TIME SERIES ANALYSIS

Chapter 11
Neural Networks in Time Series Analysis

1 Neural network architecture

Artificial neural networks are forecasting methods which allow complex nonlinear relationships between the response variable and its predictors.

A neural network can be thought of as a network of "neurons" which are organized in layers. The predictors (or inputs) form the bottom layer, and the forecasts (or outputs) form the top layer. There may also be intermediate layers containing "hidden neurons."

The simplest networks contain no hidden layers and are equivalent to linear regressions. Figure 11.11 shows the neural network version of a linear regression with four predictors. The coefficients attached to these predictors are called "weights." The forecasts are obtained by a linear combination of the inputs. The weights are selected in the neural network framework using a "learning algorithm" that minimizes a "cost function" such as the MSE. Of course, in this simple example, we can use linear regression which is a much more efficient method of training the model. This is known as a multilayer feed-forward network, where each layer of nodes receives inputs from the previous layers. The outputs of the nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output. For example, the inputs into hidden neuron j in the above figure are combined linearly to give

$$z_j = b_j + \sum_{i=1}^{4} \omega_{ij} x_i$$

In the hidden layer, this is then modified using a nonlinear function such as a sigmoid,

$$s(z) = \frac{1}{1 + e^{-z}}$$

to give the input for the next layer. This tends to reduce the effect of extreme input values, thus making the network somewhat robust to outliers.

The parameters b_1, b_2, b_3 and ω_{11}, ω_{43} are "learned" from the data. The values of the weights are often restricted to prevent them from becoming too large. The parameter that restricts the weights is known as the "decay parameter".

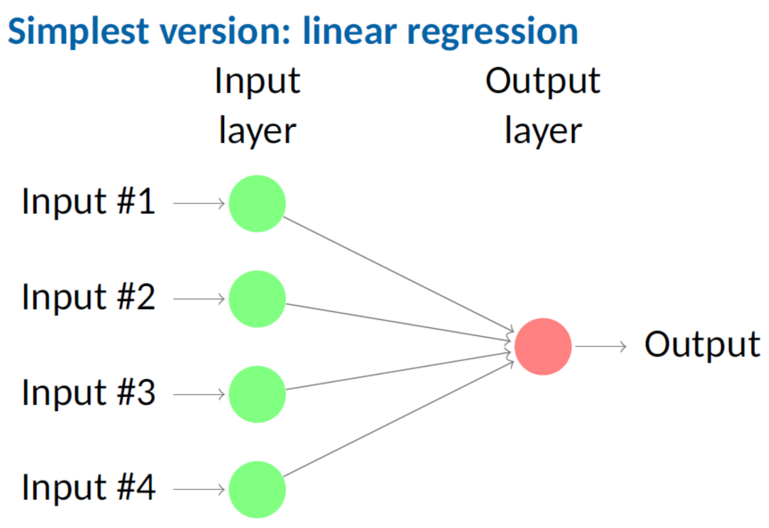

Figure 1:

Once we add an intermediate layer with hidden neurons, the neural network becomes non-linear. A simple example is shown below.

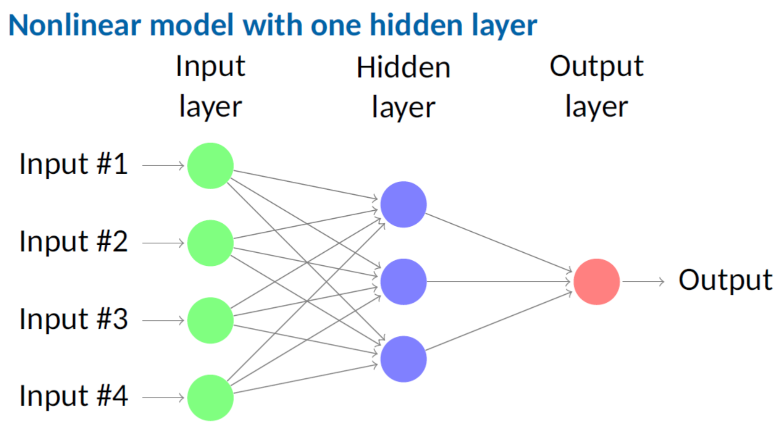

Figure 2:

The weights take random values to begin with, and these are then updated using the observed data. Consequently, there is an element of randomness in the predictions produced by a neural network. Therefore, the network is usually trained several times using different random starting points, and the results are averaged.

The number of hidden layers, and the number of nodes in each hidden layer, must be specified in advance. We will consider how these can be chosen using cross-validation later in this chapter.

Also neural nets are known to not work well with the trend data. We should therefore, de-trend or difference the data before running neural net model.

2 Neural network autoregression

With time series data, lagged values of the time series can be used as inputs to a neural network, just as we used lagged values in a linear autoregression model. We call this a neural network autoregression or NNAR model.

We only consider feed-forward networks with one hidden layer, and we use the notation $NNAR(p, k)$ to indicate there are p lagged inputs and k nodes in the hidden layer. For example, a NNAR(9,5) model is a neural network with the last nine observations $(y_{t-1}, \cdots, y_{t-9}$ used as inputs for forecasting the output and with five neurons in the hidden layer. A $NNAR(p, 0)$ model is equivalent to an ARIMA(p,0,q) model, but without the restrictions on the parameters to ensure stationarity.

With seasonal data, it is useful to also add the last observed values from the same season as inputs. Generally, an $NNAR((p, P, k)_s$ model has inputs $(y_{t-1}, \cdots, y_{t-p}, y_{t-m}, y_{t-2m}, \cdots, y_{t-Pm}$ and k neurons in the hidden layer. A $NNAR(p, P, 0)_s$ model is equivalent to a seasonal ARIMA$(p, 0, 0) \times (P, 0, 0)_s$ model but without the restrictions on the parameters that ensure stationarity.

The $nnetar()$ function fits an $NNAR((p, P, k)_s$. If the values of p and P are not specified, they are selected automatically. For non-seasonal time series, the default is the optimal number of lags (according to the AIC) for a linear AR(p) model. For seasonal time series, the default values are $P = 1$ and p is chosen from the optimal linear model fitted to the seasonally adjusted data. If k is not specified, it is set to $k = \frac{p+P+1}{2}$ (rounded to the nearest integer).

Neural nets has inherent random component. Therefore, it is suggested that the neural net model is run several times, 20 is the minimum requirement. Final result is then presented as mean or median

There are many packages that allows one to compute neural net models. However,$nnetar()$ from forecast is most user friendly. One handy thing about $nnetar()$ is automatic selection of parameters. For more advanced implementation of the neural nets one can look at $mlp()$ function form nnfor package.

When it comes to forecasting, the network is applied iteratively. For forecasting one step ahead, we simply use the available historical inputs. For forecasting two steps ahead, we use the one-step forecast as an input, along with the historical data. This process proceeds until we have computed all the required forecasts.

Here, the last 10 observations are used as predictors, and there are 6 neurons in the hidden layer. The cyclicity in the data has been modelled well. We can also see the asymmetry of the cycles has been captured by the model, where the increasing part of the cycle is steeper than the decreasing part of the cycle. This is one difference between a NNAR model and a linear AR model Ů while linear AR models can model cyclicity, the modelled cycles are always symmetric.

Example 1 The surface of the sun contains magnetic regions that appear as dark spots. These affect the propagation of radio waves, and so telecommunication companies like to predict sunspot activity in order to plan for any future difficulties. Sunspots follow a cycle of length between 9 and 14 years. In Figure 11.13, forecasts from an NNAR(10,6) are shown for the next 30 years. We have set a Box-Cox transformation with lambda=0 ■

```
library(forecast)
fit <- nnetar(sunspotarea, lambda=0)
autoplot(forecast(fit,h=30))
```

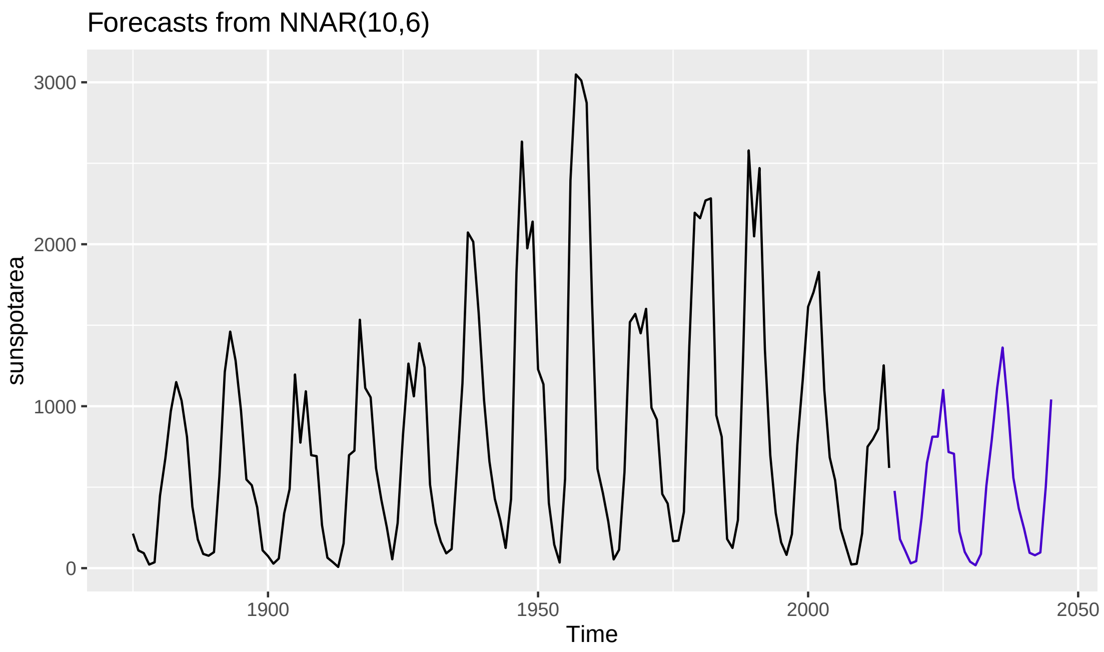

Figure 3:

This chapter refers to Forecasting: Principles and Practice by Hyndman and George Athanasopoulos.