26th International Congress on Modelling and Simulation, Adelaide, SA, Australia, 30 Nov–4 Dec 2025
mssanz.org.au/modsim2025

Generating Initial Values using Artificial Neural Network

F. Chan [a]

[a]*School of Accounting, Economics and Finance, Curtin University, GPO BOX U1987, Perth, Western Australia, 6845*
Email: F.Chan@curtin.edu.au

Abstract: The estimation of econometric models generally relies on M-estimator. Well known examples include various least squares estimators (LS), maximum likelihood estimator (MLE) and generalised method of moments (GMM) estimator. The computation of these estimators requires the solution of the following optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg \underset{\boldsymbol{\theta}}{\text{opt}}\ L(\boldsymbol{\theta}; \mathbf{Z}) \tag{1}$$

where $\boldsymbol{\theta}$ is a $K \times 1$ parameter vector that one wishes to estimate based on the input data $\mathbf{Z}$, a $N \times (K+1)$ data matrix, by minimising or maximising the objective function L. The optimisation as defined in Equation (1) often does not have closed form solution. In such case, the solution of the optimisation require optimisation algorithms, such as gradient decent, which is an iterative procedure of the form:

$$\hat{\boldsymbol{\theta}}_{n+1} = \hat{\boldsymbol{\theta}}_n + \eta \nabla L(\hat{\boldsymbol{\theta}}_n) \tag{2}$$

where η denotes the step. In order to initiate the algorithm, the initial values of the parameter vector $\hat{\boldsymbol{\theta}}_0$ must be provided by the researchers. It is well known that most gradient descent type algorithms would require this initial value to be "close" to the global optimum but yet, there is no general approach on how to effectively identify initial values. The objective of this paper is to examine if artificial neural network (ANN) can be used to generate robust initial values to ensure convergence of gradient descent type algorithms.

The basic idea is to simulate the data from a specific econometric model using a range of parameter vectors and then use the simulated data to develop ANN to produce the vector $\hat{\boldsymbol{\theta}}_0$. In order words, the simulated data will be used as the input of the ANN while the parameter vector $\boldsymbol{\theta}$ will be used as the target for purpose of training the ANN.

The paper applies this approach to two common econometrics models namely, the logit model and the Smooth Transition Autoregressive model. Initial evidence suggests that ANN can out-perform the conventional method of obtaining initial values of these models in the context of number of iterations required to achieved convergence.

Keywords: *M-estimator, initial values, optimisation, artificial neural network, machine learning*

1 INTRODUCTION

In many econometric and statistic applications, the estimation of a model, or more precisely, the estimation of the unknown parameters in a model is often achieved by solving the following optimisation problem:

$$\hat{\boldsymbol{\theta}} = \arg \underset{\boldsymbol{\theta}}{\text{opt}}\ L(\boldsymbol{\theta}; \mathbf{Z}) \tag{3}$$

where $\boldsymbol{\theta}$ denotes the unknown vector of parameters of size K, $L(\circ) : \mathbb{R}^K \to \mathbb{R}$ denotes the objective function conditional on the input $\mathbf{Z}$, which presumed to be a $N \times K + p$ matrix, with $p \geq 1$. This setting relates to the family of M-estimator, which generality was first discussed in Huber (1964) and the three main estimators in econometrics namely, Least Squares Estimator, (Quasi) Maximum Likelihood Estimator and Generalized Method of Moments, can all be viewed as a special cases of this family. While in some cases, the estimator, or the solution of Equation (3), can be expressed as a function of the input matrix $\mathbf{Z}$, in general, the solution cannot be expressed as a closed-form solution and must rely on iterative algorithms.

Most of these algorithm takes the following forms:

$$\hat{\boldsymbol{\theta}}_{n+1} = \hat{\boldsymbol{\theta}}_n + g(\mathbf{Z}, \hat{\boldsymbol{\theta}}_n) \tag{4}$$

where $\hat{\boldsymbol{\theta}}_n$ denotes the value of $\hat{\boldsymbol{\theta}}$ at the n^{th} step of the algorithm. Generally speaking, the idea is that $\hat{\boldsymbol{\theta}} = \hat{\boldsymbol{\theta}}_n$ as $n \to \infty$. In practice, the algorithm will stop when $|\hat{\boldsymbol{\theta}}_{n+1} - \hat{\boldsymbol{\theta}}_n| < \epsilon$ for some pre-defined tolerance level, ϵ. The function, $g(\mathbf{Z}, \hat{\boldsymbol{\theta}}_n)$, is used to iteratively update $\hat{\boldsymbol{\theta}}_n$ towards $\hat{\boldsymbol{\theta}}$. One of the most well known cases is the gradient descent class of algorithms which takes the form

$$\hat{\boldsymbol{\theta}}_{n+1} = \hat{\boldsymbol{\theta}}_n + \eta \nabla L(\hat{\boldsymbol{\theta}}_n; \mathbf{Z}) \tag{5}$$

where η is the step size and $\nabla L(\hat{\boldsymbol{\theta}}_n; \mathbf{Z})$ denotes the gradient vector L with respect to $\hat{\boldsymbol{\theta}}$ evaluated at $\hat{\boldsymbol{\theta}}_n$.

In all these cases, the vector of initial values, $\hat{\boldsymbol{\theta}}_0$, is required to start the algorithm. It should not be surprised that the value of n is small if $\hat{\boldsymbol{\theta}}_0$ is sufficiently 'close' to $\hat{\boldsymbol{\theta}}$. Thus, the choice of $\hat{\boldsymbol{\theta}}_0$ will directly affect the computation time of $\hat{\boldsymbol{\theta}}$. More importantly, if $\hat{\boldsymbol{\theta}}_0$ is too 'far' away from $\hat{\boldsymbol{\theta}}$, then it is possible that $\hat{\boldsymbol{\theta}}_n$ will deviate from $\hat{\boldsymbol{\theta}}$ as n increases. This is well known for Newton-Raphson type algorithms or if the objective functions have multiple local optima. See, for examples, Shanno (1970) and Burden & Faires (1993) for further discussions.

While the importance of initial values is widely acknowledged, there is a lack of consistency in the generation of appropriate initial values in practice. It is important to note that while some of these models are challenging to estimate i.e., solution to Equation (3) is difficult to obtain, they can be simulated efficiently. This means one can simulate large amount of synthetic data based on different parameter vectors and these data can then be used to train various machine learning models with the aim to produce appropriate initial values i.e., parameter vector that are 'close' to the true parameter vector. Among many machine learning algorithms, artificial neural network (ANN) appears to be a strong candidate for this task due to its property as an universal approximator as demonstrated in Cybenkot (1989), Hornik et al. (1990).

This paper will examine the performance of ANN to generate more efficient initial values for two well known models namely, the logit model and the Smooth Transition Autoregressive (STAR) model of Terasvirta (1994). The paper developed an ANN for each of the two models and then train the ANNs using simulated data over different parameter vectors. The trained ANNs will then be used to generate initial values for these models. The performance of these ANNs will then be examined using Monte Carlo experiments. Results suggest that the initial values generated by ANNs can reduce the number of iterations for both models in the context of BFGS algorithm as proposed in Broyden (1970) and see also Fletcher (1970).

The organisation of the paper is as follows: Section 2 introduces the specification of the ANNs for each of the two models. The specification of the training routines will also be discussed. The performance of the ANNs will be examined in Section 3 via Monte Carlo simulations and Section 4 will contain some concluding remarks.

2 METHODOLOGY

This section introduces the specification of each of the two ANNs to generate initial values for the toy models namely, logit and STAR models. The section first introduces the two toy models and defines their maximum likelihood estimators in the form of optimisation problems. This is followed by the specification of each of the two ANNs designed to generate appropriate values to compute the MLE using BFGS algorithm.

2.1 TOY MODELS

The two toy models considered in this paper are the logit model and the STAR model. Let y_i be a binary variable which takes on 0 or 1. Let $\mathbf{x}_i$ be a $K \times 1$ vector of covariates (predictors) for $i = 1 \dots N$ then the logit model is defined as

$$\Pr(y_i = 0) = \frac{\exp\left(\mathbf{x}_i'\boldsymbol{\theta}\right)}{1 + \exp\left(\mathbf{x}_i'\boldsymbol{\theta}\right)}. \tag{6}$$

Given the data $\mathbf{Z} = \{(y_i, \mathbf{x}_i)\}_{i=1}^N$, the MLE of the logit model is the solution to the following maximisation problem:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} y_i \log\left(\frac{\exp\left(\mathbf{x}_i'\boldsymbol{\theta}\right)}{1 + \exp\left(\mathbf{x}_i'\boldsymbol{\theta}\right)}\right) + (1 - y_i) \log\left(\frac{1}{1 + \exp\left(\mathbf{x}_i'\boldsymbol{\theta}\right)}\right). \tag{7}$$

The second toy model is the Smooth Transition Autoregressive (STAR) model of Terasvirta (1994), which is defined as

$$y_t = \left(\phi_{10} + \phi_{11} y_{t-1}\right) G(y_{t-1}; \gamma, c) + \left(\phi_{20} + \phi_{21} y_{t-1}\right) \left[1 - G(y_{t-1}; \gamma, c)\right] + \varepsilon_t \quad \varepsilon_t \sim D(0, \sigma^2)\ t = 1, \dots, T \tag{8}$$

where $G : \mathbb{R} \to [0, 1]$ denotes the transition function. Popular choice includes the logistic function (LSTAR), $G(y_{t-1}) = \left[1 + \exp\left(-\gamma(y_{t-1} - c)\right)\right]^{-1}$, and exponential function (ESTAR), $G(y_{t-1}) = 1 - \exp(-\gamma(y_{t-1} - c)^2)$. The (Quasi) Maximum Likelihood Estimator for STAR model under normality is defined to be

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathbf{y}) \equiv -\frac{T}{2} \log \sigma^2 - \frac{1}{2} \sum_{t=2}^{T} \frac{\varepsilon_t^2}{\sigma^2} \tag{9}$$

where $\varepsilon_t = y_t - \left(\phi_{10} + \phi_{11} y_{t-1}\right) G(y_{t-1}; \gamma, c) + \left(\phi_{20} + \phi_{21} y_{t-1}\right) \left[1 - G(y_{t-1}; \gamma, c)\right]$ and $\boldsymbol{\theta} = \left(\phi_{10}, \phi_{11}, \phi_{20}, \phi_{21}, \gamma, c, \sigma^2\right)'$. For further details on the structural and statistical properties of STAR models, see Terasvirta (1994), Dijk et al. (2002), Chan & McAleer (2002, 2003).

The computation of solutions to Equations 7 and 9 often rely on optimization routines based on BFGS or Neader-Mead (Nelder & Mead (1965)) types algorithms, which require initial values to initiate the algorithms. Conventionally, the initial values for logit is based on the OLS estimator i.e., $\hat{\boldsymbol{\theta}}_0 = \left(\mathbf{X}'\mathbf{X}\right)^{-1} \mathbf{X}'\mathbf{y}$ where $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)'$ and $\mathbf{y} = (y_1, \dots, y_N)'$.

In terms of the initial values for Logistic STAR model, it also relies on the OLS estimator on a linearised version of the model. Specifically, it replaces the transition function with its fourth order Taylor approximation around $\gamma = 0$ and $c = 0$. The choice of the order is not arbitrary as the coefficients of the fourth order Taylor polynomial leads to exactly identification of the initial parameter vector of the LSTAR model from its linearised version. For details see Terasvirta (1994).

2.2 ANN SPECIFICATION

ANN has been one of the most powerful and developed techniques in the literature of machine learning. In its basic form, ANN can be interpreted as a nonlinear approximation to a function through compositions of many nonlinear functions. Such structure, under certain conditions, can be shown to be an *universal approximator*

Model	Specifications	$\mathbf{F}_1$	$\mathbf{F}_2$	$\mathbf{F}_3$	$\mathbf{F}_4$	$\mathbf{F}_5$
logit	$\alpha_{i,j}$ $\mathbb{D}^{i-1} \to \mathbb{D}^i$	$\sigma(x)$ $\mathbb{R}^{K+1} \to \mathbb{R}^{10}$	CELU $\mathbb{R}^{10} \to \mathbb{R}^{10}$	$\sigma(x)$ $\mathbb{R}^{10} \to \mathbb{R}^{10}$	$i(x)$ $\mathbb{R}^{10} \to \mathbb{R}^{K}$	
STAR	$\alpha_{i,j}$ $\mathbb{D}^{i-1} \to \mathbb{D}^i$	$\sigma(x)$ $\mathbb{R}^{N} \to \mathbb{R}^{10}$	CELU $\mathbb{R}^{10} \to \mathbb{R}^{20}$	$\sigma(x)$ $\mathbb{R}^{20} \to \mathbb{R}^{20}$	RELU $\mathbb{R}^{20} \to \mathbb{R}^{7}$	$i(x)$ $\mathbb{R}^{7} \to \mathbb{R}^{7}$

Table 1. ANN Specifications of Logit and STAR

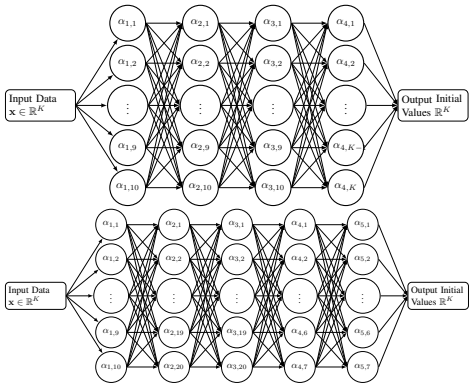

Figure 1. ANN for Generating Initial Values for Logit and STAR Models

that can be used to approximate a wide range of unknown functions arbitrarily well. Mathematically, this can be expressed as

$$\hat{\boldsymbol{\theta}}_0 = \mathbf{F}_m \circ \mathbf{F}_2 \circ \ldots \circ \mathbf{F}_1(\mathbf{Z}) \tag{10}$$

where $\mathbf{F}_i : \mathbb{D}^{i-1} \to \mathbb{D}^i$ where $\mathbb{D}^i$ denotes the range of the function $\mathbf{F}_i$. This implies that the domain of $\mathbf{F}_i$ is the range of $\mathbf{F}_{i-1}$.

Figure 1 provides the graphical representations of the two ANNs. They differ in terms of both the number of layers i.e., m and the number of nodes in each layer i.e., $\mathbb{D}^i$ for all i. In this paper, $\mathbf{F}_i(\mathbf{Z}) = \mathbf{F}_i\left(\alpha_{i,1}(\mathbf{w}'_{i,1}\mathbf{x}_i + b_{i,1}), \ldots, \alpha_{i,N_i}(\mathbf{w}'_{i,N_i}\mathbf{x}_i + b_{i,N_i})\right)$ for $i = 1, \ldots, m$ and N_i denotes the number of nodes in layer i. In the present context, N_i is also the dimension of the range of $\mathbf{F}_i$, that is $\mathbf{F}_i : \mathbb{R}^{N_{i-1}} \to \mathbb{R}^{N_i}$. In terms of the activation functions, $\alpha_{i,j}$, this paper utilises four well known functions in the literature, namely, *Continuously Differentiable Exponential Linear Unit* (CELU),

$$CELU(X) = \begin{cases} x & x \geq 0 \\ \exp(x) - 1 & x < 0 \end{cases}$$

Rectified Linear Unit (RELU), $RELU(x) = \max(0, x)$, *sigmod function*, $\sigma(x) = (1 + \exp(-x))^{-1}$ and the *identity function*, $i(x) = x$. In both logit and STAR cases, the activation functions are specified to be the same within each layers but can be different across different layers. Table 1 contains the specification of each ANN and the activation functions within each layer.

The training of both ANNs i.e., determination of $\mathbf{w}_{i,j}$ and $b_{i,j}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, N_i$, follows the steps below:

Step 1. Set the number of replication, R.

Step 2. Set $\boldsymbol{\theta}_r$ by randomly drawing from a multivariate uniform distribution.

Step 3. Simulate data of N observations based on $\boldsymbol{\theta}$.

Model	True	ANN	Conventional
Logit $K = 5$	14.88	16.21	18.44
Logit $K = 7$	18.43	21.57	22.39
STAR	426.85	1029.29	3952.13

Table 2. Average number of iterations based on different generation of initial values

Step 4. Update $\mathbf{w}_{i,j}$ and $b_{i,j}$ using the simulated data with $\boldsymbol{\beta}_r$ being the target.

Step 5. Set $r = r + 1$ and repeat Steps 2 to 4 until $r = R$.

For both models $R = 5000$ with $N = 500$. In the case of logit, this paper considers the case $K = 5$ i.e., the logit model contains five covariates. The paper also considers the case of using the trained ANN with $K = 5$ to generate the initial values when $K = 7$. The idea is to randomly select five covariates at time to generate the initial values for the associated coefficients and repeat the procedures until a minimum number of initial value for each coefficient has been obtained. The final initial value vector can then be obtained by taking the average of all the initial values for each coefficient.

In order to examine the performance of the ANN, a set of simulated data will be generated based on a fixed parameter vector. These data will then be used to estimate the parameter vector using the BFGS algorithm under three different sets of initial values namely, the true parameter vector, the ANN generated initial values and the initial values obtained by using the conventional method. The performance of each initial values will be examined based on the number of iterations until convergence. The number of replication is 5000.

3 RESULT

Table 2 contains the average number of iterations until convergence under different methods of generating initial values, while Figures 2 to 4 contain the histograms for the distribution of iterations across the three different initial values generation schemes. As shown in Table 2 and Figure 2, there is some evidence to suggest that the ANN can help to identify more efficient initial values for purpose of computing the maximum likelihood estimates. As expected, using the true parameter vector as initial values should be the most efficient on average but ANN does seem to provide improvement over the conventional method in this case.

The advantage of ANN seems to have diminished when $K = 7$. While it is still better than the conventional method, the improvement seems marginal. This may indicate that the ANN needs to be trained for specific value of K and/or a more sophisticated algorithm is required to generate the initial value with $K = 7$ based on the ANN that was trained under $K = 5$.

The result for STAR model is perhaps the most fascinating. While on average the ANN offers improvement over the conventional method as shown in Table 2, the distributions of iteration tell an interesting story as shown in Figure 4. It would appear that using the true value is more robust than the other two methods but ANN often offer a lower number of iterations than even the true value. However, it can sometimes generate initial values that do not need to convergence. The conventional method, however, performs poorly by comparison, with many initial values do not lead to convergence within the iteration limit. Note that this is not a criticism of the conventional method, but rather, a reflection on the challenge in obtaining maximum likelihood estimates for the STAR model, which is well known in the literature.

4 CONCLUSION

This paper explored the potential benefits in applying ANN to generate initial values for purposes of obtaining maximum likelihood estimates for nonlinear models. The initial results suggests that the ANN can offer significant improvement over conventional method in terms of number of iterations before convergence. However, these results may not be robust and further investigation is required to understand the condition in which ANN can identify appropriate initial values. It is also worth highlighting that the ANN developed in this paper are specific to the number of observations and the number of covariates, or lag order. The results showed that using such specific ANNs for model with different specifications may reduce their benefits. This also requires further investigation. Overall, however, the paper obtained sufficient evidence to suggest that this could be a potentially useful avenue of research.

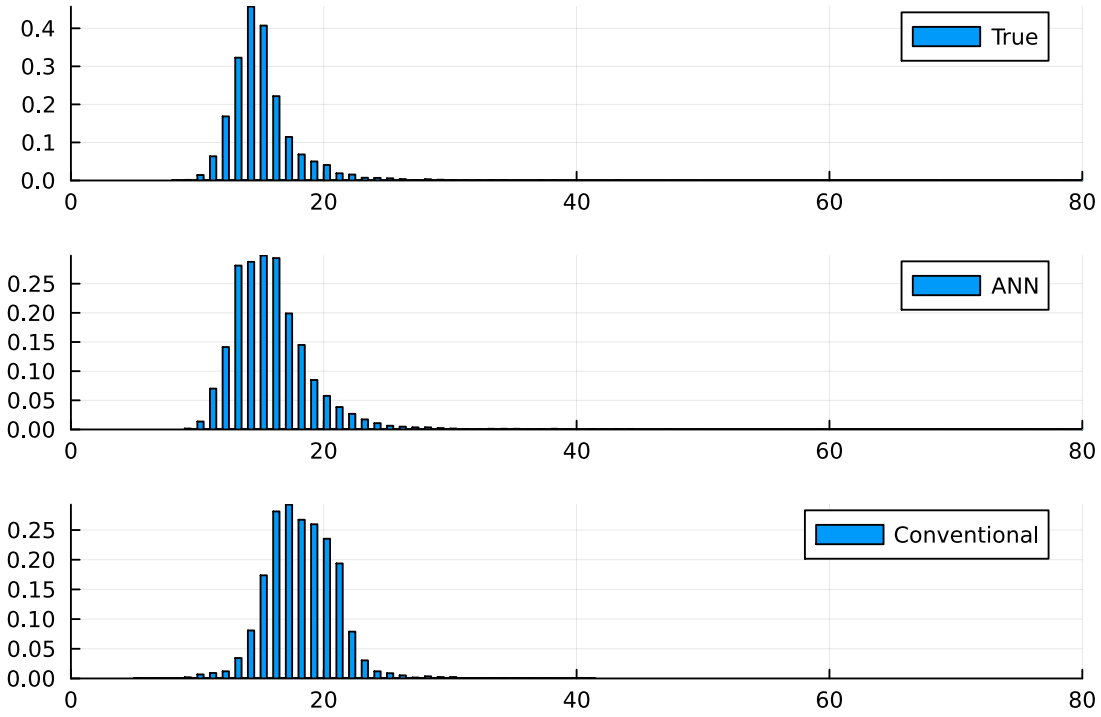

Figure 2. Distributions of iterations from different initial values: Logit $K = 5$

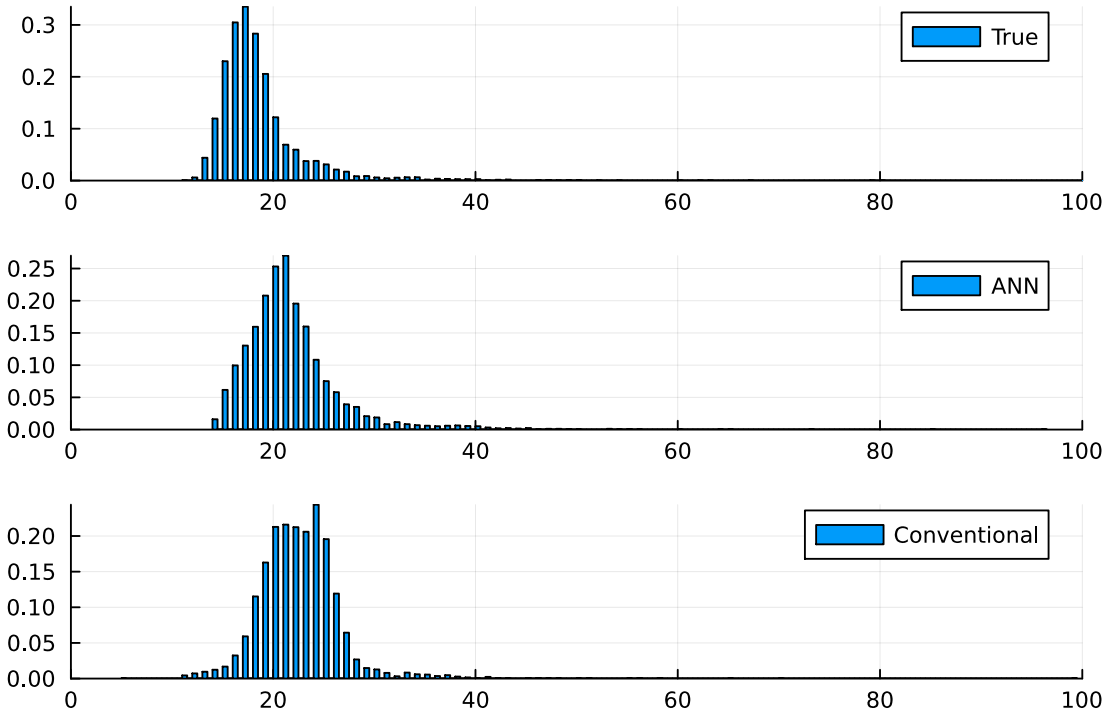

Figure 3. Distributions of iterations from different initial values: Logit $K = 7$

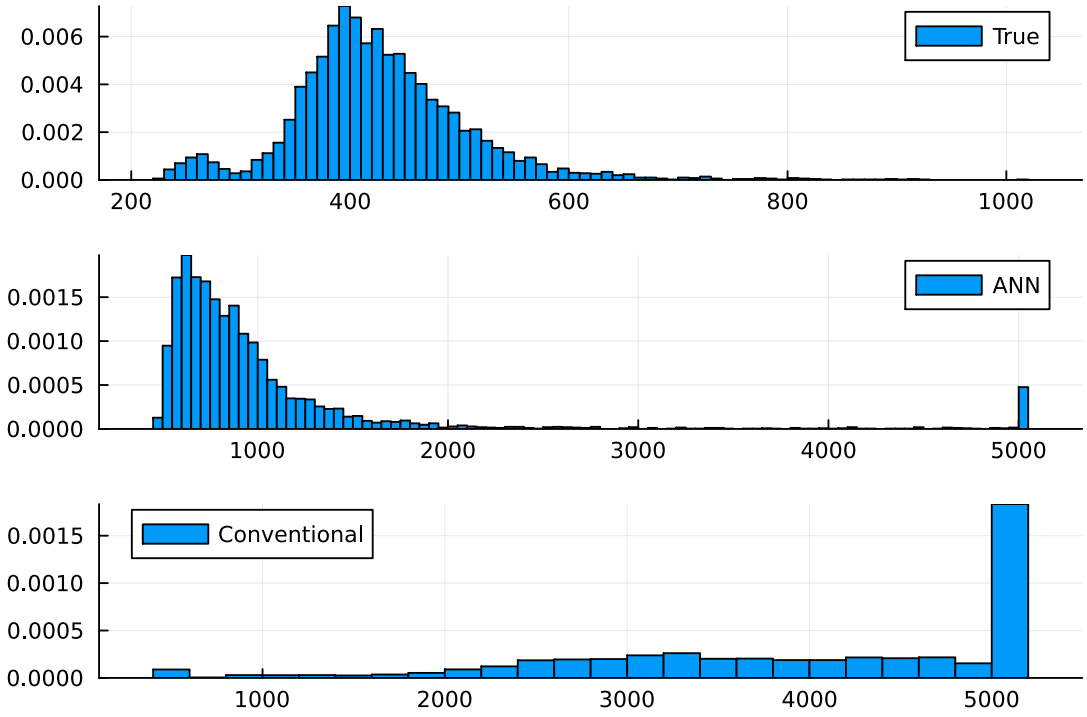

Figure 4. Distributions of iterations from different initial values: STAR

References

Broyden, C. G. (1970), 'The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations', *IMA Journal of Applied Mathematics* **6**(1), 76–90.
URL: *https://academic.oup.com/imamat/article-lookup/doi/10.1093/imamat/6.1.76*

Burden, R. & Faires, J. (1993), *Numerical Analysis*, Fifth edn, PWS Publishing Compnay.

Chan, F. & McAleer, M. (2002), 'Maximum likelihood estimation of STAR and STAR-GARCH models: Theory and Monte Carlo evidence', *Journal of Applied Econometrics* **17**(5).

Chan, F. & McAleer, M. (2003), 'Estimating smooth transition autoregressive models with GARCH errors in the presence of extreme observations and outliers', *Applied Financial Economics* **13**(8).

Cybenkot, G. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals, and Systems* **2**, 303–314.

Dijk, D. v., Teräsvirta, T. & Franses, P. H. (2002), 'SMOOTH TRANSITION AUTOREGRESSIVE MODELS — A SURVEY OF RECENT DEVELOPMENTS', *Econometric Reviews* **21**(1), 1–47. Publisher: Taylor & Francis.
URL: *http://www.tandfonline.com/doi/abs/10.1081/ETC-120008723*

Fletcher, R. (1970), 'A new approach to variable metric algorithms', *The Computer Journal* **13**(3), 317–322.

Hornik, K., Stinchcombe, M. & White, H. (1990), 'Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks', *Neural Networks* **3**(5), 551–560.
URL: *https://linkinghub.elsevier.com/retrieve/pii/0893608090900056*

Huber, P. J. (1964), 'Robust Estimation of a Location Parameter', *The Annals of Mathematical Statistics* **35**(1), 73–101.

Nelder, J. A. & Mead, R. (1965), 'A Simplex Method for Function Minimization', *The Computer Journal* **7**(4), 308–313.
URL: *https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.4.308*

Shanno, D. F. (1970), 'Conditioning of Quasi-Newton Methods for Function Minimization', *Mathematics of Computation* **24**(111), 647–656.

Terasvirta, T. (1994), 'Specification, Estimation, and Evaluation of Smooth Transition Autoregressive Models', *Journal of the American Statistical Association* **89**(425), 208–218.

