## ***AutoRegressive Models (AR) :***

1. In an autoregression model, we forecast the variable of interest using a linear combination of *past values of the variable*.
2. The term *auto*regression indicates that it is a regression of the variable against itself. Thus, an autoregressive model of order p can be written as :

AR(p) : yt = c + b1\*yt-1 + b2\*yt-2 + b3\*yt-3 + ……… + bp\*yt-p + et,

et is white noise

1. We normally restrict autoregressive models to stationary data, in which case some constraints on the values of the parameters are required. When p≥3, the restrictions are much more complicated.

## ***Moving Averages Models (MA) :***

1. Rather than using past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.
2. An moving average model of order can be written as :

MA(q) : yt = c + d1\*et-1 + d2\*et-2 + d3\*et-3 + …….. + + dq\*et-q

1. We do not *observe* the values of et, so it is not really a regression in the usual sense. Each value of yt can be thought of as a weighted moving average of the past few forecast errors.
2. It is possible to write any stationary AR(p) model as an MA(∞) model. The reverse result holds if we impose some constraints on the MA parameters. Then the MA model is called **invertible**.

## ***ARIMA(Non-seasonal) :***

1. If we combine differencing with autoregression and a moving average model, we obtain a non-seasonal ARIMA model. ARIMA is an acronym for AutoRegressive Integrated Moving Average (in this context, “integration” is the reverse of differencing).
2. The full model can be written as :

y’t = c + b1\*y’t-1 + …….. + bp\*y’t-p + d1\*et-1 + …….. + dq\*et-q + et

where y’t is the differenced series.

1. The model is called ARIMA(p, d, q), where :
   1. p = order of AR part
   2. d = degree of first differencing involved
   3. q = order of MA part
2. The auto.arima() function is useful, but anything automated can be a little dangerous, and it is worth understanding some of the behavior of the models even when you rely on an automatic procedure to choose the model for you.
3. Akaike’s Information Criterion (AIC), which was useful in selecting predictors for regression, is also useful for determining the order of an ARIMA model. It can be written as:

AIC = -2\*log(L) + 2(p + q + k + 1)

where L is the likelihood of the data and k = 1 if c != 0 & k = 0 if c = 0.

1. For ARIMA models, the corrected AIC can be written as:

AICc = AIC + 2(p + q + k + )(p + q + k + 2)/(T - p - q - k - 2)

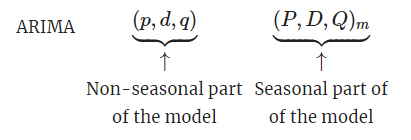
And the Bayesian Information Criterion can be written as :

BIC = AIC + [log(T) - 2](p + q + k + 1)

1. Good models are obtained by minimizing the AIC, AICc or BIC. Our preference is to use the AICc.

## ***SARIMA (Seasonal ARIMA) :***

1. A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:



where m is the number of observations per year.

1. The seasonal part of the model consists of terms that are similar to the non-seasonal components of the model, but involve backshifts of the seasonal period.

### ***SARIMAX :***

1. The Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors ([SARIMAX](https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/)) is an extension of the SARIMA model that also includes the modeling of exogenous variables.
2. Exogenous variables are also called covariates and can be thought of as parallel input sequences that have observations at the same time steps as the original series.

## ***VAR :***

1. The Vector Autoregression (VAR) method models the next step in each time series using an AR model. It is the generalization of AR to multiple parallel time series, e.g. multivariate time series.
2. The method is suitable for multivariate time series without trend and seasonal components.

## ***VARMA :***

1. The Vector Autoregression Moving-Average (VARMA) method models the next step in each time series using an ARMA model. It is the generalization of ARMA to multiple parallel time series, e.g. multivariate time series.
2. The method is suitable for multivariate time series without trend and seasonal components.

### ***VARMAX :***

1. The Vector Autoregression Moving-Average with Exogenous Regressors (VARMAX) is an extension of the VARMA model that also includes the modeling of exogenous variables. It is a multivariate version of the ARMAX method.
2. The method is suitable for multivariate time series without trend and seasonal components with exogenous variables.

## ***Simple Exponential Smoothing (SES) :***

1. The Simple Exponential Smoothing (SES) method models the next time step as an exponentially weighted linear function of observations at prior time steps.
2. The method is suitable for univariate time series without trend and seasonal components.

## ***Holt Winters Exponential Smoothing (HWES) :***

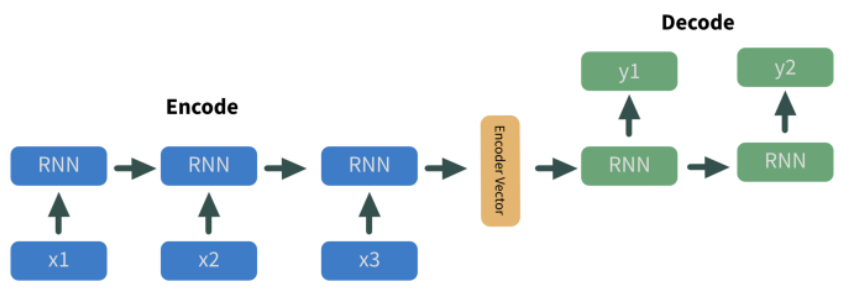
1. The [Holt Winters Exponential Smoothing](https://machinelearningmastery.com/how-to-grid-search-triple-exponential-smoothing-for-time-series-forecasting-in-python/) (HWES) also called the Triple Exponential Smoothing method models the next time step as an exponentially weighted linear function of observations at prior time steps, taking trends and seasonality into account.
2. The method is suitable for univariate time series with trend and/or seasonal components.

## ***Prophet :***

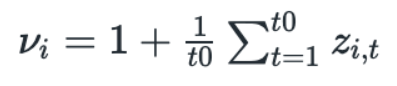
1. Prophet is an open-source library developed by Facebook and designed for automatic forecasting of univariate time series data.
2. It is easy to use and designed to automatically find a good set of hyperparameters for the model in an effort to make skillful forecasts for data with trends and seasonal structure by default.
3. Implements a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

## ***DeepAR :***

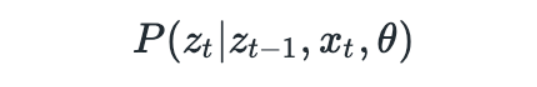
1. DeepAR is an LSTM RNN with some bells and whistles to improve accuracy on complex data.
2. Advantages :
   1. DeepAR is effective at learning seasonal dependencies with minimal tuning.
   2. DeepAR can use covariates with little training history.
   3. DeepAR makes probabilistic forecasts.
   4. DeepAR supports a wide range of likelihood functions.
3. Building on RNN architecture, DeepAR uses LSTM cells to fit our predictor variables to our variables of interest.Here’s how:
4. Sequence to Sequence Encoder-Decoder



1. Scaling to the data



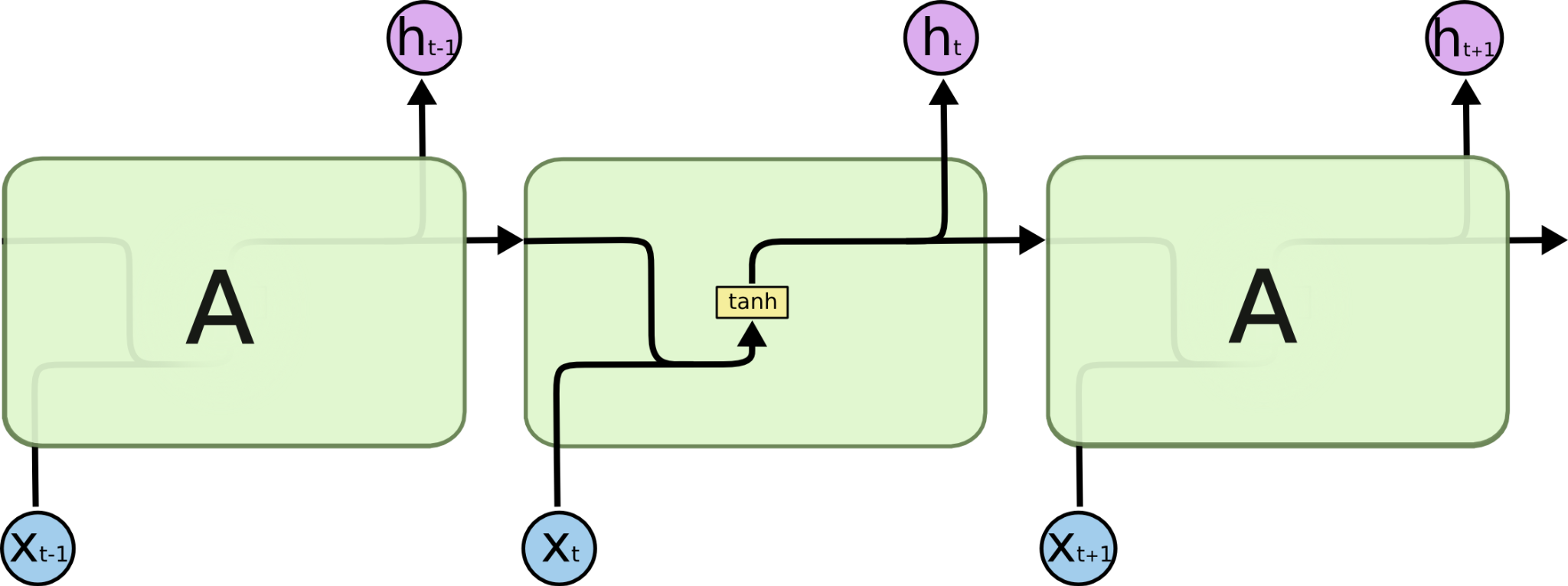
1. Fitting using a likelihood function.



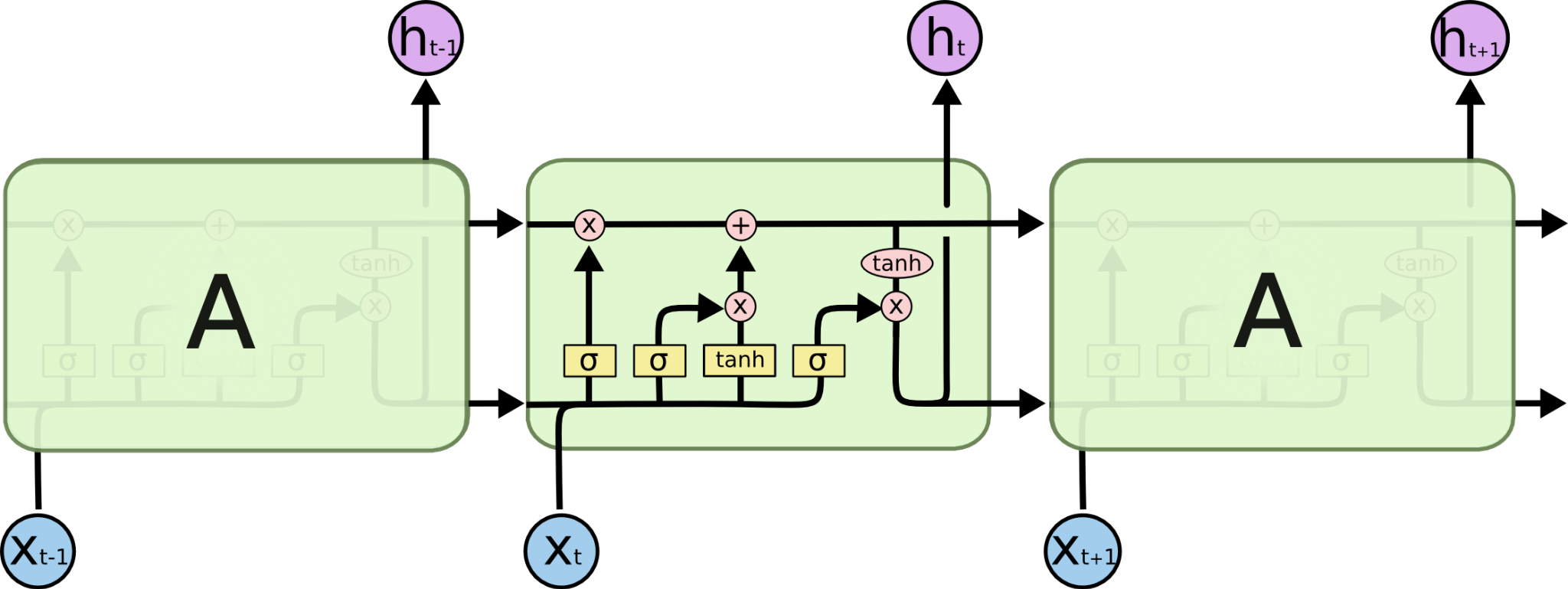
## ***LSTM:***

LSTMs (Long Short Term Memory) are a special kind of RNN, explicitly designed to avoid the long-term dependency problem in RNNs.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

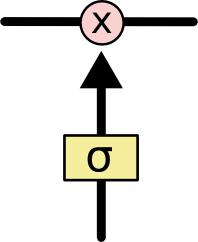


LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



Due to this aditional complexity, there are slower.

Since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

tf.keras.layers.LSTM(

units,

activation='tanh',

recurrent\_activation='sigmoid',

use\_bias=True,

kernel\_initializer='glorot\_uniform',

recurrent\_initializer='orthogonal',

bias\_initializer='zeros',

unit\_forget\_bias=True,

kernel\_regularizer=None,

recurrent\_regularizer=None,

bias\_regularizer=None,

activity\_regularizer=None,

kernel\_constraint=None,

recurrent\_constraint=None,

bias\_constraint=None,

dropout=0.0,

recurrent\_dropout=0.0,

return\_sequences=False,

return\_state=False,

go\_backwards=False,

stateful=False,

time\_major=False,

unroll=False,

\*\*kwargs

)

## 

## 

## 

## 

## ***References :***

* AR/MA/ARMA/ARIMA/SARIMA : <https://otexts.com/fpp2/arima.html>
* Prophet : [https://machinelearnin’multiplicative’gmastery.com/time-series-forecasting-with-prophet-in-python/](https://machinelearningmastery.com/time-series-forecasting-with-prophet-in-python/)
* 11 Classical Time Series Forecasting Methods : <https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/>