Kalman Filtering and Neural Networks for Global Temperature Forecasting

Qin, Yi
Undergraduate Student ID: 59207768
School of Information Science and Technology, ShanghaiTech University
qinyi1@shanghaitech.edu.cn

***Abstract*—Global warming problem has been a hot point since the 20th century. The attention paid to the global temperature inspires us to build a model to predict the temperature datas. Traditionally, we can build a state space model according to its trend and seasonal natures. However, sometimes the data cannot be adequately described by simple analytical equations. In such case, we can use the Neural Network model. By comparing the results on the temperature forecasting, we can know more about the differences between the traditional and modern statistical approaches. Here we choose two typical approaches on behalf of traditional and modern, Kalman Filtering and Jordan Neural Networks.**

***Index Terms*—**Kalman filter, state space, Neural Networks, forecast, global temperature

I. Introduction

Signals recorded according to time sometimes have some statistical analysis natures, such as trends, seasons and cycles, so that we can build a model called state space model to forecast them. The state-space model that results is then treated by making judicious use of the celebrated Kalman filters and smoothers, developed originally for estimation and control in space applications. [1]

Theoretically, the *Kalman Filter* has been called the *linear least mean squares estimator* (LLSME) because it minimizes the mean-squared estimation error for a linear stochastic system using noisy linear sensors. It has also been called the *linear quadratic estimator* (LQE) because it minimizes a quadratic function of estimation error for a linear dynamic system with white measurement and disturbance noise. [2] Even today, more than half a century after its discovery, it remains a unique accomplishment in the history of estimation theory. It is the only practical finite-dimensional solution to the real-time optimal estimation problem for stochastic systems, and it makes very few assumptions about the underlying probability distributions except that they have finite means and second central moments (covariances).

In practice, the presence of trend and seasonal variation can be hard to estimate and/or remove. It is often very difficult to define trend and seasonality satisfactorily. Moreover, even if you can correctly identify the trend, it is important to ensure the right sort of trend (linear or nonlinear, local or global) is modeled. This is important in traditional statistical approaches because they require the specification of an assumed signal model. Use of traditional statistical approach requires considerable experience and skill to select the appropriate type of model for a given dataset. The great thing about *neural networks* is that you do not need to specify the exact nature of the relationship (linear, non-linear, seasonality, trend) that exists between the input and output. The *hidden layers* of a neural network remove the need to prespecify the nature of the data generating mechanism. This is because they can approximate extremely complex decision functions.

We use Kalman Filtering (traditional) and Jordan Neural Networks (modern), two approaches to forecast the global temperature. At the meantime of comparing the two predict approaches, we want to figure out the key with global warming.

II. The Global Warming Problem

Consider the global temperature series record shown in $Fig.$ 1. The data are the global mean land-ocean temperature index from 1880 to 2017, with the base period 1951-1980. [3] In particular, the data are deviations, measured in degrees centigrade, from the 1951-1980 average, and are an update of Hansen et al. (2006).

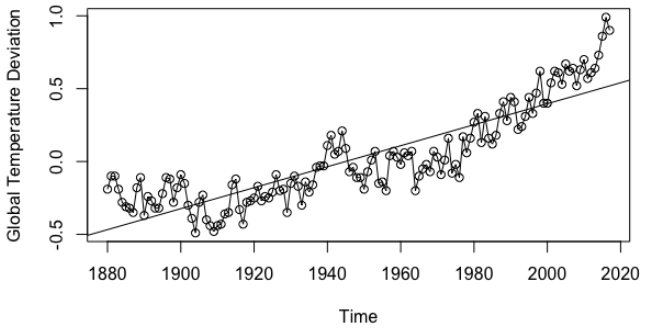

Fig. 1. Yearly average global temperature deviations in degrees centigrade with regression line on time.

We note an apparent *upward trend* in the series during the latter part of the twentieth century and the earlier part of the twenty-first century that has been used as an argument for the global warming hypothesis. Note also the leveling off at about 1935 and then another rather sharp upward trend at about 2010. The question of interest for global warming proponents and opponents is whether the overall trend is natural or whether it is caused by some human-induced interface.

Definition 1 The autocorrelation function (ACF) is defined as

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}},$$

where $\gamma(t,s) = E[(x_t - \mu_t)(x_s - \mu_s)]$.

The ACF measures the linear predictability of the series at time t, say x_t, using only the value x_s. If we can predict x_t *perfectly* from x_s through a linear relationship, $x_t = \beta_0 + \beta_1 x_s$, then the correlation will be $+1$ when $\beta_1 > 0$, and -1 when $\beta_1 < 0$. Hence, we have a rough measure of the ability to forecast the series at time t from the value at time s.

Definition 2 The partial autocorrelation function (PACF) of a stationary process, x_t, denoted Φ_{hh}, for $h = 1, 2, \cdots$, is

$$\Phi_{11} = corr(x_{t+1}, x_t) = \rho(1),$$

and

$$\Phi_{hh} = corr(x_{t+h} - \hat{x}_{t+h}, x_t - \hat{x}_t), \ h \geq 2.$$

The PACF, Φ_{hh}, is the correlation between x_{t+h} and x_t with the linear dependence of $\{x_{t+1}, \cdots, x_{t+h-1}\}$ on each, removed.

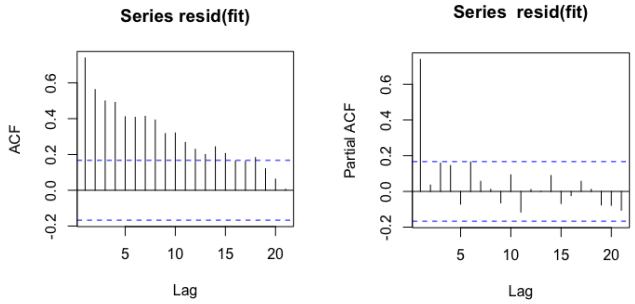

Fig. 2. ACF and PACF of the global temperature's detrended series

Consider the residuals of the regression, which are the detrended datas in $Fig.$ 2. We note that the ACF tails off, and the PACF cuts off after lag 6. It shows that the signal might be relevant to the previous six datas, and the ACF doesn't show any seasonal pattern.

III. Kalman Filtering Approach

A. State Space Model

According to the analysis before, we only need to consider about the *trend component*. By adding a slope term ν_t, we obtain the local linear trend model

$$\begin{aligned} y_t &= \mu_t + \epsilon_t, & \epsilon_t &\sim N(0, \sigma_\epsilon^2), \\ \mu_{t+1} &= \mu_t + \nu_t + \xi_t, & \xi_t &\sim N(0, \sigma_\xi^2), \\ \nu_{t+1} &= \nu_t + \zeta_t, & \zeta_t &\sim N(0, \sigma_\zeta^2), \end{aligned} \tag{1}$$

If $\xi_t = \zeta_t = 0$ then $\nu_{t+1} = \nu_t = \nu$, say, and $\mu_{t+1} = \mu_t + \nu$ so the trend is exactly linear and (1) reduces to the deterministic linear trend plus noise model. The form (1) with $\sigma_\xi^2 > 0$ and $\sigma_\zeta^2 > 0$ allows the trend level and slope to vary over time.

The model can be written in the form

$$\begin{aligned} y_t &= (1 \ \ 0) \begin{pmatrix} \mu_t \\ \nu_t \end{pmatrix} + \epsilon_t, \\ \begin{pmatrix} \mu_{t+1} \\ \nu_{t+1} \end{pmatrix} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} \mu_t \\ \nu_t \end{pmatrix} + \begin{pmatrix} \xi_t \\ \zeta_t \end{pmatrix} \end{aligned} \tag{2}$$

B. Kalman Filter

For convenience we restate the linear Gaussian state space model [4] (we don't need to consider the seasonal and cycle component here) as

$$\begin{aligned} y_t &= Z_t \alpha_t + \epsilon_t, & \epsilon_t &\sim N(0, H_t), \\ \alpha_{t+1} &= T_t \alpha_t + R_t \eta_t, & \eta_t &\sim N(0, Q_t). \end{aligned} \tag{3}$$

Let Y_{t-1} denote the set of past observations $y_1, \cdots, y_{t-1}$ for $t = 2, 3, \cdots$, while Y_0 indicates that there is no prior observation before $t = 1$. Starting at $t = 1$ in (6)(7) and building up the distributions of α_t and y_t recursively, it is easy to show that $p(y_t|\alpha_1, \cdots, \alpha_t, Y_{t-1}) = p(y_t|\alpha_t)$ and $p(\alpha_{t+1}|\alpha_1, \cdots, \alpha_t, Y_t) = p(\alpha_{t+1}|\alpha_t)$.

Lemma 1 ThThe conditional distribution of x given y is normal with mean vector

$$E(x|y) = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \tag{4}$$

and variance matrix

$$Var(x|y) = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{xy}^T. \tag{5}$$

Lemma 2 Whether (x, y) is normally distributed or not, the estimate $\hat{x} = E(x|y)$ is a *minimum variance linear unbiased estimate* (MVLUE) of x given y and its error variance matrix is given by

$$\begin{aligned} Var(\hat{x} - x) &= Var(\Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y) - (x - \mu_x)] \\ &= \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{xy}^T. \end{aligned} \tag{6}$$

Lemma 3 The posterior density of x given y is normal with posterior mean vector (4) and posterior variance matrix (5).

Lemma 4 The linear posterior mean $\hat{x}$ is a *minimum variance linear posterior mean estimate* MVLPME and its error variance matrix is given by (6).

We shall base the derivation on classical inference using Lemma 1. It follows from Lemmas 2 to 4 that the basic results are also valid for minimum variance linear unbiased estimation and for Bayesian-type inference with or without the normality assumption. Returning to the assumption of normality, our object is to obtain the conditional distributions of α_t and α_{t+1} given Y_t for $t = 1, \cdots, n$.

Let $a_{t|t} = E(\alpha_t|Y_t)$, $a_{t+1} = E(\alpha_{t+1}|Y_t)$, $P_{t|t} = Var(\alpha_t|T_t)$ and $P_{t+1} = Var(\alpha_{t+1}|Y_t)$. Since all distributions are normal, it follows from Lemma 1 that conditional distributions of subsets of variables given other subsets of variables are also normal; the distributions of α_t given Y_t and α_{t+1} given Y_t are therefore given by $N(a_{t|t}, P_{t|t})$ and $N(a_{t+1}, P_{t+1})$. Start

with the initial state $\alpha_1 \sim N(a_1, P_1)$ where a_1 and P_1 are known. Let

$$\begin{aligned} v_t &= y_t - E(y_t|Y_{t-1}) \\ &= y_t - E(Z_t\alpha_t + \epsilon_t|Y_{t-1}) \\ &= y_t - Z_t a_t. \end{aligned} \tag{7}$$

Thus v_t is the one-step ahead forecast error of y_t given Y_{t-1}. When Y_{t-1} and v_t are fixed then Y_t is fixed and vice versa. Thus,

$$a_{t|t} = E(\alpha_t|Y_t) = E(\alpha_t|Y_{t-1}, v_t),$$

$$a_{t+1} = E((\alpha_{t+1}|Y_t) = E(\alpha_{t+1}|Y_{t-1}, v_t).$$

But

$$E(v_t|Y_{t-1}) = E(Z_t\alpha_t + \epsilon_t - Z_t a_t|Y_{t-1}) = 0.$$

Consequently, $E(v_t) = 0$ and for $j = 1, \cdots, t-1$,

$$Cov(y_j|v_t) = E[y_j E(v_t|Y_{t-1})^T] = 0.$$

Now apply Lemma 1 to the conditional joint distribution of α_t and v_t given Y_{t-1}.

$$a_{t|t} = E(\alpha_t|Y_{t-1}) + Cov(\alpha_t, v_t)[Var(v_t)]^{-1}v_t \tag{8}$$

Here, $E(\alpha_t|Y_{t-1}) = a_t$ by definition of a_t, and

$$\begin{aligned} Cov(\alpha_t, v_t) &= E[\alpha_t(Z - t\alpha_t + \epsilon_t - Z_t a_t)^T|Y_{t-1} \\ &= E(\alpha_t(\alpha_t - a_t)^T Z_t^T|Y_{t-1}] = P_t Z_t^T \end{aligned}$$

by definition of P_t. Let

$$\begin{aligned} F_t &= Var(v_t|Y_{t-1}) \\ &= Var(Z_t\alpha_t + \epsilon_t - Z_t a_t|Y_{t-1}) \\ &= Z_t P_t Z_t^T + H_t. \end{aligned} \tag{9}$$

Then

$$a_{t|t} = a_t + P_t Z_t^T F_t^{-1} v_t. \tag{10}$$

By Lemma 1 we have

$$\begin{aligned} P_{t|t} &= Var(\alpha_t|Y_t) = Var(\alpha_t|Y_{t-1}, v_t) \\ &= Var(\alpha_t|Y_{t-1}) - Cov(\alpha_t, v_t)[Var(v_t)]^{-1}Cov(\alpha_t, v_t)^T \\ &= P_t - P_t Z_t^T F_t^{-1} Z_t P_t. \end{aligned} \tag{11}$$

We assume that F_t is nonsingular; this assumption is normally valid in wellformulated models. Relations (10) and (11) are sometimes called the *updating step* of the Kalman filter.

We now develop recursions for a_{t+1} and P_{t+1}. Since (3), we have

$$a_{t+1} = E(T_t\alpha_t + R_t\eta_t|Y_t) = T_t E(\alpha_t|Y_t),$$

$$P_{t+1} = Var(T_t\alpha_t + R_t\eta_t|Y_t) = T_t Var(\alpha_t|Y_t)T_t^T + R_t Q_t R_t^T.$$

Substituting from (10) gives

$$a_{t+1} = T_t a_{t|t} = T_t a_t + K_t v_t, \tag{12}$$

where

$$K_t = T_t P_t Z_t^T F_t^{-1}. \tag{13}$$

The matrix K_t is referred to as the *Kalman gain*. We observe that a_{t+1} has been obtained as a linear function of the previous value at and the forecast error v_t of y_t given Y_{t-1}. Substituting from (11) and (13) gives

$$P_{t+1} = T_t P_t (T_t - K_t Z_t)^T + R_t Q_t R_t^T. \tag{14}$$

Relations (12) and (14) are sometimes called the *prediction step* of the Kalman filter.

The recursion (7)-(14) is called the *Kalman filter*. Once $a_{t|t}$ and $P_{t|t}$ are computed, it suffices to adopt the relations

$$a_{T+1} = T_t a_{t|t},$$

$$P_{t+1} = T_t P_{t|t} T_t^T + R_t Q_t R_t^T,$$

for predicting the state vector α_{t+1} and its variance matrix at time t.

C. *Global Temperature Forecasting*

Apply the Kalman Filter Approach to the global temperature prediction application. The results fit the actual signals well with such appropriate model. The results figure is as $Fig.3$:

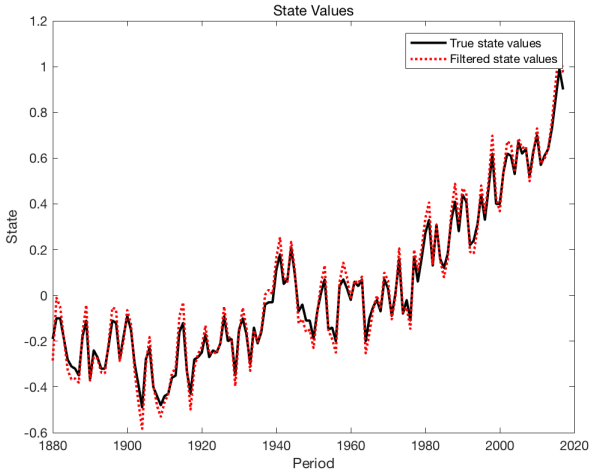

Fig. 3. Kalman filter prediction for the global temperature

IV. Neural Networks Approach

A. *Basic Neural Networks*

A *feed-forward neural network* (often called a multi-layer perceptron) is constructed from a number of interconnected nodes known as neurons, see $Fig.$ 4. [5] These are usually arranged into layers. A typical feedforward neural network will have at a minimum an input layer, a hidden layer and an output layer. The $input\ layer$ nodes corresponds to the number of features or attributes you wish to feed into the neural network. These are akin to the covariates (independent variables) used in a linear regression model. The number of $output\ layer$ nodes correspond to the number of items we wish to predict or classify. The $hidden\ layer$ nodes are generally used to perform non-linear transformation on the original input attributes. In their simplest form, feed-forward neural

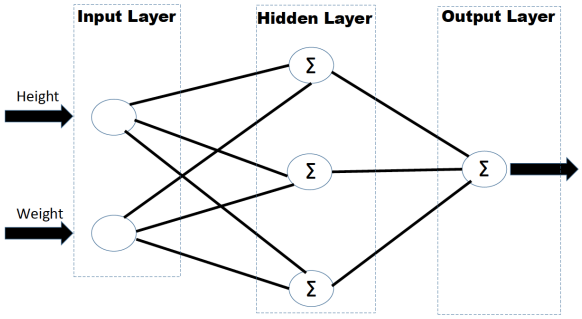

Fig. 4. A basic neural network

networks propagate attribute information through the network to make a prediction, whose output is either continuous for regression or discrete for classification.

At the heart of an artificial neural network is a mathematical node, unit or neuron. It is the basic $processing$ element. The input layer neurons receive incoming information which they process via a mathematical function and then distribute to the hidden layer neurons. This information is processed by the hidden layer neurons and passed onto the output layer neurons.

1) The Working of the Neuron Simplified: Given a sample of input attributes $x_1, \cdots, x_n$, a weight w_{ij} is associated with each connection into the neuron; and the neuron then sums all inputs according to:

$$u = \sum_{i=1}^{n} w_{ij} x_j + b_j \tag{15}$$

The parameter b_j is known as the bias, and is similar to the intercept in a linear regression model. It allows the network to shift the activation function "upwards" or "downwards". This type of flexibility is important for successful machine learning.

2) Activation Functions: Each neuron contains an *activation function*, see $Fig.$ 5, and a *threshold value*. The threshold value is the minimum value that an input must have to activate the neuron. The activation function is applied and the output passed to the next neuron(s) in the network.

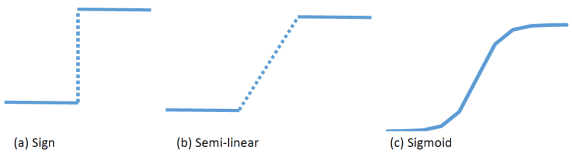

Fig. 5. Three activation functions

The task of the neuron is to perform a weighted sum of input signals, and apply an activation function before passing the output to the next layer.

The sigmoid (or logistic) function is a popular choice. It takes a real-valued number and "squashes" it into a range between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1. It is given by:

$$f(u) = \frac{1}{1 + exp(-cu)} \tag{16}$$

It gained popularity partly because the output of the function can be interpreted as the probability of the artificial neuron "firing".

3) Gradient Descent: In general, we want to find the weights and biases which minimize the error function. Gradient descent updates the parameters iteratively to minimize the overall network error. It iteratively updates the weight parameters in the direction of the gradient of the loss function until a minimum is reached. In other words, we follow the direction of the slope of the loss function downhill until we reach a valley. The basic idea is roughly illustrated in $Fig.$ 6.

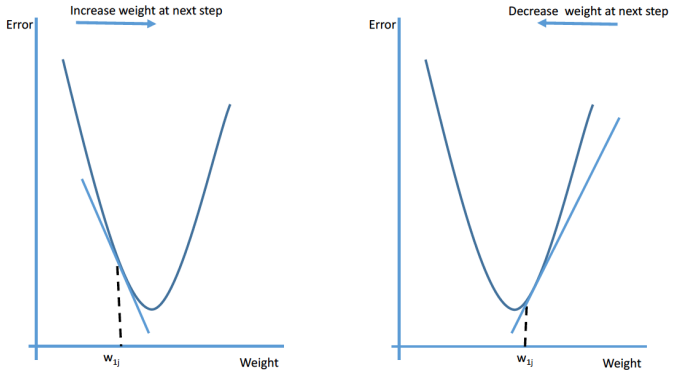

Fig. 6. Basic idea of Stochastic Gradient minimization

Stochastic Gradient Descent (SGD) is an approximation of the true gradient. At each iteration, it randomly selects a single example to update the parameters, and moves in the direction of the gradient with respect to that example. It therefore follows a noisy gradient path to the minimum. Due in part to the lack of redundancy, it often converges to a solution much faster than traditional gradient descent.

4) Learning Rate: It determines the size of the steps taken to reach the minimum by the gradient descent algorithm. With a large learning rate, the network may learn very quickly, a lower learning rate takes longer to find the optimum value. But if the learning rate is too high, the network may miss the global minimum and not learn very well or even at all.

5) Momentum: It can take a value between 0 and 1. It adds this fraction of the previous weight update to the current one. A high value for the momentum parameter can reduce training time and help the network avoid getting trapped in local minima. However, setting the momentum parameter too high can increase the risk of overshooting the global minimum.

B. *Jordan Neural Networks*

A Jordan neural network is a single hidden layer feed-forward neural network. it is often known as a simple *recurrent neural network* (RNNs). The context (delay) neurons are fed from the output layer, see $Fig.$ 7, it therefore "remembers" the output from the previous time-step.

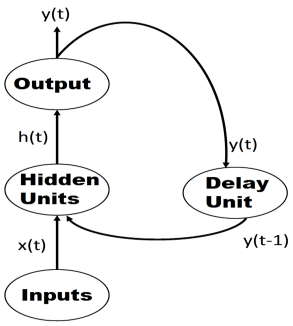

Fig. 7. Structure of the Jordan Neural Network

1) Recurrent Neural Network: RNNs contain hidden states which are distributed across time. This allows them to efficiently store a lot of information about the past. As with a regular feed-forward neural network, the non-linear dynamics introduced by the nodes allows them to capture complicated time series dynamics. At each time step, the network processes:

- The input attributes (x_t);
- Updates its hidden state via activation functions (h_t);
- and uses it to make a prediction of its output (y_t).

2) Backpropagation Through Time: Unfolding simple unrolls the recurrent loop over time to reveal a feedforward neural network. The unfolded RNN, see $Fig.$ 8, is essentially a deep neural network where each layer corresponds to a time step in the original RNN.

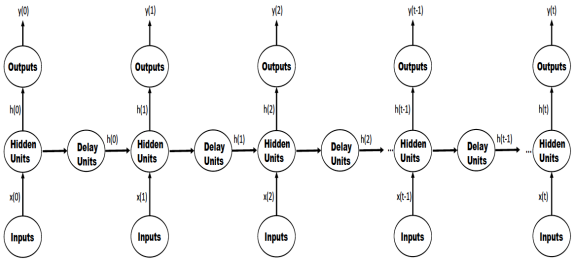

Fig. 8. Unfolding a RNN

C. *Global Temperature Forecasting*

We use the Jordan Neural Networks model to forecasting the global temperature. We need to scale the data in a range, choose lags from 1 to 6 (see $Fig.$ 2) and clean up the empty datas before training. Since we have 138 datas, we can choose 110 of them into training model.

By evaluating model performance in $Fig.$ 9, we find that the error falls sharply within the first 100 or so iterations; and by around 300 iterations it is stable. By assessing test set performance, we find that the squared correlation coefficient is relatively high at close to 0.848.

$Fig.$ 10 shows the actual and predicted values for our model. Notice how well it captures the underlying dynamics of the global temperature signals. Although, not an exact fit,

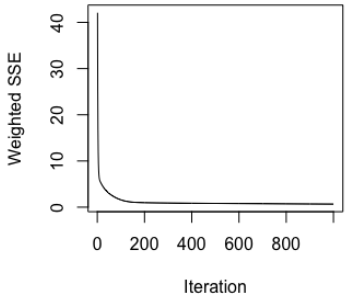

Fig. 9. Training error by iteration

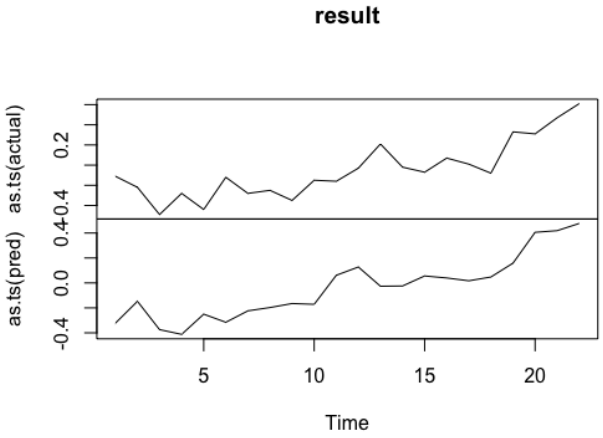

Fig. 10. Actual (top) and predicted values using a Jordan neural network

the model provides a great benchmark from which to compare alternative model specifications. We can at least say, our initial model has good success in predicting the global temperature.

V. Conclusion

Focusing on the global warming problem, we use the State Space model and Neural Networks model to forecast, and both get good results. We use Kalman Filter to process the state space model with trends. In case it cannot be analyzed linearly, we apply the Jordan Neural Networks in another way. It turns out that both of them can describe and predict the global temperature very well.

References

[1] Robert H. Shumway and David S. Stoffer, "*Time Series Analysis and Its Applications*", 3rd ed. Springer. 2011
[2] Mohinder S. Grewal and Angus P. Andrews, "*Kalman Filtering: Theory and Practice Using MATLAB*", 4th ed. John Wiley&Sons, Inc. 2015
[3] http://data.giss.nasa.gov/gistemp/graphs/
[4] J. Durbin and S. J. Koopman, "*Time Series Analysis by State Space Methods*", 2nd ed. OXFORD. 2012
[5] N.D. Lewis, "*Neural Networks for Time Series Forecasting with R*". 2017