<http://stats.stackexchange.com/questions/8000/proper-way-of-using-recurrent-neural-network-for-time-series-analysis>

J.T. Connor, R.D. Martin, and L.E. Atlas. Recurrent neural networks and robust time series prediction. IEEE Transactions on Neural Networks, Mar 1994. Volume 5, Issue 2, pp. 240 - 254.

http://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

http://stackoverflow.com/questions/25967922/pybrain-time-series-prediction-using-lstm-recurrent-nets?rq=1

http://stackoverflow.com/questions/16830946/request-for-example-recurrent-neural-network-for-predicting-next-value-in-a-seq?rq=1

http://stats.stackexchange.com/questions/2328/how-to-perform-neural-network-modelling-effectively?rq=1

**Time Series Prediction and Neural Networks-neural30-notes 重点阅读. (该文含周期性的时序案例，还有疑问)**

**reconstruct the state space using the index (the time features), such as weekly index, monthly index, season index, and holiday index.**

**reference:**

**1Modelling and forecasting daily electricity loadcurves- a hybrid approach**

**1Supplementary document for “Modelling andforecasting daily electricity load curves\_ a hybridapproach\_**

#### neural networks

##### reconstruct the state space

Consider a single variable *x* which varies with time, one common approach is to sample *x* at regular time intervals to yield a series of observations *x*τ-2, *x*τ-1,*x*τand so on. We can then take such observations and present them as the input vector to the network and use observation *x*τ+1 as the target value. By stepping along the time axis one sample at a time we can form the training set for the problem.

Because of the trend and periodical, the sampling rate (or sampling frequency) between observations often requires empirical optimization. There are methods for helping us out but these are advanced methods from digital signal processing (DSP), dynamical systems theory (DST) and information theory (such as, **phase-space projection** and **false neighbour removal**, to analyse the data, and to help predict what the memory size should be and also how often the data should be sampled.).

It can be useful to remove trend and other components (such as seasonal trends), only if we are able to detect such components. It depends on the topologies of neural network!

(**spectral analysis/signal processing**—sampling rate(抽样频率)🡪spectrum map(频谱图)

🡪period(周期))

Different combinations of input variables can have significant impact on the model performance.

###### the slidingwindow technique

Input part of the time series is called window, By shifting the window over time series the items of training set are made. For example, using Xt-2, Xt-1, Xt predict Xt+1 is a sliding window of three time steps.

In this case, we maybepreprocess the data and determine the window size (it depends).

1. de-trending/de-seasonal

time series decomposition method can be used.

1. window size/embedding dimension

This implicit transformation of a one-dimensional time vector into an infinite dimensional spatial vector is called embedding. The input space to our predictor must be finite. At each instant t, truncate the history to only the previous d samples. d is called the embedding dimension.

you can use**mutual information** to determine the sampling rate, and**false nearest neighbors** to determine the embedding dimension.or you can use linear ACF/PACF to start!

* + 1. Takens Embedding Theorem

A way to reconstruct the state space was introduced by Packard et al. and mathematically analyzed by Takens. A state vector is defined as



 is a state vector (This is a way to reconstruct the state space).

delay time: 

embedding dimension: 

 is the time span represented by an embedding vector.

attractor dimension: 



示例：



6 input 1 output



注：

1. 若和无法确定，则用ACF/PACF确定嵌入维度.
2. 当含有周期性时，可用周期赋值.

reference:

**1Using Support Vector Machines for Time Series Prediction-MSRS00-notes**

<http://www.scholarpedia.org/article/Attractor>

<http://www.scholarpedia.org/article/Attractor_dimensions>

<http://www.scholarpedia.org/article/Attractor_reconstruction>

In the [mathematical](https://en.wikipedia.org/wiki/Mathematics) field of [dynamical systems](https://en.wikipedia.org/wiki/Dynamical_system), an **attractor** is a set of numerical values toward which a system tends to evolve, for a wide variety of starting conditions of the system.[[1]](https://en.wikipedia.org/wiki/Attractor#cite_note-1) System values that get close enough to the attractor values remain close even if slightly disturbed.

In finite-dimensional systems, the evolving variable may be represented [algebraically](https://en.wikipedia.org/wiki/Algebra) as an *n*-dimensional [vector](https://en.wikipedia.org/wiki/Coordinate_vector). The attractor is a region in [*n*-dimensional space](https://en.wikipedia.org/wiki/Space_(mathematics)). If the evolving variable is two- or three-dimensional, the attractor of the dynamic process can be represented [geometrically](https://en.wikipedia.org/wiki/Geometry) in two or three dimensions. If the variable is a [scalar](https://en.wikipedia.org/wiki/Scalar_(mathematics)), the attractor is a subset of the real number line.

**Linear equation or system**

A single-variable (univariate) linear [difference equation](https://en.wikipedia.org/wiki/Difference_equation) of the [homogeneous form](https://en.wikipedia.org/wiki/Homogeneous_equation) {\displaystyle x\_{t}=ax\_{t-1}} diverges to infinity if |*a*| > 1 from all initial points except 0; there is no attractor and therefore no basin of attraction. But if |*a*| < 1 all points on the number line map asymptotically (or directly in the case of 0) to 0; 0 is the attractor, and the entire number line is the basin of attraction.

###### trend

A trend-stationary (TS) series can be made stationary by removing its deterministic trend, whereas difference-stationary (DS) series or series with stochastic trend can be made stationary by differencing (TS, DS, and detrending related issues will be discussed in detail in Section II) These two detrending approaches are not equivalent and should not be used interchangeably.

linear deterministic trend (LDT), linear stochastic trend (LST), nonlinear deterministic trend (NDT), nonlinear stochastic trend (NST), deterministic trend with structure break (SBD), and stochastic trend with structure break (SBS)

the LDT case, linear detrending is the most effective way for NNs to significantly outperform other methods in out-of-sample forecasting performance.

As most realworld time series are nonlinear and/or stochastic, we may be able to conclude that differencing data first is the best practical approach to building an effective NN forecasting model.

reference:

Trend Time–Series Modeling and Forecasting With Neural Networks Min Qi and G. Peter Zhang

https://forecasters.org/pdfs/IEEE\_TNN\_2008.pdf

###### seasonal

In order to identify the true seasonality complicated with the trend factor. We can apply a simple rule of thumb to test if there is a significant correlation between time-series values separated by four lags.

If the time series include the seasonal trends, we can using the sliding window technique as before, then just plus the season lags.

For an elementary discussion of trend and various other practical problems in forecasting time series with NNs, such as seasonality, see Masters (1993). For a more advanced discussion of NN forecasting of economic series, see Moody (1998).

Both de-seasonalization and de-trending is very helpful in improving NN performance, it make time series more stationary, thus simplifying the modeling task.

reference:

Masters, T. (1993). Practical Neural Network Recipes in C++, San Diego: Academic Press.

Moody, J. (1998), "Forecasting the economy with neural nets: A survey of challenges and solutions," in Orr, G,B., and Mueller, K-R, eds., Neural Networks: Tricks of the Trade, Berlin: Springer.

Quarterly Time-Series Forecasting With Neural Networks G. Peter Zhang and Douglas M. Kline

https://forecasters.org/pdfs/IEEE\_TNN\_2007.pdf

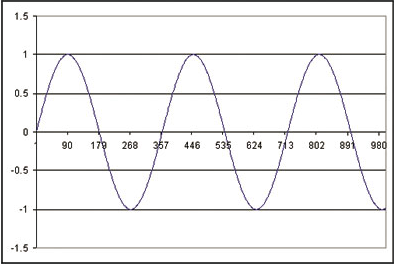
###### transformation

the Box-Cox power transformation

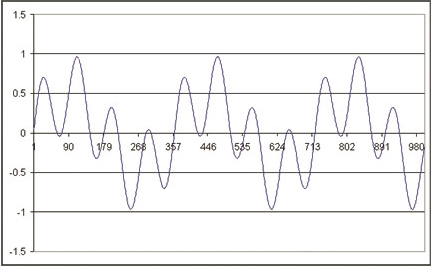
##### topologies

**input layer**

(the hill walking analogy)



If you imagine walking along the simple time series, as you reach a peak (the slope decreases negatively) you know you will go down in a little while. Equally as you reach a dip (the slope decreases positively) you know you should go up in a bit. Also depending on how accurately you have measured the slope you also know how far along you are. But how accurately and for how long should you remember the slope for? The last 5 steps, 15, 30, 90 steps?



It is even worse for the complicate of this time series! The local slope at each peak and trough are the same, but the peaks and troughs are at differing heights! Imagine standing on the peak at 100°? You know you should go down, but how far? If you where standing on the peak at 180° you still have to go down, but this time you go down further ‘*absolute*’ as you started this time at +0.3 but at the 100° peak you where at +1. The relative distance is still the same, but the absolute starting height is different! Remember you cannot cheat and look at the graph you only know where you have ‘just’ come from (this is similar to the local minima problem in neural net training) . In other words you can only look at the last ‘n’ samples you have.

So we have to work out a sampling frequency and a size of memory.Remember the ‘memory’ is an input vector to the network, so that defines the number of input nodes of the network. We will term the ‘frequency’ as how often we remember a step. So for example we will remember every fifth step and keep a running note of the last twenty. In other words we will have walked 100 steps, but only remember the 1st, 6th, 11th so on (frequency=5). As you can imagine the number of possibilities grows astronomically large very quickly.

Using this ‘wordy’ notation the training sets that we will be using are described in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| frequency | memory | distance | input vectors |
| 1 | 5 | 5 | t-1, t-2, t-3, t-4, t-5 |
| 2 | 10 | 20 | t-2, t-4, t-6, …, t-20 |
| 5 | 5 | 25 | t-5, t-10, t-15, t-20, t-25 |

Table 1 simple time series

|  |  |  |  |
| --- | --- | --- | --- |
| frequency | memory | distance | input vectors |
| 1 | 20 | 20 | t-1, t-2, t-3,..., t-19, t-20 |
| 5 | 40 | 200 | t-5, t-10, t-15, …t-200 |
| 10 | 10 | 100 | t-10, t-20, …t-100 |

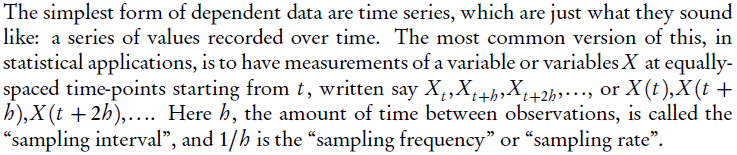
Table 2 complicate time series

frequency=5, memory=5:



notes:

For complicate time series, we need more “memory” to capture the patterns.



**frequencyis the number of observations per unit of time**.frequency is used when the series is sampled an integral number of times in each unit time interval. For example, one could use a value of 7 for frequency when the data are sampled daily, and the natural time period is a week, or 12 when the data are sampled monthly and the natural time period is a year. （12同时也是interval）

frequency returns the number of samples per unit time and deltat the time interval between observations (see [ts](http://127.0.0.1:37004/help/library/stats/help/ts)). (**frequency = 1/deltat**)

注意:

1. **重点在于根据特定频率(frequency)抽样得到样本！（而不在于样本间的时间间隔(interval)）.**
2. 该抽样频率并不考虑样本自身的周期，但抽样频率(frequency)与抽取样本数(memory)乘积应大于或等于样本周期.

**hidden layer**

a ‘rule of thumb’ called the *Baum-Haussler rule*. This states that;

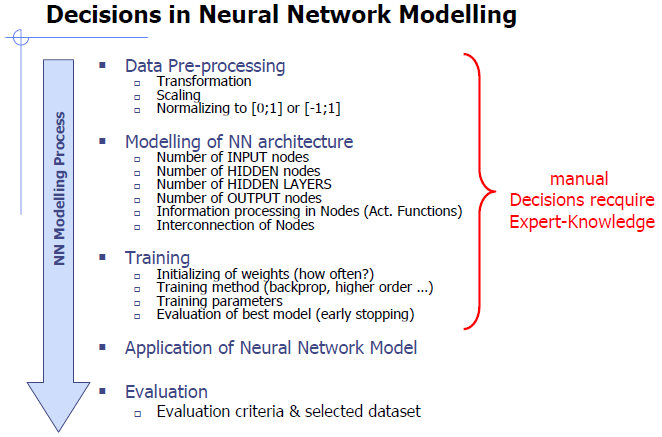


Where Nhidden is the number of hidden nodes, Ntrain is the number of training patterns, Etolerance is the error we desire of the network, Ninput and Noutput are the number of input and output nodes respectively. This rule of thumb *generally* ensures that the network generalises rather than memorises the problem.

**output layer**

we are only making a ‘one step ahead’ prediction network so we only need one output node.

##### flow-charts



Let's assume you have monthly data recorded over several years, so you have 36 values. Let's also assume that you only care about predicting one month (value) in advance.

1. Exploratory data analysis: Apply some of the traditional time series analysis methods to estimate the lag dependence in the data (e.g. auto-correlation and partial auto-correlation plots, transformations, differencing). Let's say that you find a given month's value is correlated with **the past three month**'s data but not much so beyond that. (deciding the "sliding window")
2. Partition your data into training and validation sets: Take the first 24 points as your training values and the remaining points as the validation set.
3. Create the neural network layout: You'll take the past three month's values as inputs and you want to predict the next month's value. So, you need a neural network with an input layer containing three nodes and an output layer containing one node. You should probably have a hidden layer with at least a couple of nodes. Unfortunately, picking the number of hidden layers, and their respective number of nodes, is not something for which there are clear guidelines. I'd start small, like 3:2:1.
4. Create the training patterns: Each training pattern will be four values, with the first three corresponding to the input nodes and the last one defining what the correct value is for the output node. For example, if your training data are values

x1,x2…,x24

then

pattern1:x1,x2,x3,x4

pattern2:x2,x3,x4,x5

…

pattern21:x21,x22,x23,x24

1. Train the neural network on these patterns
2. Test the network on the validation set (months 25-36): Here you will pass in the three values the neural network needs for the input layer and see what the output node gets set to. So, to see how well the trained neural network can predict month 32's value you'll pass in values for months 29, 30, and 31

In the example above, you may find that 21 training patterns is not enough; different input data transformations lead to a better/worse forecasts; varying the number of hidden layers and hidden layer nodes greatly affects forecasts; etc. The exploratory work to determine what that structure is is often the most time consuming and difficult part.

This sketches out the main point: you need to create training patterns that reasonably contain the correlation structure of the series you are trying to forecast.

##### Minimum sample size

1. There should be approximately 30 times more training cases than the number of weights.
2. General generalization rule: there should be 10 times more training cases than the VC dimension of the hypothesis set. In NN case the VC dimension is usually assumed to be around the number of weights, so you should have 10 times more training cases than the weights.

Sontag (1998) shows VC dimensions as functions of P for different type of networks and different activation functions, if you need a strict mathematical bound.

<http://www.mit.edu/~esontag/FTP_DIR/vc-expo.pdf>

reference:

<ftp://ftp.sas.com/pub/neural/FAQ3.html#A_hu>

Dr. Abu-Mostafa’s [book](http://amlbook.com/)

##### conclusions

1. Practical experience indicate that simpler models, in general, outperform more complex models. Simplicity means both the number of input nodes used and the number of hidden nodes selected. A majority of the best models uses zero or one hidden node, indicating simple linear autoregressive or NN models perform the best. But for unprocessed raw observations, more complex models may be necessary.

Sampling data correctly and choosing the correct network topology can have huge effects on time-series prediction, and sometimes it is more important to have a couple of hidden layers with a few nodes rather than lots of nodes on one hidden layer.

A more serious limitation of neural network is the implicit **assumption** that the statistical properties of the data generator are time dependant. If the generator is not time dependant then online learning methods have too be employed so the network can ‘track time’ in other words track the changing statistical properties of the generator.

延伸：

基于不同抽样频率的建模方法

reference:

<http://neuroph.sourceforge.net/TimeSeriesPredictionTutorial.html>

case study:

<http://neuroph.sourceforge.net/TimeSeriesGenerators.zip>

<http://neuroph.sourceforge.net/tutorials/StockMarketPredictionTutorial.html>

<http://neuroph.sourceforge.net/tutorials/ChickenPricePredictionTutorial.htm>

<http://www.arpapress.com/Volumes/Vol9Issue3/IJRRAS_9_3_16.pdf>详细

<http://stats.stackexchange.com/questions/10162/how-to-apply-neural-network-to-time-series-forecasting>

##### tricks

Forecasting with Neural Networks Tutorial SFCrone\_www.neural-forecasting.com

Business Forecasting with Artificial Neural Networks\_www.neural-forecasting.com

You need to make sure there are no outliers/leverage points in your data to skew those connections though.

make sure that your inputs are all properly conditioned. Have you orthogonalized and then re-scaled them? Caret can also do this pre-processing for you via it's pcaNNet function.

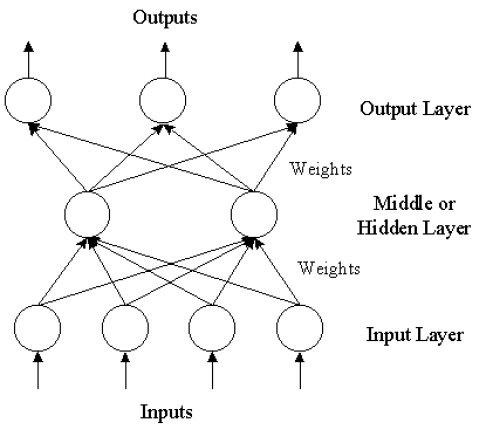
in caret is the avNNet that makes an ensemble learner out of multiple neural networks to reduce the effect of the initial seeds.

reference:

<http://stats.stackexchange.com/questions/23235/how-do-i-improve-my-neural-network-stability>

#### topologies

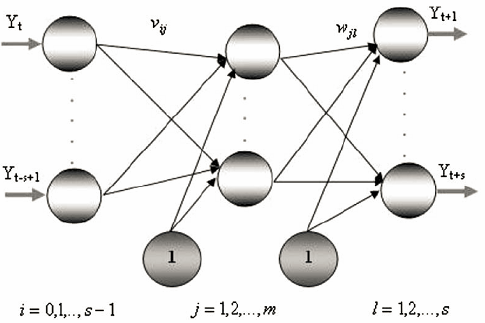
##### feed forward network



**The three-layer feed forward ANN architecture**

There may be more than one hidden layer.

##### seasonal artificial neural networks



**SANN architecture for seasonal time series**

##### nax networks

In this type of time-series problem, you would like to predict future values of a time series y(t) from past values of that time series and past values of a second time series x(t). This form of prediction is called nonlinear autoregressive with exogenous (external) input, or NARX ("NARX Network" (narxnet, closeloop)), and can be written as follows:

y(t) = f(y(t – 1), ..., y(t – d), x(t – 1), ..., (t – d))

This model could be used to predict future values of a stock or bond, based on such economic variables as unemployment rates, GDP, etc. It could also be used for system identification, in which models are developed to represent dynamic systems, such as chemical processes, manufacturing systems, robotics, aerospace vehicles, etc.

##### Recurrent networks

Recurrent networks store information about past values in the network itself.

There is a kind of neural networks named recurrent neural networks (RNNs. One advantage of using these models is you do not have to define an sliding window for the input examples. A variant of RNNs known as Long-Short Term Memory (LSTM) can potentially take into account many instances in the previous time stamps and a "forget gate" is used to allow or disallow remembering the previous results from the previous time stamps.

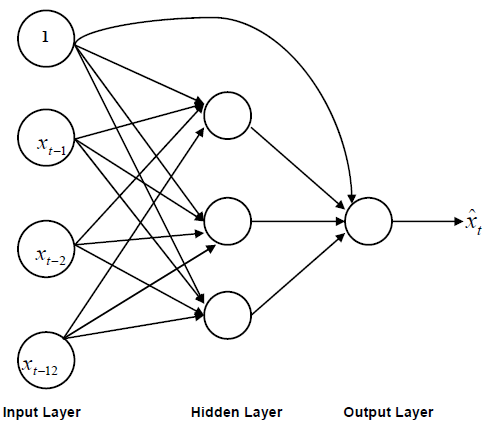
However, there are some problems that cannot be dealt with via recurrent networks alone. For example, many time series exhibit trend, meaning that the target values tend to go up over time, or that the target values tend to go down over time. The simplest methods for handling trend are difference. Sometimes it is necessary to compute differences of differences. (de-trending/de-seasonal)

software:

rnn: Recurrent Neural Network

h2o

###### time lagged neural network



**A typical TLNN architecture for monthly data**

In time-delay neural networks (Lang, Waibel, and Hinton 1990), the outputs for predicting target Y\_{i-1} are used as inputs when processing target Y\_i.

###### Jordan networks

Jordan networks (Jordan, 1986) are similar to time-delay neural networks except that the feedback is an exponential smooth of the sequence of output values.

###### Elman networks

In Elman networks (Elman, 1990), the hidden unit activations that occur when processing target Y\_{i-1} are used as inputs when processing target Y\_i.

###### LSTM

h2o

mxnet [Here](https://github.com/dmlc/mxnet/blob/master/R-package/vignettes/CharRnnModel.Rmd) you have an example of LSTM in R with this library.

I highly recommend looking at the [neural\_forecasting](http://www.neural-forecasting-competition.com/index.htm) website, which contains tons of information on neural network forecasting competitions. The [Motivations](http://www.neural-forecasting-competition.com/motivation.htm) page is especially useful.

reference :

**An Introductory Study on Time Series Modeling and Forecasting**

**neural\_FAQ\_www.neural-forecasting.com/FAQ7.html**

**Neural Networks for Time Series Processing\_Dorf96详细阅读**

##### ensemble

Ensemble methods:

The first ensemble approach created with non-varying network architectures trained using different initial random weights is not effective in improving the accuracy of prediction.

The second ensemble approach created with different neural network structures consistently perform well in a variety of situations and, hence, appear to be a preferred way to improve forecasting performance. It is more beneficial to include networks with different numbers of lags in an ensemble than to include models with different numbers of hidden nodes. This conclusion matches the expectation that it is the number of lags in the neural network model that largely determines the autocorrelation structure of a time series.

The third ensemble approach based on different partitions of the data are more effective than those developed with the full training data in out-of-sample forecasting, reducing **correlation** among forecasts made by the ensemble members by utilizing data partitioning techniques is the key to success for the neural ensemble models.

There are two different data partitioning methods. First, non-overlapping systematic subsamples from the original training series (systematic ensemble); Second, chronologically separated subsamples (serial ensemble).

reference:

Time series forecasting with neural network ensembles: an application for exchange rate predictionGP Zhang,VL Berardi