the AFS analogue of the structure in Example 1 :

entry layer :

$$\begin{aligned} E[\exp(u_1 Z_{11})|X] &= \exp[\alpha(u_1)(c'_{10}X + d_{10}) + b(u_1)], \\ E[\exp(u_2 Z_{12})|X] &= \exp[\alpha(u_2)(c'_{20}X + d_{20}) + b(u_2)]. \end{aligned}$$

hidden layer :

$$\begin{aligned} E[\exp(u_1 Z_{21})|Z_1] &= \exp[\alpha(u_1)(c'_{11}Z_1 + d_{11}) + b(u_1)], \\ E[\exp(u_2 Z_{22})|Z_1] &= \exp[\alpha(u_2)(c'_{21}Z_1 + d_{21}) + b(u_2)]. \end{aligned}$$

exit layer :

$$E(\exp(uY)|Z_2) = \exp[u(\gamma' Z_2 + d_2) + u^2/2].$$

The above architecture assumes common activation functions α and the conditional independence of Z_{11}, Z_{12} given X and of Z_{21}, Z_{22} given Z_1, as in the Restricted Boltzmann Machine (RBM).

There exist different representations of a state space model by means of conditional densities, conditional Laplace transforms, or by state equations that describe how the state variables are updated following stochastic shocks. This latter representation is the standard one in the Gaussian linear dynamic network. Let us show this alternative representation in an AFSNN with count neurons in the intermediated layers. Let us introduce the binomial thinning operator (or p-fraction operator) that associates with a nonnegative integer random variable V the variable : $p \otimes V = \sum_{i=1}^{V} U_i$, where the $U'_i s$ are i.i.d.

Bernoulli variables (of parameter p) independent of V. Then the state space representation can be written as :

entry layer :

$$Z_{1l} = p \otimes Z^*_{1l}, \text{with } Z^*_{1l} \sim \mathcal{P}[c'_{l0}X + d_{l0}], l = 1, 2,$$

hidden layer :

$$Z_{2l} = p \otimes Z^*_{2l}, \text{ with } Z^*_{2l} \sim \mathcal{P}(c'_{l1}Z_1 + d_{l1}), l = 1, 2,$$

exit layer :

$$Y = (\gamma' Z_2 + d_2) + \varepsilon, \text{ with } \varepsilon \sim N(0, 1).$$

In the Gaussian linear dynamic network the state equations are written by adding a Gaussian shock to the deterministic linear transformation as in the above exit layer. In the above nonlinear dynamic framework two types of shocks are introduced :
i) the first ones explain how to pass from the conditional expectation $E(Z^*_{1l}|X) = c'_{10}X + d_{10}$ to the count variable Z^*_{1l} itself. This is a conditionally heteroscedastic additive shock to account for the discrete support of the Z^*_{1l} variable.
ii) The second shocks are through the thinning operator with a stochastic intensity applied to Z^*_{1l} instead of a deterministic one to get Z_{1l} with discrete values. They do not appear in an additive way and the number of shocks, i.e. $U_1, \ldots U_v, V = Z^*_{1l}$ depends on the neuron and is stochastic.

4 Statistical Inference/Learning (Static Framework)

The connecting weights C [12] are the parameters that have to be estimated (learned) from observed variables. The statistical inference has to be adjusted to account for the specification in terms of the conditional Laplace transforms. These adjustments concern i) the choice of the (asymptotic) estimation criterion, which is either a moment calibration criterion, or a quasi log-likelihood. This criterion has to facilitate the numerical optimizations (numerical efficiency) while providing reasonable estimators (statistical efficiency) ii) the numerical algorithms used to optimize the criterion. In this respect we mainly focus

[12] In models with biases (intercept) parameters denoted by d both C and d have to be considered. We focus on the C parameters for expository purpose.

on the online inference through the stochastic gradient descent and their alternating analogues.

4.1 Observations and Estimation Criterion

The network is parametrized by the weights $C_j, j = 0, \ldots, J$, that have to be estimated (learned). In the machine learning literature, it is generally assumed that we have independent identically distributed (i.i.d.) observations $(X_i, Y_i), i = 1, \ldots, n$, of the input-output variables, the inner layer state variables $Z_{ji}, j = 1, \ldots, J, i = 1, \ldots, n$, being latent (unobserved). Two types of data can be used : i) The data have been drawn from a given population database by a sampling scheme and the i.i.d. assumption could have to be checked/tested ex-post. [13] ii) In other applications this set is a training sample controlled by the statistician/data scientist, and the i.i.d. assumption has to be taken into account in the training design [see Hinton, Osindero and Teh (2006). Section 6.1 for an example of training sample constructed from the MNIST database].
The parameter $C = (C_0, \ldots, C_J)$ is estimated by optimizing an empirical calibration criterion, that can be a moment criterion to be minimized or a quasi-maximum likelihood criterion to be maximized..

4.1.1 The Method of Moments

In our framework a method of moments based on the conditional Laplace transform can be used. Let us denote $\theta = vecC = (\theta_0', \ldots, \theta_J')'$, with $\theta_j = vecC_j$ the parameter, and introduce instrumental variables $\gamma(X_i)$, functions of the input. The moment restrictions are :

$$m(u, \gamma; \theta) = E\{[\exp[u'Y) - \exp[A'_{J+1}(u, C)X)]\gamma(X)\} = 0, \tag{4.1}$$

for any $u, \gamma(.)$. Let us select a design $\tilde{u}_k, \gamma_k, k = 1, \ldots, K$, and the associated K moments. Then the objective function to be minimized is :

$$L_n(\theta) = m_n(\theta)' V m_n(\theta), \tag{4.2}$$

where V is a (K, K) weighting matrix and the components of $m_n(\theta)$ are :

[13] Typically the i.i.d. assumption is not compatible with data drawn by stratified sampling and does not account for the potential endogenous selectivity biases.

$$m_{nk}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\{[\exp(\tilde{u}_k' Y_i) - \exp\{A_{j+1}(\tilde{u}_k, C)X_i)]\gamma_k(X_i)\}, k = 1, \ldots, K.$$

Then the moment estimator of θ is :

$$\hat{\theta}_n = \arg\min_{\theta} L_n(\theta) = \arg\min_{\theta} m_n(\theta)' V m_n(\theta). \tag{4.3}$$

4.1.2 The Quasi-Maximum Likelihood

By performing a Taylor series expansion of the conditional log-Laplace transform of Corollary 1, we get also closed form expressions of the conditional mean and variance-covariance matrix of Y given X.

Corollary 4 : We have :

$$\begin{aligned} E(Y|X) &= \frac{\partial A_{J+1}'(0, C)}{\partial u} X \equiv \mu(\theta, X), \\ V(Y|X) &= \sum_{k=1}^{p} \frac{\partial^2 A_{J+1,k}(0, C)}{\partial u \partial u'} X_k \equiv \Sigma(\theta, X), \end{aligned}$$

where $A_{J+1,k}$ denotes the k^{+h} component of A_{J+1} and $\theta = vecC$.

Remark 4 : The conditional mean and variance are linear functions of X. Since in practice the input X is often obtained from nonlinear transformations of the underlying input X^* with an economic interpretation, they are nonlinear functions of X^*. Moreover, we deduce that the prediction of Y^2 is a quadratic function of X and therefore a much more complicated nonlinear function of X^*.

The parameter $\theta = vecC$ can also be consistently estimated by maximizing the Gaussian quasi-maximum likelihood :

$$\hat{\theta}_n = \arg\min_{\theta} L_n(\theta),$$

where $L_n(\theta)$ is the opposite of the Gaussian log-likelihood :

$$L_n(\theta) \propto \frac{1}{2}\sum_{i=1}^{n} \log\det \Sigma(\theta, X_i) + \frac{1}{2}\sum_{i=1}^{n} [Y_i - \mu(\theta, X_i)]' \Sigma(\theta, X_i)^{-1} [Y_i - \mu(\theta, X_i)], \tag{4.4}$$

where $\propto$ means up to an additive constant in θ.

4.1.3 Gradient Descent Algorithm

In the state-space model with a quasi-likelihood or moment criterion, the objective function has no closed-form expression in general and it is often approximated by an Expectation-Maximization approach [Dempster et al. (1977)]. This numerical step can be avoided in our framework by using a method of moment based on the Laplace transforms or a quasi maximum likelihood. Indeed, closed form expressions of the conditional moments $E[\exp(u'Y)|X]$, $E(Y|X), V(Y|X)$ are available by Proposition 2 and Corollary 3.

Then, the optimization with respect to θ can be performed numerically by applying a gradient descent approach. The estimate $\hat{\theta}_n$ is approximated as a limit of a sequence $\tilde{\theta}_{nq}, q = 0, 1, \ldots$, such that :

$$\tilde{\theta}_{n,q+1} = \tilde{\theta}_{n,q} - \lambda_{nq} \frac{\partial L_n}{\partial \theta}(\tilde{\theta}_{n,q}), \tag{4.5}$$

where λ_{nq} is a learning rate that can depend on the number of observations n and on the iteration step q. Several choices of learning rates have been proposed in the literature. They may be based on the Barzilai-Borwein approach [Barzilai and Borwein (1988), Raydan (1993)] with :

$$\lambda_{n,q} = \frac{(\tilde{\theta}_{n,q-1} - \tilde{\theta}_{n,q-2})' [\frac{\partial L_n}{\partial \theta}(\tilde{\theta}_{n,q-1}) - \frac{\partial L_n}{\partial \theta}(\tilde{\theta}_{n,q-2})]}{\frac{\partial L_n}{\partial \theta}(\tilde{\theta}_{n,q-1}) - \frac{\partial L_n}{\partial \theta}(\tilde{\theta}_{n,q-2})]' [\frac{\partial L_n}{\partial \theta}(\tilde{\theta}_{n,q-1}) - \frac{\partial L_n}{\partial \theta}(\tilde{\theta}_{n,q-2})]}. \tag{4.6}$$

This is a quasi-Newton approach in which the matrix learning rate is replaced by a scalar learning rate to get weaker requirement, which is a necessary condition for networks with a large number of parameters.[14]

[14]See Bottou, Curtis and Nocedal (2018) for a general presentation of this type of algorithms and of some of their numerical properties.

4.1.4 Online Inference

The Gradient Descent algorithm is not appropriate from an online perspective when the parameter estimate has to be updated when a new observation arises. The idea of online inference is to replace the sequence of GD algorithms indexed by n by a unique algorithm providing directly an estimator $\tilde{\theta}_n$. The approach is especially simple when the objective function is the opposite of a quasi log-likelihood, that is when :

$$L_n(\theta) = -\sum_{i=1}^{n} \log f(y_i, x_i; \theta).$$

Then the Average Stochastic Gradient Descent (ASGD) estimator $\tilde{\theta}_n$ is defined by the recursions :

$$\left\{ \begin{array}{rcl} \theta_n & = & \theta_{n-1} + \gamma_n \dfrac{\partial \log f}{\partial \theta}(y_n, x_n; \theta_{n-1}), \\ \\ \tilde{\theta}_n & = & \tilde{\theta}_{n-1} + \dfrac{1}{n}(\theta_n - \tilde{\theta}_{n-1}), n = 1, 2, \ldots \end{array} \right. \tag{4.7}$$

with a learning rate $\gamma_n = \gamma / n^c, \gamma > 0, c \in (0.5, 1)$, and a given initial value[15].

The first recursion mimicks the recursion (4.5), but with only the score $\dfrac{\partial \log f}{\partial \theta}$ corresponding to the new observation. Then, the results of the first recursion are averaged online. The ASGD estimator $\tilde{\theta}_n$ differs from the quasi maximum likelihood estimator $\hat{\theta}_n$. However, it has been shown that, if the model is well-specified, it shares the same asymptotic properties of consistency, asymptotic normality and that they have the same asymptotic variance covariance matrix [Ruppert (1988), Polyak, and Juditsky (1992)]. The ASGD approach can be extended to other objective functions, in particular, to the M-estimators and moment calibration criteria [see Anastasiou et al. (2019), Dalla Vecchia and Basu (2023), Chen et al. (2025) and an application in Section 7].

[15]The change of sign in the first recursive equation is introduced since this is a maximization problem.

4.2 Alternating ASGD

At each iteration step q of the gradient descent algorithm, or n of the ASGD algorithm, an additional loop can be introduced to take advantage of the layer structure of the network by splitting the parameter θ into subparameters $\theta_J, \theta_{J-1}, \ldots, \theta_0$. Instead of applying the iteration step to θ, we apply it recursively to θ_J while keeping the other parameters constant, then to θ_{J-1}, and so on, up to θ_0. More precisely, the first recursive equation in the standard ASGD (4.7) can be written by subvectors as :

$$\theta_{j,n} = \theta_{j,n-1} + \gamma_n \frac{\partial \log f}{\partial \theta_j}(y_n, x_n, \underline{\theta_{j-1,n-1}}, \bar{\theta}_{j,n-1}), j = 1, \ldots, J, \tag{4.8}$$

where $\underline{\theta_{j-1}} = (\theta_1, \ldots, \theta_{j-1}), \bar{\theta}_j = (\theta_j, \ldots, \theta_J)$.
In the Alternating ASGD (AASGD), these equations are replaced for instance by :

$$\theta_{j,n} = \theta_{j,n-1} + \gamma_n \frac{\partial \log f}{\partial \theta_j}(y_n, x_n, \underline{\theta_{j-1,n}}, \bar{\theta}_{j,n-1}), j = 1, \ldots, J. \tag{4.9}$$

in forward iterations in j (There exists the backward analogue of these equations starting from θ_J).
The AASGD is the online analogue of the alternating optimization, also called the zig-zag approach, or Gauss-Seidel method [Vrahatis et al. (2003), Hautsch et al. (2023)].

This is an alternating method that can improve the convergence to a global minimum of a calibration criterion. Indeed the criterion value $L_n(x_n, y_n, \theta)$ can be convex with respect to θ_j for the other $\theta_k, k \neq j$, fixed, and this for any j, and nonconvex jointly in θ. This partial convexity property is taken into account in the AASGD and ensures that the objective criterion decreases along the iteration steps of the AASGD, not necessarily of the ASGD.
Note also that such property concerns the moment approach based on the LT and its backward application is more relevant due to Corollary 2.

5 Dynamic Framework

There exists a large time series literature on nonlinear stochastic state space models, whereas the analysis of networks in machine learning is still at its

infancy, with dynamic network still often based on the simple LSTM. This could be due to the following reasons :
i) In applications the number of observation dates is often much smaller than the number of observed individuals. For instance the observations can concern 50 millions of individual bank accounts daily. Then the number of individuals is of about ten thousand larger than the number of observation dates.
ii) When the variables Y to predict are continuous as returns, the dynamic has to account for return histories, i.e. for lags. Even when Y has a small dimension say $\dim Y = 2$, if the input includes three lags, say, then the number of continuous inputs is larger than 6. The curse of dimensionality of nonparametric inference is quickly reached.
iii) The activation functions introduced in machine learning as the logistic are not appropriate to capture complicated nonlinear features as speculative bubbles, extreme risks observed on the data as well as the multiple memory effects and the fact that they depend on the transformations of output of interest.

At a first sight, the reader of textbooks in (nonlinear) time series and (dynamic) neural networks can find the following differences :

i) The transitions are stochastic in state space modelling and deterministic in machine learning models.
ii) The machine learning models have several intermediate layers whereas the state space model are presented with one layer in general, but several "dynamic" neuron, called dynamic factors.

Since the AFS NN and the MLPNN have similar architectures, the comparison of their extensions to the dynamic framework will be easier.

5.1 Recurrent Neural Network

The models considered in the previous sections are static with an assumption of i.i.d. input-output observations. However, in applications to Automatic Speech Recognition (ASR) for instance, the variables are indexed by time and the objective is to predict future trajectories. The deep neural network technology (i.e. the network architecture and the learning algorithm) has been extended to the dynamic framework by considering at each period t a

Deep Neural Network (DNN) as in example 1, but with two types of observed inputs : $X_t = (X_t^*, K_{t-1})$, where X_t^* are "exogenous" inputs, i.e. fixed outside the system, and K_{t-1} are summaries of the previous state variables. These extensions are called Recurrent Neural Networks (RNN) and are usually formalized by deterministic state space models [see e.g. Yu and Deng (2015), Chapter 13]. The introduction of a time delay operator gives rise to the memory structure. The recurrent network architecture can be structured to produce different memories, as in the basic Long Short Term Memory (LSTM) model, initially introduced by Hochreiter and Schmidhuber (1997) (see online appendix 3). These models differ by the choice of the summaries K_{t-1}, but also by the assumptions on the future evolutions of the exogenous inputs, that can assume given future evolutions, usually called scenarios, or stochastic evolutions.

Let us now extend the static AFS NN to the dynamic framework .

5.2 The Markov AFS RNN

Let us consider the AFS model with intercept (see Appendix 1). We denote by θ the vector of all parameters, we index the variables by time and assume :

$$X_t = Y_{t-1}. \tag{5.1}$$

Thus there is no exogenous inputs and $K_{t-1} = Y_{t-1}$ We get the following scheme.

Figure 5 : Causal Scheme

$$\to Y_{t-1} \to Z_{1t} \to Z_{2t} \dots Z_{Jt} \to Y_t \to Z_{1,t+1}, \dots,$$

The sequence of conditional distributions is such that :

$$\begin{aligned} E[\exp(u'Y_t)|Y_{t-1}] &= \exp[A'_{J+1}(u;\theta)Y_{t-1} + a_{J+1}(u;\theta)] \\ &= \exp[B'_0(u;\theta)Y_{t-1} + b_0(u,\theta)], \end{aligned} \tag{5.2}$$

and

$$\begin{cases} E[\exp(u'Y_t)|Z_{j,t}] &= \exp[B'_j(u,\theta)Z_{j,t} + b_j(u,\theta)], \\ E[\exp(u'Z_{j,t}|Y_{t-1}] &= \exp[A'_j(u,\theta)Y_{t-1} + a_j(u,\theta)]. \end{cases} \tag{5.3}$$

The causal scheme shows that we have embedded homogenous Markov processes. More precisely,

Proposition 4 : In the Markov AFS RNN,

i) Each process $Y_t \ (= Z_{0t}), (Z_{jt}), j = 1, \ldots, J$, is a Markov process with respect to its own information set.
ii) The transition of process (Z_{jt}) is given by :

$$\begin{aligned} & E[\exp(u'Z_{jt})|Z_{j,t-1}] \\ = \ & \exp[B'_j[A_j(u;\theta);\theta]Z_{j,t-1} + a_j(u,\theta) + b_j[A_j(u;\theta);\theta]]. \end{aligned}$$

Proof : Let us derive the expression of the transition by means of the conditional Laplace transform. We have :

$$\begin{aligned} & E[\exp(u'Z_{j,t})|Z_{j,t-1}] \\ = \ & E\{E[\exp(u'Z_{j,t})|Y_{t-1}]|Z_{j,t-1}\} \\ = \ & E[\exp(A'_j(u;\theta)Y_{t-1}) + a_j(u;\theta)|Z_{j,t-1}] \\ = \ & \exp\{B'_j[A_j(u;\theta);\theta]Z_{j,t-1} + a_j(u;\theta) + b_j[A_j(u;\theta);\theta]\}. \end{aligned}$$

QED

Such a RNN is automatically deep with a large number $J \times T$ layers, where J is the number of layers within each period and T the number of periods. Moreover, this number of periods increases with the number of observations. Therefore, it is important to discuss the asymptotic behaviour of these Markov processes. Let us denote by o the composition operator of functions $A(u), B(u)$ defined by : $AoB(u) = A[B(u)]$, and by A^{oh} the function A composed h times with itself.

Proposition 5 : i) The observed process (Y_t) is strictly stationary if and only if : $\lim_{h\to\infty} A^{oh}_{J+1} = 0$ (or equivalently $\lim_{h\to\infty} B^{oh}_0 = 0$).

ii) Then its stationary distribution has the Laplace transform : $E[\exp(u'Y_t)] = \exp \nu_{J+1}(u;\theta)$, where $\nu_{J+1}(u,\theta) = \nu_{J+1}[A_{J+1}(u;\theta);\theta] + a_{J+1}(u;\theta)$.

Proof : See Darolles et al. (2006).

QED

This shows that the distribution of process (Y_t) can be equivalently characterized by the functions $[A_{J+1}(u), a_{J+1}(u)]$, or by the functions $[A_{J+1}(u), \nu_{J+1}(u)]$. An analogue of Proposition 5 can be written for the other processes $Z_j = (Z_{jt}), j = 1, \ldots, J$.

Corollary 5 : i) The hidden process (Z_{jt}) is strictly stationary if and only if the observed process $(Y_t = Z_{J+1,t})$ is strictly stationary.
ii) Then its stationary distribution has the Laplace transform : $E[\exp(u'Z_{j,t})] = \exp[\nu_j(u;\theta)]$, where $\nu_j(u;\theta) = \nu_{J+1}[A_j(u,\theta),\theta] + a_j(u,\theta)$.

Proof : Indeed we have :

$$\begin{aligned} & E[\exp(u'Z_{j,t})] \\ = \; & E\{E[\exp(u'Z_{j,t})|Y_{t-1}]\} \\ = \; & E[\exp[A_j'(u;\theta)Y_{t-1} + a_j(u;\theta)]] \\ = \; & \exp\{\nu_{J+1}[A_j(u;\theta),\theta] + a_j(u;\theta)]. \end{aligned}$$

The result follows.

QED

The Markov AFS RNN assumes at least as many uncertainties, i.e. stochastic shocks, as the number of latent and observed processes (Z_t, Y_t). It also contains a deterministic equation $X_t = Y_{t-1}$, without any structural interpretation. The Markov AFS RNN allows for deriving the distribution of process (Y_t), or equivalently the distribution of the joint process $(Y_t, X_t) = (Y_t, Y_{t-1})$. This corresponds to an unsupervised learning problem in the machine learning terminology.

5.3 System with control variates

Let us now consider general specifications with input X_t that can include exogenous component. These inputs arise at each date and change overtime. They are usually considered as (partly) controllable. Compared to the Markov AFSNN that leads to unsupervised analyses, these extensions can be partly supervised. Therefore the role of X_t and its exogeneity/controllability features have to be understood.

5.3.1 The model

The architecture of the Markov AFS RNN differs from the architecture of the LSTM in online appendix 3, even if both architectures can capture short and long term effects. Indeed the elements of $A^{oh}_{J+1}(u;\theta)$ do not necessarily tend to zero at the same speed when h tends to infinity. Moreover these speeds can depend on u and/or θ. The main difference is in the presence of deterministic transitions in the LSTM. In the general framework these deterministic features can be due to either the definitions of some variables, or the possibility to control exactly some variables. Let us discuss more precisely this point in the framework of Markov process. We consider the joint processes (X_t, Z_t, Y_t). When this process is homogenous Markov, its parametrized transition density is denoted by :

$$l(X_t, Z_t, Y_t | X_{t-1}, Z_{t-1}, Y_{t-1}; \theta).$$

This transition can be decomposed by applying the Bayes formula as :

$$\begin{aligned} & l(X_t, Z_t, Y_t | X_{t-1}, Z_{t-1}, Y_{t-1}; \theta) \\ = \; & l(Y_t | X_t, Z_t, X_{t-1}, Z_{t-1}, Y_{t-1}; \theta_1) l(Z_t | X_t, X_{t-1}, Z_{t-1}, Y_{t-1}; \theta_2) \\ & l(X_t | X_{t-1}, Z_{t-1}, Y_{t-1}; \theta_3), \end{aligned} \tag{5.4}$$

in which the parameters $\theta_1, \theta_2, \theta_3$ are indexed by the "layers" [16]. Whereas the two transitions from X_t to Z_t, and X_t, Z_t to Y_t (given the past) are assumed stochastic, we have to discuss more carefully the status of the input X_t and of the "input gate". Two cases have to be distinguished.

[16] This decomposition formula is valid in both the deterministic model (with deterministic functions characterizing the transitions) and the stochastic model (with l being the conditional density).

i) The input X_t can be controlled exactly. Then the transition $l(X_t|X_{t-1}, Z_{t-1}, Y_{t-1}; \theta_3)$ can correspond to a degenerate point mass conditional distributions and to a deterministic transition.

ii) The input X_t cannot be controlled, or is controlled, but the deterministic control strategy is not perfectly known. In this case the input gate is itself stochastic conditional to the past.

Then in both cases affine feedforward stochastic NN can be used for each stochastic transition.

The notion of controllability differs from the notion of (strong) exogeneity.

Definition 1 : The input (X_t) is strongly exogenous if and only if :

$$l(X_t|X_{t-1}, Z_{t-1}, Y_{t-1}; \theta_3) = l(X_t|X_{t-1}; \theta_3).$$

Thus we get a classification of the recurrent neural networks in four classes I, II, III, IV (see Table 1), by distinguishing the status of the input variable and the status of the first transition.

Table 1 : Types of RNN

input gate \ input	exogenous	not exogenous
stochastic	I	II
deterministic	III	IV

Any RNN modelling has to be clear on the assumptions concerning controllability and exogeneity. It has to precise the class I-IV to which the model belongs.

5.3.2 Conditional vs unconditional analysis

In the standard analysis, the variables X_t, Y_t are assumed observable and the variables Z_t are hidden. Then, the observable process (X_t, Y_t) is no longer Markov in general and the conditional transition of (X_t, Y_t) depends on all the past trajectory and of all parameters. Let us denote it by : $l^*(X_t, Y_t|\underline{X_{t-1}}, \underline{Y_{t-1}}; \theta)$, where $\underline{X_{t-1}} = (X_{t-1}, X_{t-2}, ..), \underline{Y_{t-1}} = (Y_{t-1}, Y_{t-2}, ...)$. We can reapply the Bayes formula to this joint observable transition to get :

$$l^*(X_t, Y_t|\underline{X_{t-1}}, \underline{Y_{t-1}}; \theta) = l^*(Y_t|\underline{X_t}, \underline{Y_{t-1}}; \theta) l^*(X_t|\underline{X_{t-1}}, \underline{Y_{t-1}}; \theta). \tag{5.5}$$

"An important difference between much of the econometric literature and the machine learning literature is that the econometric literature is often focussed on questions beyond simple pointwise prediction" [Athey and Imbens (2019), Section 6]. These questions are the derivation of prediction intervals, the estimation of structural parameters, the impulse response functions (IRF), or the identification issues. These different questions can require different analyses. In particular can we answer such questions from a conditional analysis of the transitions $l^*(Y_t|\underline{X_t}, \underline{Y_{t-1}}; \theta)$ only, or do we need to base it on the joint transition $l^*(X_t, Y_t|\underline{X_{t-1}}; \underline{Y_{t-1}}; \theta)$?
Let us illustrate this point with the example of an exogenous input.

Example 2 : If the input (X_t) is strongly exogenous, (cases I, III in Table 1) decomposition (5.5) becomes :

$$l^*(X_t, Y_t|\underline{X_{t-1}}, \underline{Y_{t-1}}; \theta) = l^*(Y_t|\underline{X_t}, \underline{Y_{t-1}}; \theta_1, \theta_2) l^*(X_t|X_{t-1}; \theta_3). \tag{5.6}$$

Thus the transition for Y_t given X_t and the past is sufficient to estimate (θ_1, θ_2) by the maximum likelihood. It is also sufficient to predict Y_{T+1} from observation $\underline{X_T}, \underline{Y_T}$. However it is not sufficient to predict Y_{T+2}, Y_{t+3}, for which X_{T+1}, X_{T+2} will have also to be predicted and this will require the knowledge of parameter θ_3.

Remark 5 : In the static framework the machine learning terminology supervised-unsupervised is used to indicate a conditional analysis of output given input, and a joint analysis of input-output, respectively [Athey and Imbens (2019), Section 2.2]. [17] Example 2 shows that in a dynamic framework a supervised, that is a conditional analysis, is not appropriate in general.

5.3.3 Control and supervision

Let us now discuss in more detail the controlled vs supervised learning in an extended version of model (5.6), where the input gate $l_t(X_t|X_{t-1}, Z_{t-1}, Y_{t-1}; \theta_3)$ could also depend on the date t. The supervised learning becomes relevant if

[17] An alternative terminology is discriminative/generative model or classifier. These conventional terminologies are currently not well established, even if they are largely used in the machine learning literature.

parameter θ_3 is known, that can be the case if the model is used by the controller himself. Then the future inputs can be generated either in a deterministic or in a stochastic way depending on the assumption on the input gate and this known input gate defines the strategy of the controller. Typically a central bank can fix the prime rate as a function of observed lagged inflation rates and changes in GDP along a Taylor rule. This leads to either deterministic, or stochastic scenarios on the future values of the input. However the strategy of the controller is not necessarily revealed by the controller (the Central Bank) to the analyst (the statistician). Then, even if there is an underlying control, the input gate becomes stochastic with unknown parameter θ_3 and this leads to a joint unsupervised (generative) analysis. This arises when the behaviour (reaction) of the controller has to be anticipated in a general equilibrium framework.

5.3.4 The affine recurrent architectures

Let us now discuss the different affine state space architectures based on the recursive decomposition in the right hand side of eq. (5.4). The recursive model is defined by the associated Laplace transforms with 3 layers. [18]

input layer :

$$\begin{aligned} & E[\exp(u'X_t)|\underline{X_{t-1}}, \underline{Z_{t-1}}, \underline{Y_{t-1}}] \\ = \; & \exp[a'_{01}(u)C'_{01}X_{t-1} + a'_{02}(u)C'_{02}Z_{t-1} + a'_{03}(u)C'_{03}Y_{t-1} + b_0(u, C_0)]. \end{aligned}$$

hidden layer :

$$\begin{aligned} & E[\exp(u'Z_t)|\underline{X_t}, \underline{Z_{t-1}}, \underline{Y_{t-1}}] \\ = \; & \exp[a'_{11}(u)C'_{11}X_t + a'_{12}(u)C'_{12}X_{t-1} + a'_{13}(u)C'_{13}Z_{t-1} + a'_{14}(u)C'_{14}Y_{t-1} + b_1(u, C_1)] \end{aligned}$$

output layer :

[18] The hidden layer could also be decomposed into a sequence of intermediate hidden layers.

$$
\begin{aligned}
& E[\exp(u'Y_t)|\underline{X_t}, \underline{Z_t}, \underline{Y_{t-1}}] \\
= \; & \exp[a'_{21}(u)C'_{21}X_t + a'_{22}(u)C'_{22}Z_t + \\
& a'_{23}(u)C'_{23}X_{t-1} + a'_{24}(u)C'_{24}Z_{t-1} + a'_{25}(u)C'_{25}Y_{t-1} + b_2(u, C_2)].
\end{aligned}
$$

It is easily checked by applying the iterated expectation theorem that the joint conditional log-Laplace transform :

$$\log E[\exp(u'Y_t + v'Z_t + \omega'X_t)|\underline{X_{t-1}}, \underline{Z_{t-1}}, \underline{Y_{t-1}}],$$

is an affine function of $X_{t-1}, Z_{t-1}, Y_{t-1}$ for any arguments u, v, ω. In particular the joint process (X_t, Z_t, Y_t) is Markov of order 1.

When taking into account the lagged values, the number of weights is exploding. The ML literature on deterministic recurrent neural networks has often eliminated several interaction effects. In our affine stochastic framework the analogues of these constrained deterministic models would be :

input layer* :

$$
\begin{aligned}
& E[\exp(u'X_t)|\underline{X_{t-1}}, \underline{Z_{t-1}}, \underline{Y_{t-1}}] \\
= \; & \exp[a'_{01}(u)C'_{01}X_{t-1} + a'_{02}(u)C'_{02}Z_{t-1} + b_0(u, C_0)].
\end{aligned}
$$

hidden layer* :

$$
\begin{aligned}
& E[\exp(u'Z_t)|\underline{X_t}, \underline{Z_{t-1}}, \underline{Y_{t-1}}] \\
= \; & \exp[a'_{11}(u)C'_{11}X_t + a'_{12}(u)C'_{12}Z_{t-1} + b_1(u, C_1)].
\end{aligned}
$$

output layer* :

$$
\begin{aligned}
& E[\exp(u'Y_t)|\underline{X_t}, \underline{Z_t}, \underline{Y_{t-1}}] \\
= \; & \exp[a'_{22}(u)C'_{22}Z_t + b_2(u, C_2)].
\end{aligned}
$$

This constrained version assumes that all the information useful to predict nonlinearly Y_t is contained in Z_t. Moreover X_t, Z_{t-1} summarizes all the information useful to predict the network at date t (see Yu and Deng (2015), Section 13, eq(13.1), (13.2)].

The output and hidden layers are completed by an input layer that allows for different notions of inputs. For instance, the input X_t is strictly exogenous iff $C_{02} = 0$.

5.4 Statistical inference/learning (dynamic framework)

.
The learning algorithms described in Section 4 can still be used, but their asymptotic properties under the identification assumption and serial dependence have not been derived yet. These properties require the use of the mixingale theory [McLeish (1975)]. For the SGD algorithm applied in a recursive way, the consistency and asymptotic normality have been obtained in Kuan and White (1994), Theorem II.2.1, Corollary II.3.6, under a set of high level assumptions.
It does not seem that similar properties and the additional asymptotic efficiency have yet been obtained for the ASGD.

6 Identification Issues

Once a parametric model is introduced and assumed well specified, what are the model characteristics (i.e. parameters) that can be recovered from a set of observations? Can we recover the parameter itself? Can we recover what we are interested in? The identification analysis of the model should precede any estimation step (learning, training step), but is totally absent from the machine learning literature with a risk of misleading interpretations of some results.
A complete discussion of the identification in RNN is out of the scope of this paper. We prefer to illustrate the difficulty of identification in a progressive way. First we consider a deterministic static network with a fixed number of observations. Then the analysis is extended to stochastic networks. Finally we discuss the use of algorithmic learning when the model is not identifiable.

6.1 Static deterministic network

For any static deterministic network for i.i.d. $(X_t, Z_t, Y_t), t = 1, \ldots, T$, it is useful to write the output of the final layer as a function of the input by "integrating out" the hidden layers. Then we get a nonlinear relationship :

$$Y_t = H(X_t, \theta), t = 1, \ldots, T,$$

where H depends on the activation functions assumed to be known. The system is characterized by the parameter θ, that gathers all the connecting weights. Let us assume that this deterministic model is well specified, i.e. the output data have been obtained from the input data with a true value θ_0 of the parameter. Can we recover this true value or part of it ? This leads to the following definitions.

Definition 2 : Static deterministic network, finite sample.

i) The identified set is :

$$I\Theta_T = \{\theta : Y_t = H(X_t, \theta), t = 1, \ldots, T\}.$$

ii) The parameter θ_0 is identifiable if and only if $I\Theta_T = \{\theta_0\}$ reduces to the singleton θ_0.
iii) A function $h(\theta)$ of parameter θ is identifiable iff $h(I\Theta_T)$ reduces to a singleton.
iv) Two models with parameter values in the identified set are said observationally equivalent.

When θ_0 is identified, we can identify without errors the structure of the black-box.
A necessary condition for identification is the order condition.

Proposition 6 : The order condition is :

$$T \dim Y_t \geq \dim \theta.$$

Thus, the introduction of a too large number of parameters creates identification issues if the number of observations is not sufficiently large.

Let us now give examples of identification issues for deterministic networks with one hidden layer and two neurons. First we consider extreme activation functions to facilitate the discussion.

Example 3 : Linear model.

The static model for $X_t = (X_{1t}, X_{2t}), Z_t = (Z_{1t}, Z_{2t}), Y_t = (Y_{1t}, Y_{2t})$ is :

$$\begin{cases} Z_t &= C_1 X_t + d_1, \\ Y_t &= C_2 Z_t + d_2, \quad t = 1, \ldots, T. \end{cases}$$

We deduce :

$$Y_t = H(X_t; \theta) = C_2 C_1 X_t + d_2 + C_2 d_1, t = 1, \ldots, T.$$

We immediately see that we can identify the "reduced form" parameters $C^* = C_2 C_1, d^* = d_2 + C_2 d_1$, but not C_1 and C_2, d_1 and d_2 separately.
Moreover, since C_1 and d_1 are not identifiable, we cannot identify the hidden features : $Z_t = C_1 X_t + d_1$.

Example 4 : Digit model

Let us now consider binary variables $X_{1t}, X_{2t}, Z_{1t}, Z_{2t}, Y_{1t}, Y_{2t}$ and a TLU (Heaviside) activation function. The model is :

$$\begin{cases} Z_{1t} &= \mathbb{1}_{C_{011} X_{1t} + C_{012} X_{2t} + d_{01} > 0}, \\ Z_{2t} &= \mathbb{1}_{C_{021} X_{1t} + C_{022} X_{2t} + d_{02} > 0}. \end{cases}$$

$$\begin{cases} Y_{1t} &= \mathbb{1}_{C_{111} Z_{1t} + C_{112} Z_{2t} + d_{11} > 0}, \\ Y_{2t} &= \mathbb{1}_{C_{121} Z_{1t} + C_{122} Z_{2t} + d_{12} > 0}. \end{cases}$$

Since the inputs are binary, we see that, for given θ, the function H is from $\{0,1\}^2$ to $\{0,1\}^2$. Then, we get at most 16 equations to be solved for θ. This shows that the order condition for identification is not sufficient.
The pattern of the identified set can be derived explicitly. For instance it is easily checked that $X_1 = X_2 = 0$ and $Y_1 = 1$ is equivalent to :

$$\{d_{01} > 0, d_{02} > 0, C_{111} + C_{112} + d_{11} > 0\}$$

$$\cup \{d_{01} < 0, d_{02} > 0, C_{112} + d_{11} > 0\}$$

$$\cup \{d_{01} < 0, d_{02} < 0, d_{11} > 0\}$$

$$\cup \{d_{01} > 0, d_{02} < 0, C_{111} + d_{11} > 0\}.$$

This shows the type of set information on θ contained in some observations. The use of the TLU activation function in the 1960's led researchers away from the gradient descent algorithms, since the derivative of a TLU is zero almost everywhere. This explains the success of sigmoid activation functions introduced by Cowan (1967).

Let us also show that there exist other reasons for a lack of identification in deterministic neural networks.

Example 5 : Ex-ante segmentation.

The input X_t can characterize an ex-ante segmentation. Then, the components of X_t are binary variables that sum up to one. For instance, this segmentation can be constructed from a quantitative variable X_t^* by state discretization. In this framework, any function $H(X_t)$ can be written as a linear function of X_t. Therefore, the model can be rewritten with $\dim X_t$ parameters and the order condition is not satisfied.

Example 6 : Definition of neurons

Let us consider the MLPNN in Example 1. By construction the specification is symmetric with respect to the neurons of layer 1 (resp. layer 2). Thus for a given layer we cannot identify which is the first neuron of the layer.

Among the observationally equivalent models in Examples 3 and 4, there are networks with a single layer. It can be interesting to examine such models that could allow us to replace a true hidden black-box by an observationally equivalent one with a simplified structure.
In a general framework with other activation functions (logistic, RELU, tanh), other numbers of layers and neurons, and other assumptions on the supports of the observed and hidden variables, the ex-ante analysis of identification can become untractable, even in a deterministic framework. However, some identification issues can be revealed ex-post when the parameter θ is estimated numerically. Let us consider the approximation of θ through a gradient descent algorithm (see Sections 4.1.3, 4.1.4). If the parameter θ is not identifiable, these algorithms will not converge to the true value θ_0 in general. In fact the approximations at step q of the algorithm will at best converge to the identified set, and, if they converge pointwise (when q increases), the limit can depend on the starting value used in the algorithm. This dependence

on the starting value has been frequently observed in the machine learning practice and is largely due to identification issues.

6.2 Stochastic networks

The identification analysis is much more complex in stochastic networks. Indeed the introduction of shocks to each layer and neuron make impossible to gain the exact knowledge of θ_0 in finite sample, that is for a given number of observations T. Then, the definitions of an identified set and of an identifiable parameter will be written in an asymptotic framework that will depend on high level assumptions.
Let us consider the static framework. The parametric model is characterized by the specification of the conditional distribution of Y_t given X_t, or equivalently by the conditional Laplace transform :

$$E_\theta[\exp(u'Y_t)|X_t] \equiv \psi(u, X_t; \theta), u \text{ varying.}$$

This function is the analogue of the function $H(X_t; \theta)$ in the deterministic model, but depends on the additional argument u to capture all nonlinearities in Y.

Then the literature has considered two types of asymptotic analysis.

i) When X is assumed fixed to the observations $X_1, \ldots, X_T$ in repeated samples, all the asymptotic knowledge on the parameters is conveyed through the conditional Laplace transforms :

$$\psi(u; X_t; \theta), u \text{ varying, } t = 1, \ldots, T.$$

ii) When the observations $(X_t, Y_t), t = 1, \ldots, T$, are assumed i.i.d., all the asymptotic knowledge on the parameters is conveyed through the joint Laplace transform :

$$\begin{aligned} E_{\theta,\mu_0}[\exp(u'Y_t + v'X_t)] &= E_{\mu_0}[exp(v'X_t) + \psi(u, X_t, \theta)] \\ &\equiv \tilde{\psi} \quad (u, v; \theta, \mu_0), \end{aligned}$$

where μ_0 corresponds to the marginal distribution of X_t.

This leads to the following definitions that extend Definition 2 to the stochastic framework.

Definition 3 : Static stochastic network.

i) Fixed input in repeated samples (i.e. supervised)
The asymptotic identified set is :

$$A\Theta_T = \{\theta, \psi(u, X_t; \theta) = \psi_0(u, X_t), t = 1, \ldots, T, u, \text{ varying}\},$$

where $\psi_0(u, X_t)$ is the true conditional Laplace transform of Y_t given X_t. This identified set depends on $X_1, \ldots, X_T$.

ii) The i.i.d. sample (i.e. unsupervised).
The asymptotic identified set is :

$$A\Theta = \{\theta : \tilde{\psi}(u, v; \theta, \mu_0) = \tilde{\psi}_0(u, v), u, v \text{ varying}\},$$

where $\tilde{\psi}_0(u, v)$ is the true joint Laplace transform of (X_t, Y_t). This identified set depends on the marginal distribution μ_0 of the input.

iii) Under the assumption of a well-specified asymptotic model, with a true value θ_0 of the parameter, θ_0 belongs in the asymptotic identified set.

These definitions depend on the assumptions on the model. It is easily seen that the parameter θ_0 can be identifiable for a set $X_1, \ldots, X_T$ of input observations, but not identifiable for another subset of observations $\tilde{X}_1, \ldots, \tilde{X}_T$. In the i.i.d. case, it can be identifiable for some μ_0, but not for another $\tilde{\mu}_0$.

Moreover, these are high level assumptions, that cannot be checked in practice, from only a finite, even large number of observations.

6.3 Simulations under partial identification

In multivariate dynamic systems with hidden layers the identification issues are due to partial observations: we observe the time series of inputs and outputs, X_t, Y_t but not the internal hidden variables Z_t. This render difficult the identification of the internal system. Let us consider such a framework where θ is not identifiable and assume that the variables X_t, Z_t, Y_t can be easily simulated for any given θ.

There exists a lot of online learning algorithms that have been introduced to estimate θ. Since θ is not identifiable, we can at best get approximations

$\hat{\theta}_T$ that converge to the identified set, $A\Theta$, when T tends to infinity, not necessarily pointwise. Even if they converge pointwise, the limiting value can depend on the starting value used in the algorithm and are in general different from the true value θ_0.
However the generative model is often used to simulate the trajectories of X_t, Z_t, Y_t from the system in which θ is replaced by $\hat{\theta}_T$. These simulated values will depend on the selected $\hat{\theta}_T$ and, if $\hat{\theta}_T$ is close to a point θ_0^* of the identified set, they will depend on θ_0^*. However by definition of the observations and of the associated identified set, the simulated trajectories $(X_t^s(\theta_0^*), Y_t^s(\theta_0^*)]$ of (X_t, Y_t) have the same distribution as the trajectories of (X_t, Y_t) and this holds for any (θ_0^*) in $A\Theta$. Therefore, the standard simulations $[X_t^s(\hat{\theta}_T), Y_t^s(\hat{\theta}_T)]$ can be used to approximate nonparametrically any function of the distribution of (X_t, Y_t) such as the predictive distribution of Y_{t+h} given X_t, Y_t. Indeed the distribution of process (X_t, Y_t), is identifiable. However the simulations cannot be used to get the predictive distribution of the Z variable, i.e. the filtering distribution, since this one is not nonparametrically identifiable in general.

6.4 Starting Values as Identifying Restrictions

To illustrate the discussion above, let us consider a parametric model with the conditional distribution : $l(y_t|x_t;\theta) = l^*(y_t|x_t;a(\theta))$, where θ_0 (resp. $a_0 = a(\theta_0)$) denotes the true value of θ (resp. of a). We assume that the series (X_t, Y_t) is i.i.d. and that a_0 is asymptotically identifiable. Then the first subsystem of the ASGD algorithm applied online (see 4.1.4) is :

$$\tilde{\theta}_{t+1} = \tilde{\theta}_t + \gamma_t \frac{\partial a'}{\partial \theta}(\tilde{\theta}_t) \frac{\partial \log l^*}{\partial a}(y_t|x_t;a(\tilde{\theta}_t)), t = 1, \ldots, T.$$

Let us now assume that $a(.)$ is linear : $a(\theta) = \theta_1 + A\theta_2$, say.

Proposition 6 : Under the assumptions above,
i) The online ASGD estimator $\tilde{a}_t = a(\tilde{\theta}_t)$ is consistent of $a_0 = a(\theta_0)$.
ii) The starting values of $\beta = \theta_2 - A'\theta_1$ are identifying restriction for the non identifiable parameter $\beta_0 = \theta_{20} - A'\theta_{10}$.

Proof :

We get :

$$\left\{ \begin{array}{rcl} \tilde{\theta}_{1,t+1} & = & \tilde{\theta}_{1,t} + \gamma_t \dfrac{\partial \log l^*}{\partial a}(y_t|x_t; a(\tilde{\theta}_t)), \\ \\ \tilde{\theta}_{2,t+1} & = & \tilde{\theta}_{2,t} + \gamma_t A' \dfrac{\partial \log l^*}{\partial a}(y_t|x_t; a(\tilde{\theta}_t)), \end{array} \right.$$

It follows that :

$$\tilde{\theta}_{2,t+1} - A'\tilde{\theta}_{1,t+1} = \tilde{\theta}_{2,t} - A'\tilde{\theta}_{1,t} = \tilde{\theta}_{2,0} - A'\tilde{\theta}_{1,0}, \tag{6.1}$$

where $\tilde{\theta}_0$ are the starting values of the algorithm. Then the first equation of the system above can be rewritten as :

$$\tilde{\theta}_{1,t+1} = \tilde{\theta}_{1,t} + \gamma_t \frac{\partial \log l^*}{\partial a}[y_t|x_t; (Id + AA')\tilde{\theta}_{1,t} + A(\tilde{\theta}_{2,0} - A'\tilde{\theta}_{1,0})]. \tag{6.2}$$

Equations (6.1)-(6.2) have the following interpretations.

i) The parameter $\beta = \theta_2 - A'\theta_1$ is purely non identifiable and its estimator (posterior value) is equal to its starting value $\tilde{\beta}_0 = \tilde{\theta}_{20} - A'\tilde{\theta}_{10}$.
ii) The equation (6.2) corresponds to the SGD applied to θ_1 estimated under the "identifying restriction" : $\theta_2 - A'\theta_1 = \tilde{\theta}_{2.0} - A'\tilde{\theta}_{1,0}$. By the standard properties of SGD, we deduce that the online estimator $\tilde{\theta}_t$ converges to the same limit as the maximum likelihood estimator of θ under the constraint $\theta_2 - A'\theta_1 = \tilde{\theta}_{2,0} - A'\tilde{\theta}_{1,0}$. In particular it converges pointwise to a pseudo true value $\theta_0^* \equiv c(\theta_0, \tilde{\theta}_0)$. Moreover, by construction, $\tilde{a}_t = a(\tilde{\theta}_t)$ converges to $a[c(\theta_0, \tilde{\theta}_0)] = a(\theta_0) = a_0$, independent of the starting value $\tilde{\theta}_0$.

QED

More precisely, we have :

$$\begin{array}{rcl} \tilde{\theta}_{1,t} & = & (Id + AA')^{-1}[\tilde{a}_t - A(\tilde{\theta}_{2,0} - A'\tilde{\theta}_{1,0})], \\ \\ \tilde{\theta}_{2,t} & = & (Id + A'A)^{-1}[A'\tilde{a}_t + (\tilde{\theta}_{2,0} - A'\tilde{\theta}_{1,0})]. \end{array} \tag{6.3}$$

Since $\tilde{a}_t$ inherits the asymptotic properties of the unconstrained maximum likelihood estimator of parameter a, that is : $\sqrt{T}(\tilde{a}_T - a_0) \to_d N(0, V(a_0)]$, where $V(a_0)$ is the asymptotic efficiency bound for a. We deduce easily the pseudo true value and the limiting degenerate Gaussian distribution of $\tilde{\theta}_t$.

7 AFS Neural Network with Between Layer Restrictions

7.1 Constrained Model

At this stage, we have considered the unconstrained static or recurrent AFS neural networks with in general a rather large number of parameters $\theta_j, j = 1, \ldots, J$, where j indexes the layer. Their main advantage is to be easy to estimate numerically by selecting an appropriate learning criterion and by using stochastic gradient descent and backpropagation algorithms. Their main drawback is the often too large number of parameters, that can induce the identification issues discussed in Section 6.

To solve this identification issues, we can introduce constraints on parameters, that is assume that the θ parameters depends on a parameter η with a much smaller dimension, such that η is identifiable. Then two cases can be distinguished :
i) If $\eta = (\eta_j, j = 1, \ldots, J)$, where $\theta_j = \theta_j(\eta_j)$, we get within layer restrictions. Then the SGD and backpropagation algorithms are still easy to apply.
ii) However, we can have between layer restrictions $\theta = \theta(\eta)$. This will destroy the recursive layer structure of the algorithms and their numerical performances.

To circumvent this difficulty a method from machine learning called variational inference or Evidence Lower Bound (ELBO) approach will introduce an unconstrained NN easy to estimate numerically in order to get an approximation of the structured model [see Hinton and Van Camp (1993) for the introduction of the ELBO approach, and Blei et al. (2018) for a survey on variational inference for statisticians]. However, these approximate approaches do not provide statistically consistent estimated constrained models.

7.2 Indirect Inference on Generative AFSNN

Let us now assume that it is easy to simulate artificial data for given value of parameter θ (and values of y in the dynamic case) In other words, we assume a generative model. Let us also assume that the constrained model is well-specified with true value η_0 (resp. $\theta_0 = \theta(\eta_0)$) of parameter η (resp. θ). Then we can apply the indirect inference approach [Gourieroux et al.

(1993)] along the following steps :

Step 1 : Use the observations $y_1, \ldots, y_T$ to estimate θ in the unconstrained model by applying the SGD/back-propagation algorithm with given starting value $\theta_0^* = \theta(\eta_0^*)$, learning rates and stopping rules. This provides an unconstrained estimator $\hat{\theta}_T$ of θ. Note that this estimator converges statistically, but not necessarily to θ_0 due to the identification issue discussed in Section 6.

Step 2 : For each given value of η, we can create simulated data $y_t^s(\eta), t = 1, \ldots, T$, by applying the generative constrained AFSNN, and replicate such simulated sets for $s = 1, \ldots, S$.

Step 3 : These artificial data sets can be used to reapply the SGD/Backpropagation algorithm on the unconstrained model with the same starting values $\theta_0^* = \theta(\eta_0^*)$, learning rates and stopping rules. This provides S unconstrained estimators of θ $: \hat{\theta}_T^s(\eta), s = 1, \ldots, S$.

If T is large, the $\hat{\theta}_T^s(\eta_0), s = 1, \ldots, S$ and $\hat{\theta}_T$ would converge to the same value in the identified set, but this value could be different from $\theta_0 = \theta(\eta_0)$.

Step 4 : To adjust for this possible asymptotic bias and deduce a statistically consistent estimator of y_0 in the constrained model, we can solve the problem :

$$\hat{\eta}_T^S = \arg\min_{\eta} \left(\frac{1}{S} \sum_{s=1}^{S} \hat{\theta}_T^s(\eta) - \hat{\theta}_T \right)' \left(\frac{1}{s} \sum_{s=1}^{S} \hat{\theta}_T^s(\eta) - \hat{\theta}_T \right).$$

It has been shown recently that this matching problem can be solved numerically efficiently by a SGD algorithm, even if the $\hat{\eta}_T^S$ function does not have the form of an M-estimator [Chen et al (2025)]. It is called SGMM algorithm. Note that this latter optimization is with respect to a much lower number of parameters η than the first optimization that are with respect to θ.

To summarize the use of the algorithms :

i) The constrained model cannot be directly estimated from the observations by a simple SGD algorithm.
ii) In the indirect approach, we get two loops of SGD/backpropagation algorithms. The pair SGD/backpropagation algorithms is used in steps 1 and 3 on observations and simulated data, respectively. Then the SGMM

algorithm is used in step 4.

The indirect inference approach is statistically consistent. Because it rests on numerical stochastic optimizations [Robins and Monro (1951), Kushner and Yin (1997)], it tends to be numerically faster than alternative approaches based on Monte-Carlo Markov Chain (MCMC).

8 Concluding Remarks

The aim of this paper was to relate the machine learning approach of deep neural network with the classical literature on stochastic state space models. We have shown that the architecture of the multilayer perceptron neural network can be represented as a deterministic state space model and that standard recurrent neural networks such as the LSTM also have deterministic hidden state equations. These deterministic transitions imply nonlinear accounting equations between the hidden and observed variables, that create degeneracies in statistical inference and instabilities when applying the estimation (learning) approaches. They also largely explain the lack of probabilistic tools in the machine learning literature such as confidence interval, prediction intervals, or diagnostic tools.

To solve this problem, we have introduced a class of stochastic state space models for static or recurrent neural networks. These models are based on affine dynamic structures largely used in applications to Finance—Insurance, and more generally to the analysis of risk dynamics. In particular the class of AFS neural networks assumes a number of stochastic shocks larger or equal than the total number of variables to avoid arbitrary accounting equations. We show that the AFS neural networks provide an architecture of layers, neurons, activation functions similar to the architecture of the MLP neural network, but written on the predictive distributions through their conditional Laplace transforms, instead of being written in "expectation" only. We also show that these models can be estimated by the quasi maximum likelihood or by the method of moments based on Laplace transforms. They can be estimated online by the average stochastic gradient descent or an alternating variant. A chain rule (backpropagation) can also be used to compute the scores appearing in the SGD algorithms.

We also explain how these online estimators can be used when the models are not identifiable, and why the starting values in the algorithmic approaches

can be seen as identification restrictions. Then we considered AFSNN with constraints between layers and we show how they can be estimated by indirect inference using two loops of numerical stochastic optimizations, the first one for the estimation of the unconstrained AFSNN, the second one for the bias adjustment.

References

[1] Agarwal, A., Negahban, S., and M., Wainwright (2010) : "Fast Global Convergence Rates of Gradient Methods for High-Dimensional Statistical Recovery', in Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and A., Culotta (eds), Advances in Neural Information Processing Systems, 23, Cambridge, MIT Press.

[2] Al-Osh, M., and A., Azaid (1987) : "First-Order Integer Valued Autoregressive (INAR (1)) Processes", Journal of Time Series Analysis, 8, 261-275.

[3] Athey, S., and G., Imbens (2019) : "Machine Learning Methods that Economists Should Know About", Annual Review of Economics, 11, 685-725.

[4] Anastasion, A., Balasmebramanian, K., and M., Erdogdu (2019) : "Normal Approximation for Stochastic Gradient Descent via Non-Asymptotic Rates of Martingale CLT", in Beygelzimer, A., and D., Han (eds), Proceeding of Machine Learning Research, p 115-117, Cambridge.

[5] Barzilai, J., and M., Borwein (1988) : "Two-Point Step Size Gradient Methods", IMA Journal of Numerical Analysis, 8, 141-148.

[6] Bishop, C. (1995) : "Neural Networks for Pattern Recognition", Oxford University Press.

[7] Blei, D., Kucukelbir, A., and J., McAuliffe (2018) : "Variational Inference : A Review for Statisticians", arXiv 1601.00670v9.

[8] Bottou, L., Curtis, F., and J., Nocedal (2018) : "Optimization Methods for Large Scale Machine Learning", SIAM Review, 60, 223-311.

[9] Breiman, L. (2001) :"Statistical Modeling:the Two Cultures", Statistical Review, 16, 199-231.

[10] Chen, X., Lee, S., Liao, Y., Seo, M., Shin, Y., and M., Song (2025) : "SGMM : Stochastic Approximation to Generalized Method of Moments", Journal of Financial Econometrics, 23-

[11] Conn, D., and G., Li (2019) : "An Oracle Property of the Nadaraya-Watson Kernel Estimator for High Dimensional Nonparametric Regression", Scand. J. Statist., 46, 735-764.

[12] Cowan, J. (1967) : "A Mathematical Theory of Central Nervous Activity", Unpublished Ph.D. dissertation, University of London.

[13] Dalla Vecchia, R., and D., Basu (2023) : "Online Instrumental Variable Regression : Regret Analysis and Bandit Feedback", ArXiv 2302.09357.

[14] Darolles, S., Gourieroux, C., and J., Jasiak (2006) : "Structural Laplace Transform and Compound Autoregressive Models", Journal of Time Series Analysis, 27, 477-503.

[15] Dempster, A., Laird, N., and D., Rubin (1977) : "Maximum Likelihood from Incomplete Data via the EM Algorithm", JRSS, B, 39, 1-39.

[16] Duffie, D., Filipovic, D., and W., Schachermayer (2003) : "Affine Processes and Application in Finance", Annals of Applied Probability, 13, 984-1053.

[17] Feller, W. (1968) : "An Introduction to Probability Theory and its Applications 1", 3rd ed., Wiley, New-York.

[18] Feller, W. (1971) : "An Introduction to Probability Theory and its Applications 2", 2nd ed, Wiley, New-York.

[19] Gallant, R., (1981) : "On the Bias in Flexible Functional Forms and an Essentially Unbiased Form : The Fourier Flexible Form", Journal of Econometrics, 15, 211-245.

[20] Gallant, R., and H., White (1991) : "On Learning the Derivative of an Unknown Mapping with Multilayer Feedforward Networks", Neural Networks, 5, 129-138.

[21] Geenens, G. (2011) : "Curse of Dimensionality and Related Issues in Nonparametric Functional Regression", Statistics Surveys, 5, 30-43.

[22] Gourieroux, C., and J., Jasiak (2006) : "Autoregressive Gamma Processes", Journal of Forecasting, 25, 129-152.

[23] Gourieroux, C., Monfort, A., and E., Renault (1993) : "Indirect Inference", Journal of Applied Econometrics, 8, 85-118.

[24] Hautsch, N., Okhrin, O., and A., Ristig (2023) : "Maximum Likelihood Estimation using the Zig-Zag Algorithm", Journal of Financial Econometrics, 21, 1346-1375.

[25] Haykin, S. (1998) : "Neural Networks : A Comprehensive Foundation", (2 ed.), Prentice Hall.

[26] Hinton, G., Osindero, S., and Y., Teh (2006) : "A Fast Learning Algorithm for Deep Belief Nets", Neural Computation, 18, 1527-1534.

[27] Hinton, G., and D., Van Camp (1993) : "Keeping the Neural Networks Simple by Minimizing the Description Length of the Weights", in Computational Learning Theory.

[28] Hochreiter, S., and J., Schmidhuber (1997) : "Long Short Term Memory", Neural Computation, 9, 1735-1780.

[29] Kuan, C., and H., White (1994) : "Artificial Neural Networks : an Econometric Perspective", Econometric Reviews, 13, 1-91.

[30] Kushner, H., and G., Yin (1997) : "Stochastic Approximation Algorithms and Applications", Springer New-York.

[31] Lee, K., Kim, J., Cheng, S., and J., Shin (2017) : "Simplified Stochastic Feedforward Neural Networks", arXiv 170403188v1.

[32] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F, Halverson, J, Soljacic, M, Hou, T., and M., Tegmark (2024) : "KAN : Kolmogorov-Arnold Networks", arXiv 2404. 1975604.

[33] McKenzie, E. (1985) : "Some Simple Models for Discrete Variate Time Series", Water Resources Bulletin, 21, 645-650.

[34] McLeish, D. (1975) : "A Maximal Inequality and Dependent Strong Laws", Annals of Probability, 3, 829-839.

[35] Polyak, B., and A., Juditsky (1992) : "Acceleration of Stochastic Approximations by Averaging", SIAM Journal on Control and Optimization, 30, 838-853.

[36] Raydan, M. (1993) : "On the Barzilai and Borwein Choice of Step Length for the Gradient Method", IMA Journal of Numerical Analysis, 13, 321-326.

[37] Robbins, H., and S., Monro (1951) : "A Stochastic Approximation Method", The Annals of Mathematical Statistics, 39, 1327-1332.

[38] Rosenblatt, F. (1957) : "The Perceptron : A Perceiving and Recognizing Automation", Report 85-460-1, Cornell Aeronautical Laboratory.

[39] Rosenblatt, F. (1958) : "The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain", Psychological Review, 65, 386-408.

[40] Rumelhart, D., Hinton, G., and R., Williams (1986) : "Learning Internal Representations by Error Propagation", in Parallel Distributed Processing Vol 1. Explorations in the Microstructure of Cognition Foundations,

[41] Ruppert, D. (1988) : "Efficient Estimation from a Slowly Convergent Robbins-Monro Process", Technical Reports Cornell University, Operations Research and Industrial Engineering.

[42] Salakhutdinov, R., and G., Hinton (2009) : "Deep Boltzmann Machines", Artificial Intelligence and Statistics, PMLR5, 448-455.

[43] Steutel, F., and K., van Harn (1979) :"Discrete Analogues of Self-Decomposability and Stability", Annals of Probability, 7, 893-899.

[44] Twumasi, C., and J., Twumasi (2022) : "Machine Learning Algorithms for Forecasting and Backcasting Blood Demand Data with Missing Values and Outliers : A Study of Tema General Hospital of Ghana", International Journal of Forecasting, 38, 1258-1277.

[45] Venkatraman, A., Sun, W., Hebert, M., Bagnell, J., and B., Boots (2016) : "Online Instrumental Variable Regression with Applications in Online Linear System Identification", in Proceedings of the AAAI Conference on Artificial Intelligence, Vol 30, Cambridge MA : AAAI Press.

[46] Vrahatis, M;, Magoulas, G., and V., Piagianakos (2003) : "From Linear to Nonlinear Iterative Methods", Applied Numerical Mathematics, 45, 59-77.

[47] Yu, D., and Deng, L. (2015) : "Automatic Speech Recognition : A Deep Learning Approach", in Signals and Communication Technology, Springer, 317 pages.

Appendix 1

Extended Recursions

The aim of this appendix is to explain how to construct the α activation functions in the AFS neural network and introduce "intercepts" (biases) in the AFS model by analogy with the practice in MLP neural network. We first consider the case of decomposable multivariate activation, that allows to focus on the one-dimensional α. Then we give examples of non decomposable multivariate activation functions, that can allow for simultaneity and/or feedback effects in RNN. In the last subsection of this appendix, we provide the form of recursive equations for these extended versions including intercepts.

1.1 One dimensional activation with intercept.

The intercept effects can be introduced in two ways either i) in the transformation of Z_{j-1} variables (as in Example 1) and/or ii) in the definition of the underlying log-Laplace transform itself.
The extended version of the conditionally independent AFS network in Proposition 1 is :

$$\begin{aligned} & E[\exp(u'Z_j)|Z_{j-1}] \\ = \quad & \exp\left\{\sum_{l=1}^{L}[\alpha_{j-1,l}(u)(c_{l,j-1}Z_{j-1}+d_{l,j-1})+\beta_{j-1,l}(u)]\right\}, \end{aligned}$$

where the $\alpha_{j-1,l}, \beta_{j-1,l}$ are chosen ex-ante and the connection (or synoptic) weights $c_{l,j-1}$ and "biases" $d_{l,j-1}$ are parameters to be estimated.
To ensure that the quantity above is really a conditional Laplace transform, we can assume that :

i) any $\alpha_{j-1,l}$ is the log-Laplace transform of a univariate infinitely divisible distribution;[19]
ii) $c_{l,j-1}Z_{j-1}+d_{j-1,l}$ is nonegative;
iii) $\beta_{j-1,l}$ is any log Laplace transform.

These forms of transitions can be obtained from the transitions of Compound

[19] A distribution with log-Laplace transform $\alpha(u)$ is infinitely divisible if and only if $c\alpha(u)$, is also a log-Laplace transform for any $c > 0$.

Autoregressive process. For instance the Integer Autoregressive (INAR(1)) process [Al-Osh and Azaid (1987), Mc Kenzie (1985)] has such a transition with activation function $a(u) = \log[p\exp(u) + 1 - p]$, with $0 > p > 1$.

1.2 Nondecomposable multidimensional activation function with intercept.

The extension of the conditional Laplace transform is definition 1 ii) is :

$$E[\exp(u'Z_j)|Z_{j-1}] = \exp[\alpha_{j-1}(u)'[C_{j-1}Z_{j-1} + d_{j-1}] + \beta_{j-1}(u)],$$

where α_{j-1} is not necessarily of the form $\alpha_{j-1}(u) =$ diag $[\alpha_{j-1,l}(u_l)]$.

1.3 Recursive equations

The formulas in Proposition 2 become :

$$\begin{aligned} E[\exp(u'Y)|Z_j] &= \exp[B'_j(u;C)Z_j + b_j(u;C,d)], \\ E[\exp(u'Z_j)|Z_0] &= \exp[A'_j(u,C)Z_0 + a_j(u;C,d)], \end{aligned}$$

where the functions B_j, b_j, A_j, a_j satisfy the backward and forward recursions :

$$\begin{aligned} B_{j-1}(u;C) &= C'_{j-1}a_{j-1}[B_j(u;C)], \\ b_{j-1}(u;C,d) &= \alpha_{j-1}[B_j(u;C)]'d_{j-1} + \beta_{j-1}[B_j(u;C)] + b_j(u;C,d), \\ A_j(u;C) &= A_{j-1}[C'_{j-1}\alpha_{j-1}(u), C], \\ a_j(u;C,d) &= a_{j-1}[C'_{j-1}\alpha_{j-1}(u), C, d] + \alpha_{j-1}(u)'d_{j-1} + \beta_{j-1}. \end{aligned}$$

Note that the functions A_j depend on the connecting weights, but not on the intercept terms either d_j, or β_j

Proof : The proof is similar to the proof of Proposition 2. Let us for instance consider the backward recursion. We have :

$$
\begin{aligned}
& E[\exp(u'Y)|Z_{j-1}] \\
= \;& E\{E[\exp(u'Y)|Z_j]|Z_{j-1}\} \\
= \;& E\{\exp(B_j'(u;C)Z_j + b_j(u;C,d))|Z_{j-1}\} \\
= \;& \exp\{\alpha_{j-1}'[B_j(u;C][C_{j-1}Z_{j-1} + d_{j-1}] + \beta_{j-1}[B_j(u;C)] + b_j(u;C,d)\}.
\end{aligned}
$$

We deduce :

$$
\begin{aligned}
B_{j-1}(u,c) &= C_{j-1}'\alpha_{j-1}[B_j(u;C)], \\
\text{and} & \\
b_{j-1}(u;C,d) &= \alpha_{j-1}'[B_j(u,C)]d_{j-1} + \beta_{j-1}[B_j(u;C,d) + b_j(u;C,d).
\end{aligned}
$$

QED

Appendix 2

Recursion on Derivatives

The alternating optimization is simplified due to an appropriate decomposition of the conditional Laplace transform : $E[\exp(u'Y)|X] = \exp[B_0(u,C)'X]$, with respect to the subparameters $C_j, j = 0, \ldots, J$. Let us denote $\bar{C}_j = (C_j, \ldots, C_J), \underline{C_j} = (C_0, \ldots, C_j)$. By applying the iterated expectation theorem to conditioning with respect first to Z_j, next to Z_{j-1}, we get :

$$B_0(u,C) = A_{j-1}\{C'_{j-1}a_{j-1}[B_j(u,\bar{C}_j)], \underline{C_{j-2}}\}.$$

We deduce the derivative with respect to $\text{vec}(C_{j-1})$. We get :

$$\begin{aligned}
&\frac{\partial B_0(u,C)}{\partial(\text{vec } C_{j-1})'} \\
= \;& \frac{\partial A_{j-1}}{\partial u'}[C'_{j-1}a_{j-1}[B_j(u,\bar{C}_j), \underline{C_{j-2}}]Id_L \otimes a'_{j-1}[B_j(u,\bar{C}_j)] \\
= \;& \frac{\partial A_{j-1}}{\partial u'}[B_{j-1}(u,\bar{C}_{j-1}), \underline{C_{j-2}}]Id_L \otimes a'_{j-1}[B_j(u,\bar{C}_j)].
\end{aligned}$$

i) First-order derivatives

The expression above involves the derivatives of the functions $A_j(u, \underline{C_{j-1}})$ with respect to u. These derivatives can be computed recursively by a propagation algorithm from the equations :

$$A_j(u, \underline{C_{j-1}}) = A_{j-1}[C'_{j-1}a_{j-1}(u), \underline{C_{j-2}}].$$

We have :

$$\frac{\partial A_j(u, \underline{C_{j-1}})}{\partial u'} = \frac{\partial A_{j-1}}{\partial u'}[C'_{j-1}a_{j-1}(u), \underline{C_{j-2}}]C'_{j-1}\frac{\partial a_{j-1}(u)}{\partial u'}.$$

ii) **Second-order derivatives**

For some algorithms, especially when applying the quasi-maximum likelihood approach of section 4.1.2, and objective functions, it can be useful to derive the recursive formulas for second-order derivatives. To derive such recursions,

we have to introduce the elements A_{ij} of function A_j and the rows $c'_{j-1,k}$ of matrix C_{j-1}.
Then we have :

$$\frac{\partial A_{ij}(u,\underline{C_{j-1}})}{\partial u'} = \frac{\partial A_{i,j-1}}{\partial u'}[C'_{j-1}a_{j-1}(u),\underline{C_{j-2}}]\Sigma_{k=1}^{K}\left\{c'_{j-1,k}\frac{\partial a_{k,j-1}(u)}{\partial u'}\right\}.$$

We deduce :

$$\begin{aligned}
& \frac{\partial^2 A_{i,j}(u,\underline{C_{j-1}})}{\partial u \partial u'} \\
= \quad & \frac{\partial a'_{j-1}(u)}{\partial u}C_{j-1}\frac{\partial^2 A_{i,j-1}}{\partial u \partial u'}[C'_{j-1}a_{j-1}(u),\underline{C_{j-2}}]C'_{j-1}\frac{\partial a_{j-1}(u)}{\partial u'} \\
+ \quad & \frac{\partial A_{i,j-1}}{\partial u}[C'_{j-1}(u),\underline{C_{j-2}}]\sum_{k=1}^{K}\left\{c'_{j-1,k}\frac{\partial^2 a_{k,j-1}(u)}{\partial u \partial u'}\right\}.
\end{aligned}$$

Online Appendix 1

Approximation of Functions by Multilayer Perceptron Neural Network

We first review the Kolmogorov-Arnold representation theorem or superposition theorem, that states that every continuous function of p continuous variables can be represented by an appropriate composition of functions of one variable. We especially discuss modern variants of the superposition theorems known as universal approximation theorems, and their interpretation in terms of deterministic hidden layers. Then we discuss approximate superpositions representations based on multilayer perceptron (MLP) neural networks.

A.1 The Superposition Theorem

The theorem is usually written for continuous function defined on $[0,1]^n$. Any such function f can be written as :

$$f(x_1,\ldots,x_n)=\sum_{j=0}^{2n} g_j[\sum_{k=1}^{n}\varphi_{jk}(x_k)], \tag{a.1}$$

where φ_{jk} are continuous real functions on [0,1] and g_j continuous real functions on $I\!R$.
The initial proof of this result is due to Kolmogorov (1956), Arnold (1957), is nonconstructive and the representation is not unique.
Then the literature has looked for more specific representations with less functional parameters by considering identical g_j functions [Lorentz (1962)], inner functions φ_{jk} depending on a single function φ with appropriate shift in the argument [Sprecher (1993)], and extending the representation to other functions f. This leads to the following representation [Ismayilova and Ismailov (2023)] :

$$f(x_1,\ldots,x_n)=\sum_{j=0}^{2n} g[(\sum_{k=1}^{n}\lambda_k\varphi(x_k+a_j))+b_j], \tag{a.2}$$

λ_k, b_j are known, φ is a universal monotonic function, that is independent of function f, and g a one variable function depending on f. This representation exists also for discontinuous functions, and unbounded functions on $[0,1]^n$. Moreover, the representation is unique [Ismailov (2017)].

The interpretation of this representation in terms of feedforward deterministic neural network with two hidden layers is appealing and has been noted in Hecht-Nielsen (1987). The architecture of the two layer feedforward network (TLFN) is the following :

input layer : $x_1, \ldots, x_n$;

first-hidden layer : $z_{1,j,k} = \varphi(x_k + a_j), j = 0, \ldots, 2n, k = 1, \ldots, n.$

second hidden layer : $z_{2,j} = g[\sum_{k=1}^{n} \lambda_k z_{1,j,k} + b_j], j = 0, \ldots, 2n.$

output layer : $y = \sum_{j=0}^{2n} z_{2,j}.$

There has been a huge debate on the usefulness of this result since functions g, φ have no simple form, are rather difficult to find even numerically, and function g is wildly varying w.r.t. f, even if f is smooth [see Girosi, Poggio (1989), Kurkova (1991)].

A.2 Superposition with Multilayer Perception (MLP) Neural Network

The difficulties above have been circumvented by looking for superpositions with similar given transfer functions and approximate representations instead of exact representations.

A.2.1 Ridge functions and MLP models

The basic transformations are multivariate transformations of the form : $G(a_1x_1 + \ldots + a_nx_n + b), \equiv G(a'x + b)$, where G is a sigmoidal transformation from $I\!R$ to [0,1]. Then the approximations are obtained by superposing different layers with such transform at each node (neuron). More precisely. The single layer perceptron (SLP) neural network considers an approximation of the type :

$$f(x_1, \ldots, x_n) \simeq \sum_{j=1}^{J} c_j G(a_j' x + b_j), \tag{a.3}$$

where c_j, b_j, a_j are parameters.

The two layer perceptron (TLP) neural network is obtained by iterating the approximation above. It leads to :

$$f(x_1, \ldots, x_n) \simeq \sum_{j=1}^{J} c_j G[\sum_{k=1}^{K} c_{jk} G(a'_{jk} x + b_{jk}) + b_j], \tag{a.4}$$

and so on.
The SLP network has J nodes on the hidden layer. The TLP network KJ nodes in the first hidden layer and J nodes in the second hidden layer.

A.2.2 The approximation results

There exist different approximation results by MLP neural networks that involve the choice of function G, the number of layers and the number of nodes on each layer. The main results are given below.

Approximation 1 [Hornik et al. (1989)], Th. 2.3, Cybenko (1989), Th. 2].
For a given activation function G, the continuous function f can be uniformly approximated with error ε by a SLP neural network (a.3) with number of nodes J and parameters depending on ε.

This means that f can be well approximated with a single layer model, but a number of nodes that can be very large. Can we control for the number of nodes ?

Approximation 2 [Maiorov and Pinkus (1999), Th 4].
There exists a sigmoidal activation function G such that f can be uniformly approximated with error ε by a TLP neural network (a.4), with parameters depending on ε, but fixed number of nodes $J = 6n + 3, K = 3n$.

This results controls the number of nodes at each layer, but does not explain what is the right sigmoidal function to choose. It is often used as an argument for considering two layers only in (deterministic) deep learning. Note however that the number of nodes in the first layer is equal to $KJ + 18n^2 + 9n$ and becomes quickly rather large when the dimension n increases.

Approximation 3 : [Gripenberg (2003), Th. 2].
For a given activation function G, we can get an approximation of function f at level ε with a given number of nodes at each layer and a number of layers

depending on ε. Therefore there is a tradeoff between the number of nodes, the number of layers and the possibility to fix ex-ante the activation function G.

For practical use the approximation results do not explain how to fix the number of layers, the number of nodes and the activation function to get a good accuracy, that will depend also on function f.
Moreover the results are usually derived for functions defined on $[0,1]^n$. Some can be extended to functions defined on $\mathbb{R}^n$, but in this case the uniform approximation is on every compact on $\mathbb{R}^n$ and then can be very poor in the tails of function f.

Remark 1 : The continuity assumption of function f implies that the arguments $x_1, \ldots, x_n$ correspond to observed continuous variables and then the approximation results are not valid if some underlying variables are either qualitative (dummy variables), or count variables.

Remark 2 : There is a multiplicity of deterministic MLP networks. Indeed the initial variables $x_1, \ldots x_n$ can be replaced by transformed variables $x_1^*, \ldots, x_n^*$ in a one-to-one relationship with $x_1, \ldots, x_n$. For instance the MLP network can be written in $x_i^* = \log x_i, i = 1, \ldots, n$, instead of $x_i's$ (when the $x_i's$ are positive).

Remark 3 : Similarly, the MLP network can be written on a one-to-one transform of function f instead of function f itself. For instance let us assume that :

$$f(x_1, \ldots, x_n) = E(Y|X = x) \equiv p(x_1, \ldots, x_n).$$

where Y is a binary variable. Then $f = p$ takes values in [0,1]. The direct approximation (a.4) will induce constraints on parameters $c_j, j = 1, \ldots, J$, such that $c_j \geq 0, \forall j$, with $\sum_{j=1}^{J} c_j = 1$, that can be difficult to take into account in the learning (estimation) step. Instead the approximation (a.4) can be applied to the transformed function $G^{-1}[p(x_1, \ldots, x_n)]$ instead of being applied to $p(x_1, \ldots, x_n)$ itself. Equivalently the approximation (a.4) is replaced by :

$$p(x_1, \ldots, x_n) = G\{\sum_{j=1}^{J} c_j G[\sum_{k=1}^{K} c_{jk} G(a'_{jk} x + b_{jk}) + b_j]\}. \qquad \text{(a.5)}$$

Remark 4 : It is not known how to extend and use these approximation results for networks with stochastic transitions, or equivalently how to construct superposition approximations of the conditional Laplace transform $\psi(u, X)$ that are "uniform" and coherent with respect to argument u.

References

[1] Arnold, V. (1957) : "On Functions of Three Variables", Dokl, Akad. Nauk. SSSR, 114, 679-681.

[2] Cybenko, G. (1989) : "Approximation by Superpositions of a Sigmoidal Function", Math. Control Signals Systems, 2, 303-314.

[3] Farrell, M., Liang, T., and S., Misra (2021) : "Deep Neural Networks for Estimation and Inference", Econometrica, 89, 181-213.

[4] Girosi, F., and T., Poggio (1989) : "Representation Properties of Networks : Kolmogorov's Theorem is Irrelevant", Neural Comp., 1, 465-469.

[5] Gripenberg, G. (2003) : "Approximation by Neural Network with a Bounded Number of Nodes at Each Level", J. App. Theory, 123, 260-266.

[6] Guliyev, N., and V., Ismailov (2018) : "Approximation Capability of Two Hidden Layer Feedforward Neural Networks with Fixed Weights", Neurocomputing, 316, 262-269.

[7] Hecht-Nielsen, R (1987) : "Kolmogorov's Mapping Neural Network Existence Theorem", in Proceedings 1987 IEEE Int. Conf. on Neural Networks, IEEE Press, 3, 11-14.

[8] Hornik, K., Stinchcombe, M., and H., White (1989) : "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, 2, 359-366.

[9] Ismailov, V., (2017) : "On the Uniqueness of Representation by Linear Superpositions", Ukrainian Math. 3, 68, 1879-1883.

[10] Ismailov, V. (2021) : "Ridge Functions and Applications in Neural Networks", Mathematical Surveys and Monographs, 263, American Mathematical Society, 186 pp.

[11] Ismayilova, A., and V., Ismailov (2023) : "On the Kolmogorov Neural Networks", arXiv.

[12] Kolmogorov, A. (1956) : "On the Representation of Continuous Functions of Several Variables by Superpositions of Continuous Functions of a Small Number of Variables", Proceedings of the USSR Academy of Sciences, 108, 179-182.

[13] Kurkova, V. (1991) : "Kolmogorov's Theorem is Relevant", Neural Comp., 3, 617-622.

[14] Lorentz, G. (1962) : "Metric Entropy, Widths and Superpositions of Functions", American Mathematical Monthly, 69, 469-485.

[15] Maiorov, V., and A., Pinkus (1999) : "Lower Bounds for Approximation by MLP Neural Networks", Neurocomputing, 25, 81-91.

[16] Sprecher, D. (1965) : "On the Structure of Continuous Functions of Several Variables", Transactions of the American Mathematical Society, 115, 340-355.

[17] Sprecher, D. (1993) : "A Universal Mapping for Kolmogorov's Superposition Theorem", Neural Network, 6, 1089-1094.

Online Appendix 2

The Chain Rule and the Backpropagation

As mentioned in the text, some derivatives can be performed on functions satisfying the chain rule. In the MLPNN, this leads to the backward propagation algorithm introduced in Rumelhart, Hinton and Williams (1988) and precisely described for the LTSM in the appendix of Hochreiter and Schmidhuber (1997). Let us consider a network defined by :

$$\begin{aligned} Z_0 &= X && \text{,with } \dim Z_0 = m_0, \\ Z_1(\underline{c_1}, X) &= g_1(C_1'X), && \text{,with } \dim Z_1 = m_1, \\ Z_2(\underline{c_2}, X) &= g_2[C_2'Z_1(\underline{c_1}, X)] \\ &= g_2[C_2'g_1(C_1'X)], && \text{with } \dim Z_2 = m_2, \end{aligned}$$

up to $Z_j(\underline{c_J}, X) = g_J[C_J'Z_{J-1}(\underline{c_{J-1}}, X)]$ with $\dim Z_J = m_J$,
where $c_j = \text{ vec } C_j, j = 1, \ldots, J$. The functions $g_j, j = 1, \ldots, J$, are given and their Jacobian $\dfrac{\partial g_j(u_j)}{\partial u_j'}$ with dimension (m_j, l_j) has a closed form in practice.

Then the score appearing in the ASGD algorithm can require the derivatives of $Z_J(\underline{c_J}, X)$ with respect to $c_j = \text{ vec } C_j, j = 1, \ldots, J$. Due to the chain rule, we get the following result :

Proposition : We have :

$$\begin{aligned} &\frac{\partial Z_J(\underline{c_J}, X)}{\partial c_j'} \\ = \; &\frac{\partial g_J}{\partial u_J'}[C_J'Z_{J-1}(\underline{c_{J-1}}, X)]C_J'\frac{\partial g_{J-1}}{\partial u_{J-1}'}[C_{J-1}'Z_{J-2}(\underline{c_{J-2}}, X)]C_{J-1}' \\ &\ldots \frac{\partial g_j}{\partial u_j'}[C_j'Z_{j-1}(\underline{c_{j-1}}, X)]Id_{l_j} \otimes Z_{j-1}'(\underline{c_{j-1}}, X), j = 1, \ldots, J, \end{aligned}$$

where $\otimes$ denotes the Kronecker product.

This formula leads to a forward-backward (or backpropagation) algorithm. For given values of the $C'_j s$, the successive values of the $Z_j(\underline{c_j}, X)$ are computed forward by :

$$Z_j(\underline{c_j}, X) = g_j[C'_j Z_{j-1}(\underline{c_{j-1}}, X)], j = 1, \ldots, J.$$

and then the derivatives $\dfrac{\partial Z_J(\underline{c_J}, X)}{\partial c'_j}, j = J, J-1 \ldots$ computed backward by the formula in the Proposition above.

The formula is written without assuming the conditional independence between the neurons of a same layer and a universal activation function. It can be easily written with these special restrictions.

Online Appendix 3

The Architecture of Long Short Term Memory (LSTM) Cell

We provide below the standard architecture of an LSTM cell [Hochreiter and Schmidhuber (1997)]. It is given with the notations of the machine learning literature with variables of a same dimension denoted $x_t, i_t, f_t, c_t, \emptyset_t, h_t, y_t$, three types of activation functions : the logistic function σ, the hyperbolic tangent : $tanh$, and the linear one. The connecting weights are denoted W with appropriate indexes and the operator $\bullet$ means the componentwise product of two vectors. At the beginning of period t, the observed inputs are exogenous inputs x_t and lagged summaries h_{t-1}, c_{t-1}. The deterministic state space model has the following structure :

$$
\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{ih}h_{t-1} + W_{ic}c_{t-1} + b_i), && (i)\\
f_t &= \sigma(W_{fx}x_t + W_{fh}h_{t-1} + W_{fc}c_{t-1} + b_f), && (ii)\\
c_t &= f_t \bullet c_{t-1} + i_t \bullet \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_v), && (iii)\\
o_t &= \sigma(W_{ox}x_t + W_{oh}h_{t-1} + W_{oc}c_t + b_\sigma), && (iv)\\
h_t &= o_t \bullet \tanh(c_t), && (v)\\
y_t &= W_{yh}h_t + b_y. && (vi)
\end{aligned}
$$

In the usual terminology of neural networks, this system has 5 layers. The first layer (input layer), with two neurons is described by (i) and (ii). The second layer with dim c neurons corresponds to (iii). Similarly the third and fourth layers have one neuron and corresponds to (iv) and (v), respectively. Then (vi) is the output layer.

The LSTM literature has introduced its own terminology that explains the notations for the variables, but can be confusing. The first state equation (i) is called the input gate, the second (ii) the forget gate eliminates the information that is no longer relevant in the cell state. The equations (iii) are the cell activation equations and the equation (iv) is the output gate.

The LSTM model has been initially introduced to allow for different degrees

of persistence (short vs long persistence) in the nonlinear dynamics of h_t (and of y_t). Note that this feature can also be reached with the Markov AFS neural network of Section 5.2, without fixing ex-ante the state variables creating the long memory features.
Such LSTM cells are introduced in R-software libraries as Keras and Tensorflow [Arnold (2019)]. Look at the warning in implementing such softwares in Twumasi and Twamasi (2022), Section 3.3.8, "due to the numerous modelling steps from time series differencing, data transformations, inverse scaling after obtaining predictions".

References

[1] Hochreiter, S., and J., Schmidhuber (1997) : "Long Short Term Memory", Neural Computation, 9, 1735-1780.

[2] Twumasi, C., and J., Twumasi (2022) : "Machine Learning Algorithms for Forecasting and Backcasting Blood Demand Data with Missing Values and Outliers : A Study of Tema General Hospital of Ghana", International Journal of Forecasting, 38, 1258-1277.

Online Appendix 3

Acronyms

This machine learning literature is introducing and using a lot of acronyms. The ones used in our paper are given below with their meaning. Some acronyms were used before in statistics with other meanings. These double meanings can be confusing and are systematically mentioned.

AFS (network)	Affine Feedforward Stochastic
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASGD (algorithm)	Average Stochastic Gradient Descent
ASR	Automatic Speech Recognition
BM	Boltzmann Machine
BP (algorithm)	Back Propagation
BPTT (algorithm)	Back Propagation Through Time
CoD	Curse of Dimensionality
DNN	Deep Neural Network
ELBO	Evidence Lower Bound
GD (algorithm)	Gradient Descent
GDP	Gross Domestic Product
GMM	i) Gaussian Mixture Model
	ii) Generalized Method of Moments
HMM	Hidden Markov Model
KAN	Kolmogorov Arnold Network
LSTM	Long Short Term Memory
LT	Laplace Transform
MGF	Moment Generating Function
ML	i) Machine Learning
	ii) Maximum Likelihood
MLP (network)	Multi Layer Perceptron
NN	Neural Network
RBM	Restricted Boltzmann Machine
RELU	Restricted Linear Unit
RNN	Recurrent Neural Network
SGD (algorithm)	Stochastic Gradient Descent
SLP	Single Layer Perceptron
SNN	Stochastic Neural Network
TLU	Threshold Logic Unit.