

Jahn, Malte

Working Paper

Artificial neural network regression models: Predicting GDP growth

HWWI Research Paper, No. 185

Provided in Cooperation with:

Hamburg Institute of International Economics (HWWI)

Suggested Citation: Jahn, Malte (2018) : Artificial neural network regression models: Predicting GDP growth, HWWI Research Paper, No. 185, Hamburgisches WeltWirtschaftsInstitut (HWWI), Hamburg

This Version is available at:

<https://hdl.handle.net/10419/182108>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.



Hamburg Institute
of International
Economics

Artificial neural network regression models: Predicting GDP growth

Malte Jahn

HWWI Research

Paper 185

Corresponding author:

Malte Jahn

Hamburg Institute of International Economics (HWWI)

Oberhafenstr. 1 | 20097 Hamburg, Germany

Telephone: +49 (0)40 340576-351 | Fax: +49 (0)40 340576-150

jahn@hwwi.org

HWWI Research Paper

Hamburg Institute of International Economics (HWWI)

Oberhafenstr. 1 | 20097 Hamburg, Germany

Telephone: +49 (0)40 340576-0 | Fax: +49 (0)40 340576-150

info@hwwi.org | www.hwwi.org

ISSN 1861-504X

Editorial Board:

Prof. Dr. Henning Vöpel

Dr. Christina Boll

© by the author | September 2018

The author is solely responsible for the contents which do not necessarily represent the opinion of the HWWI.

Artificial neural network regression models: Predicting GDP growth

Malte Jahn

September 2018

Abstract

Artificial neural networks have become increasingly popular for statistical model fitting over the last years, mainly due to increasing computational power. In this paper, an introduction to the use of artificial neural network (ANN) regression models is given. The problem of predicting the GDP growth rate of 15 industrialized economies in the time period 1996-2016 serves as an example. It is shown that the ANN model is able to yield much more accurate predictions of GDP growth rates than a corresponding linear model. In particular, ANN models capture time trends very flexibly. This is relevant for forecasting, as demonstrated by out-of-sample predictions for 2017.

Keywords: neural network, forecasting, panel data

JEL codes: C45, C53, C61, O40

1 Introduction

The main goal of the paper is to demonstrate the capabilities of artificial neural network (ANN) regression models. In particular, the paper shall encourage researchers to use ANN models for regression analysis. It shall help to overcome two existing barriers to the use of ANN models: the lack of theoretical understanding on the one hand and the inability to actually implement the model on the other hand. Therefore, the theoretical framework is discussed along with the precise algorithm used to train the ANN. The academic usefulness is demonstrated by the ability of the ANN regression model to produce more accurate predictions of GDP growth rates of 15 developed countries between 1996 and 2016 than a corresponding linear model. Regarding forecasts, the flexible time trend is shown to be a major advantage of the ANN model.

There have been previous applications of ANN models for prediction/forecasting of GDP or similar economic variables.

Tkacz (2001) does an analysis of Canadian GDP growth rates, finding that his ANN model is not superior to a linear model at the quarter-to-quarter forecasting horizon but is superior at the year-to-year forecasting horizon. He believes that this is due to a non-linear influence of monetary variables which seems to be more relevant in the longer run.

Chuku et al. (2017) analyze GDP data for South Africa, Nigeria and Kenya (individually) and find that, in many of the considered specifications, the ANN model is superior to structural econometric models and ARIMA models. Their main explanation is that developing economies are "exposed to potential chaotic influences from commodity prices, external factors, and even political economy factors" (Chuku et al., 2017, p. 25). The flexibility of ANN models seems to pay off in this regard.

Kock and Teräsvirta (2014) deal with the monthly forecasting of industrial production and unemployment in industrialized economies during a crisis. They find that their autoregressive neural network does not perform significantly better than the linear autoregressive model.

Feng and Zhang (2014) investigate the evolution of the tertiary production in the Chinese province of Zhejiang between 1971 and 1997. Their result is that the ANN outperforms the alternative "gray" forecasting model. Since they don't consider classical autoregressive models as a reference, their results are a bit more difficult to interpret.

It will be shown in the following sections that the results derived in this very paper come closest to that of Tkacz (2001), both regarding the considered problem (annual GDP data in developed economies) as well as the magnitude by which the ANN model outperforms linear models.

2 The data set

The GDP panel data set includes the following 15 countries, in alphabetical order: Austria, Belgium, Denmark, Finland, France, Germany, Greece, Italy, Japan, Netherlands, Portugal, Spain, Sweden, United Kingdom and the United States of America¹.

¹The Republic of Ireland was also considered initially, but dismissed from the sample because its GDP growth of over 20% in 2015 represents an outlier which cannot be reasonably predicted by any model.

The GDP is measured in constant prices, the observed time period is from 1992 until 2016 on an annual basis. The GDP figures are not corrected for differences in purchasing power. The starting year is chosen to be 1992 because it is the first year where GDP data are available for the reunified Germany in the AMECO database. The AMECO database is the official database of the European Commission.

The aim of this paper is to compare the performance of ANN regression models to that of conventional linear regression models. Therefore, the process of choosing the optimal number of time lags for each model is omitted and the first 3 time lags are chosen for both models. Note that observations from the year 1992 are lost because no growth rate is defined and observations from the years 1993-1995 are lost because of the undefined lags.

3 The starting point: linear models

The starting point, even if a modeler is only interested in fitting an artificial neural network, should be a standard linear model. It is much easier to estimate and sets a performance standard for the neural network.

Three different types of linear models are considered initially. The fixed effects model, a dynamic panel model and the pooled model. Theoretical assumptions of these linear models are not discussed in this article. The focus is on the performance of these models in terms of the root mean squared error², as they mainly serve as a reference for the performance of the ANN model. A slightly deeper discussion of different types of linear GDP forecasting models is given in, e.g. Tkacz (2001) or Kock and Teräsvirta (2014).

A closer look at the fixed effects regression (Table 1) shows that the individual effects only explain around 4% of the error variance ($\sigma_u^2/\sigma_{total}^2$). The null hypothesis that all individual effects are zero cannot be rejected (F-statistic in Table 1). This indicates the absence of unobserved heterogeneity. Note that the constant and the time coefficient depend on the scaling of the variables. The constant can be interpreted as a constant growth rate of 1.796% and the time trend shows a decrease of 0.109 percentage points (pp) per year.

However, the fixed effects estimator may suffer from bias ("Nickell bias") due to the inclusion of lagged dependent variables. Even though the main interest is only in the predictive power of models (and not in individual coefficients), an Arellano-Bond dynamic panel model is estimated additionally (Table 2). Finally, the OLS estimator on the pooled data does not yield very different results, either (Table 3). Therefore, and because of it does not require any transformation of the input data, the pooled model is considered the most adequate of the three linear models for comparison with the ANN model.

At this point, a modeler could, of course, still try to improve the simple linear model. For example, a polynomial time-trend or an interaction term could be appropriate. However, the

²The root mean squared error is defined as $RMSE = \sqrt{1/N(\epsilon^T \epsilon)}$ for the error vector ϵ of length N (number of observations).

Table 1: Fixed effects model of GDP growth rates

gdpchange	coefficient	std. err.	t-value	$P > t $	[95% Conf.	Interval]
l1gdpchange	0.3826	0.0580	6.59	0.000	0.2683	0.4968
l2gdpchange	-0.1699	0.0615	-2.76	0.006	-0.2908	-0.0489
l3gdpchange	0.0834	0.0547	1.53	0.128	-0.0242	0.1911
time	-0.0109	0.0023	-4.75	0.000	-0.0154	-0.0064
constant	0.1796	0.0269	6.67	0.000	0.1266	0.2326
<i>RMSE</i>	0.1964					
$\sigma_u^2/\sigma_{total}^2$	0.044					
$1 - F_{F(14,296)}(0.83)$	0.6322					

Source: AMECO database; own calculations

Table 2: Arellano-Bond dynamic panel model of GDP growth rates

gdpchange	coefficient	std. err.	t-value	$P > t $	[95% Conf.	Interval]
l1gdpchange	0.3732	0.0578	6.44	0.000	0.2560	0.4868
l2gdpchange	-0.1756	0.0610	-2.88	0.004	-0.2952	-0.0560
l3gdpchange	0.0769	0.0545	1.41	0.158	-0.0299	0.1836
time	-0.0115	0.0023	-4.95	0.000	-0.0160	-0.0069
constant	0.1869	0.0274	6.83	0.000	0.1333	0.2405
<i>RMSE</i>	0.1967					

Source: AMECO database; own calculations

Table 3: Pooled model (OLS) of GDP growth rates

gdpchange	coefficient	std. err.	t-value	$P > t $	[95% Conf.	Interval]
l1gdpchange	0.4240	0.0565	7.50	0.000	0.3128	0.5353
l2gdpchange	-0.1444	0.0607	-2.38	0.018	-0.2639	-0.0248
l3gdpchange	0.1200	0.0533	2.25	0.025	0.0151	0.2250
time	-0.0092	0.0022	-4.12	0.000	-0.0134	-0.0046
constant	0.1517	0.0255	5.94	0.000	0.1014	0.2019
<i>RMSE</i>	0.1958					

Source: AMECO database; own calculations

modeler's resources are usually scarce, so he or she might want to choose a model which more or less automatically deals with complex influences of the explanatory variables. Such a model is the ANN regression model which will be introduced in the next section.

4 Neural network regression

One goal of this paper is to give scientists who are familiar with ordinary least-squares (OLS) regression a better understanding along with an example how to apply the ANN model to real-world data.

4.1 Numerical optimization

The OLS problem is to minimize the following loss function:

$$\min_{\beta} E(\beta, X, y) = \min_{\beta} \frac{1}{2} \|y - f(\beta, X)\|^2 = \min_{\beta} \frac{1}{2} \sum_i (y_i - f(\beta, x_i))^2, \quad (1)$$

where x_i denotes the i -th row of the design matrix X , i.e., the vector of variables (including a constant) belonging to the i -th observation. The dependent variable y is called target variable in the context of ANN. The notation for the regression function is f both for the vector-valued version $f(\beta, X)$ and the scalar-valued version $f(\beta, x_i)$. If the function f is linear in the parameters, $f(\beta, X) = X\beta$, it is well known that the parameter vector which solves the ordinary least squares problem is given by:

$$\hat{\beta} = (X^T X)^{-1} X^T y \quad (2)$$

This closed-form solution is in fact the reason why the OLS approach is so simple and favorable. However, this explicit solution does not generally exist if f is a non-linear function of the parameters. Numerical optimization techniques allow for a generalization of the OLS concept to those general, in particular non-linear, functions.

One of the simplest numerical optimization algorithms and the basis of many more complex algorithms is gradient descent. It means that the objective function (here: the sum of squared errors) is minimized in a stepwise procedure by moving the parameter values into the direction of the steepest descent of the objective function. Ultimately, the algorithm will run into a point where the gradient is zero (if such a point exists), implying that at least a locally optimal parameter vector is found. More information on gradient descent and more advanced versions is found in the literature (e.g. Avriel, 2003; Wilamowski, 2011).

The simple gradient descent algorithm for solving the OLS problem is:

$$\begin{aligned} \beta &= \mathcal{U}(-2, 2) \\ \text{for } k &= 1 \text{ to } K : \\ \nabla E(\beta, X, y) &= \left(\frac{\partial E(\beta, X, y)}{\partial \beta_1} \quad \dots \quad \frac{\partial E(\beta, X, y)}{\partial \beta_J} \right)^T \\ \beta &= \beta - \eta \cdot \nabla E(\beta, X, y) \end{aligned}$$

First, uniformly distributed random parameters serve as an initial guess. Then, in every iteration, the parameter is updated by moving into the direction of the negative gradient ∇E (defined as the vector containing the partial derivatives of the loss function E with respect to the J parameters). The algorithm itself uses a parameter η which is called the learning rate because it determines how big the steps will be. Bigger steps allow for faster convergence ("learning") but might lead to problems finding the exact minimum, whereas for smaller steps, it is the other way around.

Because of the quadratic loss function, the partial derivatives have the following form:

$$\frac{\partial E(\beta, X, y)}{\partial \beta_j} = \sum_i (y_i - f(\beta, x_i)) \cdot -\frac{\partial f(\beta, x_i)}{\partial \beta_j} \quad (3)$$

In least squares regression, the gradient is generally of the form "error times partial derivative of the regression function". In the standard linear case, the partial derivatives of f simplify further to:

$$\frac{\partial f(\beta, x_i)}{\partial \beta_j} = X_{ij} \quad (4)$$

Using again matrix notation, setting the gradient to zero and solving for β , it remains to the reader to verify that formula 2 is indeed replicated:

$$\nabla E(\beta, X, y) = (y - X\beta)^T X \stackrel{!}{=} 0 \quad (5)$$

Coming back to non-linear regression functions, extensions of the gradient descent method often consider the Hessian or Jacobian matrix. The Hessian matrix is the multidimensional second-order derivative. In this paper, the Levenberg-Marquardt algorithm is employed which uses the Jacobian matrix. The Jacobian of f w.r.t. β is defined as:

$$\mathcal{J} = \mathcal{J}(\beta) = \begin{pmatrix} \frac{\partial f(\beta, x_1)}{\partial \beta_1} & \dots & \frac{\partial f(\beta, x_1)}{\partial \beta_J} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\beta, x_N)}{\partial \beta_1} & \dots & \frac{\partial f(\beta, x_N)}{\partial \beta_J} \end{pmatrix} \quad (6)$$

The Jacobian contains all (first-order) partial derivatives and is used to approximate f locally around β in the following sense:

$$f(\beta + \delta, X) \approx f(\beta, X) + \mathcal{J}\delta \quad (7)$$

The goal still is to move the parameters vector β into the direction of the (assumed) minimum of the loss function E . The step δ that locally minimizes the loss function can be determined via the first order condition (analogous to equation 5):

$$\delta = \left(\mathcal{J}^T \mathcal{J} \right)^{-1} \mathcal{J}^T (y - f(\beta, X)) \quad (8)$$

The specific feature of the Levenberg-Marquardt algorithm is to avoid situations where $\mathcal{J}^T \mathcal{J}$ is close to singular by adding a diagonal (in particular invertible) matrix as a second term:

$$\delta^{LM} = \left(\mathcal{J}^T \mathcal{J} + \lambda \text{diag}(\mathcal{J}^T \mathcal{J}) \right)^{-1} \mathcal{J}^T (y - f(\beta, X)) \quad (9)$$

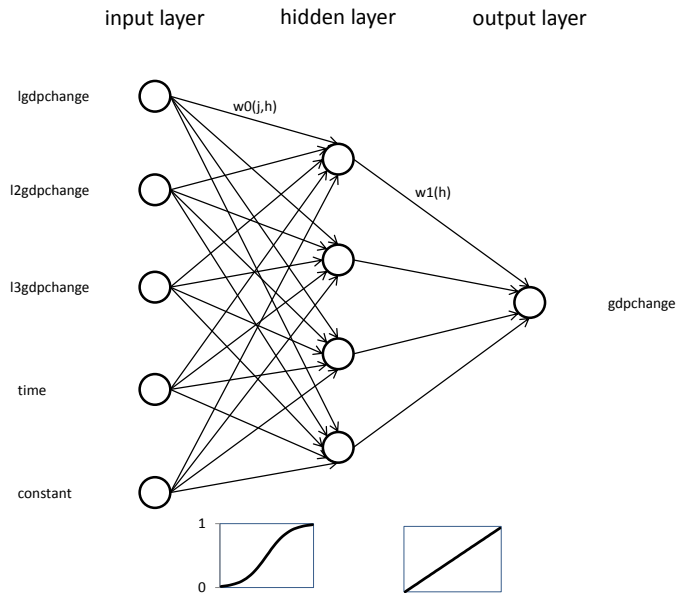
The parameter λ is a tuning parameter similar to the learning rate η in the gradient descent algorithm. The difference is that λ is adjusted from iteration to iteration. A useful in-depth discussion of the Levenberg-Marquardt algorithm in the context of ANN is given in Yu and Wilamowski (2011). The Levenberg-Marquardt algorithm is generally considered one of the most efficient training algorithm for neural networks (Hagan and Menhaj, 1994).

4.2 The artificial neural network regression model

There is a vast literature on neural networks and their different architectures (e.g. Wilamowski, 2011). The basic idea is that the true regression function is an (almost) arbitrary function of the explaining variables. In contrast to the conventional approach, no specific structure is assumed for the true regression function. Artificial neural networks are useful in this setting because the universal approximation theorem states that a single hidden layer feedforward (SHLFF) network of sufficient complexity is able to approximate any given (continuous) regression function on a compact set to an arbitrary degree. The term "single hidden layer" means that, in addition to the input and the output layer, the network consists of one hidden layer. The expression "feedforward" means that there is only one direction of information flow, from the inputs through the hidden layer to the output. The network is shown in Figure 1.

The two curves below the actual network in Figure 1 indicate which type of activation function is used to connect the adjacent layers. The $H = 4$ neurons in the hidden layer are activated by a sigmoid function and the output neuron will be activated by a linear function. The choice of the number of hidden neurons is a problem of model selection, similar to the one arising in other regression models such as the degree in polynomial regression. In order to avoid an excessively large network, a certain ratio of observations per parameter can be required as a rule of thumb. The total number of parameters in a SHLFF network is $P = (J + 1) \cdot H$. Requiring at least 10 observations per parameters yields $H \leq (0.1N)/(J + 1)$. In the present case, $H \leq 31.5/6 = 5.25$. Testing revealed that the results with 4 hidden neurons were not worse than those with 5.

Figure 1: Visual representation of the artificial neural network



Source: own illustration

The algebraic formulation of the artificial neural network is:

$$f(w^0, w^1, x_i) = \sum_h w_h^1 \left(1 + \exp \left(- \sum_j w_{jh}^0 X_{ij} \right) \right)^{-1} = \sum_h w_h^1 \phi \left(\sum_j w_{jh}^0 X_{ij} \right), \quad (10)$$

with ϕ denoting the sigmoid function. The parameters w_h^1 , w_{jh}^0 are called weights because they describe how much weight a certain neuron loads on the neurons in the following layer.

The full regression model, including i.i.d. normal errors γ_i is:

$$y_i = \sum_h w_h^1 \phi \left(\sum_j w_{jh}^0 X_{ij} \right) + \gamma_i \quad (11)$$

The value of the output neuron is a linear combination of the values of the hidden neurons which is consistent with some approaches in the relevant literature (e.g. Chuku et al., 2017), whereas other approaches (e.g. Tkacz, 2001; Qi, 2001) use a sigmoid transformation of this linear combination. Employing also a sigmoid function to connect the hidden layer to the output neuron implies that the network output will always be in the interval (0,1). This makes sense for, e.g., binary classification problems, but is unnecessarily restrictive for general applications where the values of the target variable cover a larger interval. Regarding the present case of the GDP growth rate as target variable, such a model would be unable to predict a negative growth rate.

A constant is included in the input layer because the exact same design matrix X as in the linear model shall be used. Some approaches in the literature (e.g. Qi, 2001; Kock and Teräsvirta, 2014; Chuku et al., 2017) use a constant both in the input and in the hidden layer. In Tkacz (2001) it remains unclear whether constants are used at all. From a theoretical point of view,

even without any explicit constant, the SHLFF network would be able to account for a constant component of the target variable. If the target variable was indeed just white noise, the training process would very likely result in weights such that the influences of the considered input variables neutralize each other. These weight would not necessarily be small and the probability of non-convergence of the algorithm would be rather high. On the other hand, in the same situation, a constant in the input layer (this paper’s model) should produce more stable algorithm runs where the resulting weights for all explanatory variables except the constant are small. In light of these considerations, an additional constant in the hidden layer is generally not be needed to account for a constant component of the target variable.

Finally, the autoregressive aspects of the model consists of having the lagged GDP growth rates as explanatory (i.e. input) variables (same design matrix as in the linear model). The most important aspect is that the lagged growth rates (and the other input variables) are allowed to have an almost arbitrary influence on the current growth rate. Kock and Teräsvirta (2014, equation 1) use lagged dependent variables both in the input and in the hidden layer. This seems to produce an unnecessarily complex network with more or less redundant parameters. Having the lagged dependent variables in the input layer already allows the model to capture an arbitrary, in particular linear, relationship between the lagged and the current dependent variable. The usefulness of their formulation is, however, that the linear AR model can be interpreted as a special case of their autoregressive ANN model.

In order to determine the optimal parameters of the ANN model (minimal sum of squared errors), the Levenberg-Marquardt algorithm is used. All that needs to be calculated are the (first) partial derivatives of f with respect to the parameters. The starting point is the output neuron, whose value is a linear combination of the values of the neurons of the hidden layer. Therefore, the partial derivatives of f with respect to w_h^1 are simply:

$$\frac{\partial f(w^0, w^1, x_i)}{\partial w_h^1} = \phi \left(\sum_j w_{jh}^0 X_{ij} \right) \quad (12)$$

In order to calculate the partial derivative with respect to w_{jh}^0 , the chain rule is used along with the property of the sigmoid function that its derivative is $\phi' = \phi(1 - \phi)$:

$$\frac{\partial f(w^0, w^1, x_i)}{\partial w_{jh}^0} = w_h^1 \phi \left(\sum_{j'} w_{j'h}^0 X_{ij'} \right) \left(1 - \phi \left(\sum_{j'} w_{j'h}^0 X_{ij'} \right) \right) X_{ij} \quad (13)$$

There is one additional feature that should always be implemented in ANN training algorithms. The feature concerns the concept of validation. Validation means separating the data set into a training set which is only used to derive the parameters which minimize the loss function and a validation set which is only used to assess the performance of the model. A convenient way to implement validation in the Levenberg-Marquardt algorithm is to define a stopping rule (cf. Prechelt, 1997). In every iteration, the model error is evaluated on the validation set (which was not used to update the parameter values) and if the error does not decrease for a given amount

of consecutive iterations, the algorithm (i.e. the loop over k) stops. Technically, the stopping rule prevents a loss of the model's generalizability (cf. Prechelt, 1997).

The precise algorithm in pseudo-code is:

```

 $w = \mathcal{U}(-2, 2)$ 
for  $k = 1 : 200$ 
  if  $up = 1$ 
    calculate the MSE on the training and validation data at the current weights  $w$ 
    if the current MSE on the validation data is not smaller than the stored one
      counter = counter+1
      if counter = 4
        BREAK
      else
        replace the stored MSE on the validation data by the current one
        counter = 0
    calculate the Jacobian  $\mathcal{J}$  for the training data at the current weights  $w$ 
     $w^{new} = w + \left( \mathcal{J}^T \mathcal{J} + \lambda \text{diag}(\mathcal{J}^T \mathcal{J}) \right)^{-1} \mathcal{J}^T (y - f(w, X))$ 
    if largest (absolute) new weight component is larger than 1e3
      BREAK
    if largest (absolute) step component is smaller than 1e-12
      BREAK
    if MSE on the training data is lower for the new weights  $w^{new}$ 
       $w = w^{new}$ 
       $\lambda = 0.1\lambda$ 
       $up = 1$ 
    else
       $\lambda = 2\lambda$ 
       $up = 0$ 

```

Regarding the further explanation of the algorithm, a random 80% of the observations will be selected for training and the rest for validation. The initial value of the update parameter is $up = 1$ and the initial learning rate is $\lambda = 5$. Weights are initialized as uniformly distributed random variables and, after every successful iteration (training error decrease), the learning rate λ is decreased, the weights are updated and the new Jacobian is calculated. If an iteration is not successful (no training error decrease), the learning rate is increased and a new step is taken with the old weights and old Jacobian.

There are four ways the algorithm terminates. First, it may reach the maximum number of iterations $k = 200$. Second, the error on the validation set may fail to increase for consecutive 4 (successful) iterations. Third, the weights may reach a critically high level (in absolute values). This stops the algorithm early in case of divergence. Finally, the algorithm may stop because the step length in the weight updating process becomes too small, which indicates that the (locally) optimal weights have been found.

Although the Levenberg-Marquardt algorithm is known to be more likely to find the a global optimum compared with the gradient descent algorithm (Yu and Wilamowski, 2011, p. 15), it may still convergence to a local minimum. Therefore, it makes sense to run the algorithm multiple times with different random initial weights in order to enhance the probability of finding the global minimum.

The ANN model with the lowest training error (and only a small difference to the validation error) after a few algorithm runs is shown in Table 4. The weights w_{final}^0 are represented as a matrix with the hidden neurons as rows and the explanatory variables as columns. The weights w_{final}^1 are displayed as an additional column. Regarding performance, it can be seen that the ANN model outperforms the pooled linear model (Table 3) by far regarding the (root) mean squared error.

Table 4: ANN model of GDP growth rates

hidden neuron	w_{final}^0				w_{final}^1	
	constant	l1gdpchange	l2gdpchange	l3gdpchange	time	
1	-0.3516	9.0727	-3.8875	-2.1425	0.7359	0.8328
2	0.3935	1.0711	-6.1186	-7.7060	-1.2881	-0.5032
3	3.3696	-11.4281	-10.5128	-6.2537	-0.8980	0.2364
4	-0.3896	7.7375	-2.4829	-4.8235	4.0138	-0.4947
<i>RMSE</i> (full)	0.1648					
<i>RMSE</i> (training)	0.1665					
<i>RMSE</i> (validation)	0.1579					

Source: AMECO database; own calculations

4.3 Parameter interpretation in ANN

In regression analysis, it is assumed that the regression function f describes the systematic relationship between some set of explaining variables X and the conditional expectation of the dependent variable y in the sense that $\mathbb{E}[y|X] = f(X)$ (e.g. Chernozhukov, 2006). This implies that predictions from regression models formally refer to the conditional expectation of the dependent variable. In parametric regression, the function f is described by a set of parameters.

The standard linear regression model (without polynomial or interaction terms) is not only linear in these parameters but there is a one-to-one correspondence between parameters and explanatory variables. Moreover, the parameters can be interpreted as the marginal effect of the respective explanatory variable on the conditional expectation of the dependent variable. In particular, the marginal effect of an explanatory variable is constant and does not depend on the value of other explanatory variables.

In ANN models, there is obviously no one-to-one correspondence between weights (parameters) and explanatory variables. More importantly, the marginal effect of a certain explanatory vari-

able on the conditional expectation of the dependent variable is a (non-constant) function of all explanatory variables. A usual way to visualize this function is to keep all variables not of interest at their mean or median values and let the variable of interest run over its observable range and plot the value of the regression function (conditional expectation). This will become clearer in the next section.

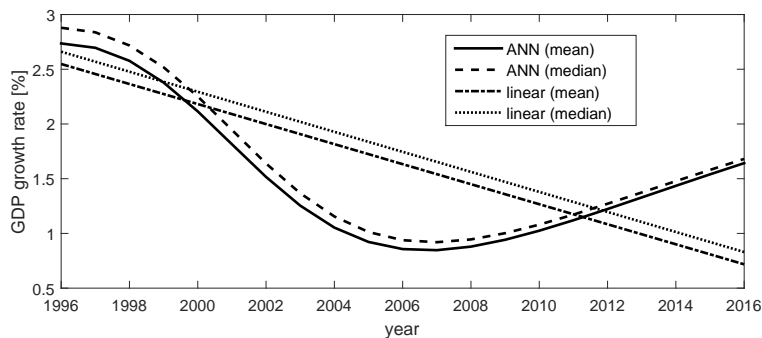
4.4 Prediction and forecast

The remaining goal is to determine whether the linear or the ANN model should be preferred. Since the two models have different numbers of parameters, the (root) MSE is usually not appropriate to compare their performances. The reason is that a model with more parameters is also more likely to fit data better (lower MSE), even if a simpler model is true. In other words, the larger model might only perform well on the available data, but not generalize well to unseen data. This aspect would also be relevant for the comparison of different ANN models.

In the present case, the generalizability of the ANN model has been ensured by validation. The fact that the ANN model performs better than the pooled OLS model (regarding the MSE) on both training and validation set is more than enough reason to conclude that the ANN model is superior. In the following, a little more insight into possible reasons is given.

Considering the marginal effect of the variable "time", Figure 2 illustrates the difference between the linear and the ANN time trend for the GDP growth rates. The curves in Figure 2 show the prediction, i.e. the conditional expectation of the GDP growth rate, given the different observed values of the time variable. The two slightly different curves for both models correspond to the predictions calculated with the other variables at their mean and at their median values, respectively. For the linear prediction, the marginal effect of time as well as the difference between the expectation at the mean and the expectation at the median is constant. For the ANN prediction, the marginal effect of time and the difference between the curves are non-constant.

Figure 2: Estimated time trends of GDP growth rates



Source: AMECO database; own calculations

Regarding the interpretation of Figure 2, the linear model suggests a general decline of growth by a certain rate (0.09 pp per year, see Table 3), whereas the ANN model suggest a decline of growth prior to the global economic crisis in 2009 and an increase afterwards which is, indeed,

more plausible. For panel data, time-specific effects can be used in the linear model to partly overcome the restriction of a linear trend. Regarding forecasting, however, some sort of trend is preferable because one would not want to make an additional forecast of the time-specific effects. Using the ANN and the pooled linear model to predict the GDP growth rates for 2017, one-year-ahead forecasts are displayed in Table 5. The forecasts are calculated as conditional expectations.

Table 5: Out-of-sample forecast of 2017 GDP growth rates by different models

	GDP growth [%]			
	prediction ANN	error ANN	prediction linear	error linear
Austria	2.48	0.46	0.29	2.64
Belgium	2.05	-0.36	0.35	1.34
Denmark	1.90	0.34	0.51	1.73
Finland	2.46	0.17	0.08	2.56
France	2.47	-0.66	0.48	1.34
Germany	2.12	0.10	0.52	1.70
Greece	0.64	0.71	-0.57	1.92
Italy	2.31	-0.81	0.07	1.43
Japan	2.18	-0.47	0.66	1.05
Netherlands	2.37	0.79	0.56	2.60
Portugal	2.47	0.20	0.37	2.30
Spain	1.93	1.13	0.92	2.14
Sweden	2.99	-0.58	1.47	0.94
United Kingdom	1.56	0.23	0.68	1.11
United States	2.31	-0.04	0.92	1.35
RMSPE		0.555		1.833

Source: AMECO database; own calculations

It can be seen that the root mean squared prediction error (RMSPE) is much lower for the ANN model. Regarding the underestimation in the linear model (positive errors), a likely reason is that the linear time trend is quite restrictive and falsely suggests generally declining growth rates over time. Note that the general analysis is based on the AMECO database as of October 2017; the GDP growth rates for 2017 were calculated from the database in June, 2018.

5 Conclusion

This paper demonstrates that an artificial neural network (ANN) regression model is able to yield much better performance than a corresponding linear model in predicting the annual GDP growth rates of 15 industrialized economies in the time period from 1996 to 2016.

The results come closest to those of Tkacz (2001) who finds that the average error is 0.25 percentage points lower for the ANN compared with the best linear model. However, the results in Tkacz (2001) are also questionable because, according to his equation (5), the network output in the approach is constrained to the interval (0,1) due to the sigmoid function connecting the hidden layer with the output layer. This makes it impossible to predict a negative growth rate. This also concerns the results in Qi (2001). The latter further suffers from an error in the def-

initiation of the ANN regression function in equation (1), where the residual is defined as part of the regression function.

The results in Chuku et al. (2017) indicate problems with the convergence of the training algorithm, maybe due to bad scaling. The trained weights exceed the value of 20,000 in the case of Kenya.

Feng and Zhang (2014) do not analyze GDP data but rather the share of the tertiary sector in regional GDP. However, the (autoregressive) ANN model is only compared with a "gray" forecasting model and not with conventional linear models such as AR or ARIMA, which makes it hard to judge its performance. Furthermore, there seem to be only 24 observations (annually from 1971 to 1994) to train the ANN model which has 25 (assuming that the bias/threshold neuron in the hidden layer is included in the stated 5 hidden neurons) or even 30 (otherwise) parameters. The network is simply too big to be reasonably trained from the available observations.

Regarding the results of Kock and Teräsvirta (2014) that ANN models generally do not outperform linear models, it must be remembered that they analyze industrial production and unemployment during a crisis, which is a somewhat different problem than generally predicting GDP growth. Furthermore, their ANN model includes both linear and non-linear terms which makes it possible to understand the linear AR model as a sub-model. However, it also means that their ANN model contains more or less redundant parameters.

Summing up, the contribution of this very paper is threefold. First, it contributes to the GDP forecasting literature by offering a flexible non-linear modeling approach. Second, the exact algorithm used to train the presented ANN regression model is given. This limits the perceived black-box character of the ANN regression model and helps analysts to understand and actually employ ANN models for regression purposes.

References

- Avriel M. (2003): *Nonlinear Programming: Analysis and Methods*, chapter 10, Courier Corporation.
- Chernozhukov V. (2006): *Statistical Method in Economics, Lecture Notes*, downloaded from <https://ocw.mit.edu/> [17.01.2018]
- Chuku C., Odour J., Simpasa A. (2017): Intelligent forecasting of economic growth for African economies: Artificial neural networks versus time series and structural econometric models, *"Forecasting Issues in Developing Economies 2017" conference paper*, Washington.
- Feng L., Zhang J. (2014): Application of artificial neural networks in tendency forecasting of economic growth, *Economic Modelling* 40, 76-80.

- Hagan M.T., Menhaj M. (1994): Training feedforward networks with the Marquardt algorithm, *IEEE Transactions on Neural Networks* 5(6), 989–993.
- Kock A.B., Teräsvirta T. (2014): Forecasting performances of three automated modelling techniques during the economic crisis 2007-2009, *International Journal of Forecasting* 30, 616-631.
- Prechelt L. (1997): Early Stopping - but when?, in: *Neural Networks: Tricks of the trade*, Lecture Notes in Computer Science 1524, Springer.
- Qi M. (2001): Predicting US recessions with leading indicators via neural network models, *International Journal of Forecasting* 17, 383-401.
- Tkacz G. (2001): Neural network forecasting of Canadian GDP growth, *International Journal of Forecasting* 17, 57-69.
- Wilamowski B. (2011): Neural Network Architectures, in: *Industrial Electronics Handbook, vol. 5 – Intelligent Systems*, chapter 6, 2nd Edition, CRC Press.
- Yu H., Wilamowski B. (2011): Levenberg-Marquardt training, in: *Industrial Electronics Handbook, vol. 5 – Intelligent Systems*, chapter 12, 2nd Edition, CRC Press.

The **Hamburg Institute of International Economics (HWWI)** is an independent economic research institute that carries out basic and applied research and provides impulses for business, politics and society. The Hamburg Chamber of Commerce is shareholder in the Institute whereas the Helmut Schmidt University / University of the Federal Armed Forces Hamburg is its scientific partner. The Institute also cooperates closely with the HSBA Hamburg School of Business Administration.

The HWWI's main goals are to:

- Promote economic sciences in research and teaching;
- Conduct high-quality economic research;
- Transfer and disseminate economic knowledge to policy makers, stakeholders and the general public.

The HWWI carries out interdisciplinary research activities in the context of the following research areas:

- Digital Economics
- Labour, Education & Demography
- International Economics and Trade
- Energy & Environmental Economics
- Urban and Regional Economics

Hamburg Institute of International Economics (HWWI)

Oberhafenstr. 1 | 20097 Hamburg | Germany

Telephone: +49 (0)40 340576-0 | Fax: +49 (0)40 340576-150

info@hwwi.org | www.hwwi.org