A Gentle Introduction to Exponential Smoothing for Time Series Forecasting in Python

by **Jason Brownlee** on April 12, 2020 in **Time Series** 68

Exponential smoothing is a time series forecasting method for univariate data that can be extended to support data with a systematic trend or seasonal component.

It is a powerful forecasting method that may be used as an alternative to the popular Box-Jenkins ARIMA family of methods.

In this tutorial, you will discover the exponential smoothing method for univariate time series forecasting.

After completing this tutorial, you will know:

- What exponential smoothing is and how it is different from other forecasting methods.
- The three main types of exponential smoothing and how to configure them.
- How to implement exponential smoothing in Python.

Kick-start your project with my new book Time Series Forecasting With Python, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.

A Gentle Introduction to Exponential Smoothing for Time Series Forecasting in Python
Photo by Wolfgang Staudt, some rights reserved.

Tutorial Overview

This tutorial is divided into 4 parts; they are:

1. What Is Exponential Smoothing?
2. Types of Exponential Smoothing
3. How to Configure Exponential Smoothing
4. Exponential Smoothing in Python

What Is Exponential Smoothing?

Exponential smoothing is a time series forecasting method for univariate data.

Time series methods like the Box-Jenkins ARIMA family of methods develop a model where the prediction is a weighted linear sum of recent past observations or lags.

Exponential smoothing forecasting methods are similar in that a prediction is a weighted sum of past observations, but the model explicitly uses an exponentially decreasing weight for past observations.

Specifically, past observations are weighted with a geometrically decreasing ratio.

> *Forecasts produced using exponential smoothing methods are weighted averages of past observations, with the weights decaying exponentially as the observations get older. In other words, the more recent the observation the higher the associated weight.*

— Page 171, Forecasting: principles and practice, 2013.

Exponential smoothing methods may be considered as peers and an alternative to the popular Box-Jenkins ARIMA class of methods for time series forecasting.

Collectively, the methods are sometimes referred to as ETS models, referring to the explicit modeling of Error, Trend and Seasonality.

Types of Exponential Smoothing

There are three main types of exponential smoothing time series forecasting methods.

A simple method that assumes no systematic structure, an extension that explicitly handles trends, and the most advanced approach that add support for seasonality.

Single Exponential Smoothing

Single Exponential Smoothing, SES for short, also called Simple Exponential Smoothing, is a time series forecasting method for univariate data without a trend or seasonality.

It requires a single parameter, called *alpha* (*a*), also called the smoothing factor or smoothing coefficient.

This parameter controls the rate at which the influence of the observations at prior time steps decay exponentially. Alpha is often set to a value between 0 and 1. Large values mean that the model pays attention mainly to the most recent past observations, whereas smaller values mean more of the history is taken into account when making a prediction.

> *A value close to 1 indicates fast learning (that is, only the most recent values influence the forecasts), whereas a value close to 0 indicates slow learning (past observations have a large influence on forecasts).*

— Page 89, Practical Time Series Forecasting with R, 2016.

Hyperparameters:

- **Alpha**: Smoothing factor for the level.

Double Exponential Smoothing

Double Exponential Smoothing is an extension to Exponential Smoothing that explicitly adds support for trends in the univariate time series.

In addition to the *alpha* parameter for controlling smoothing factor for the level, an additional smoothing factor is added to control the decay of the influence of the change in trend called *beta* (*b*).

The method supports trends that change in different ways: an additive and a multiplicative, depending on whether the trend is linear or exponential respectively.

Double Exponential Smoothing with an additive trend is classically referred to as Holt's linear trend model, named for the developer of the method Charles Holt.

- **Additive Trend**: Double Exponential Smoothing with a linear trend.

- **Multiplicative Trend**: Double Exponential Smoothing with an exponential trend.

For longer range (multi-step) forecasts, the trend may continue on unrealistically. As such, it can be useful to dampen the trend over time.

Dampening means reducing the size of the trend over future time steps down to a straight line (no trend).

> *The forecasts generated by Holt's linear method display a constant trend (increasing or decreasing) indecently into the future. Even more extreme are the forecasts generated by the exponential trend method […] Motivated by this observation […] introduced a parameter that "dampens" the trend to a flat line some time in the future.*

— Page 183, Forecasting: principles and practice, 2013.

As with modeling the trend itself, we can use the same principles in dampening the trend, specifically additively or multiplicatively for a linear or exponential dampening effect. A damping coefficient *Phi* (*p*) is used to control the rate of dampening.

- **Additive Dampening**: Dampen a trend linearly.
- **Multiplicative Dampening**: Dampen the trend exponentially.

Hyperparameters:

- **Alpha**: Smoothing factor for the level.
- **Beta**: Smoothing factor for the trend.
- **Trend Type**: Additive or multiplicative.
- **Dampen Type**: Additive or multiplicative.
- **Phi**: Damping coefficient.

Triple Exponential Smoothing

Triple Exponential Smoothing is an extension of Exponential Smoothing that explicitly adds support for seasonality to the univariate time series.

This method is sometimes called Holt-Winters Exponential Smoothing, named for two contributors to the method: Charles Holt and Peter Winters.

In addition to the alpha and beta smoothing factors, a new parameter is added called *gamma* (*g*) that controls the influence on the seasonal component.

As with the trend, the seasonality may be modeled as either an additive or multiplicative process for a linear or exponential change in the seasonality.

- **Additive Seasonality**: Triple Exponential Smoothing with a linear seasonality.
- **Multiplicative Seasonality**: Triple Exponential Smoothing with an exponential seasonality.

Triple exponential smoothing is the most advanced variation of exponential smoothing and through configuration, it can also develop double and single exponential smoothing models.

> *Being an adaptive method, Holt-Winter's exponential smoothing allows the level, trend and seasonality patterns to change over time.*

— Page 95, Practical Time Series Forecasting with R, 2016.

Additionally, to ensure that the seasonality is modeled correctly, the number of time steps in a seasonal period (*Period*) must be specified. For example, if the series was monthly data and the seasonal period repeated each year, then the Period=12.

Hyperparameters:

- **Alpha**: Smoothing factor for the level.
- **Beta**: Smoothing factor for the trend.
- **Gamma**: Smoothing factor for the seasonality.
- **Trend Type**: Additive or multiplicative.
- **Dampen Type**: Additive or multiplicative.
- **Phi**: Damping coefficient.
- **Seasonality Type**: Additive or multiplicative.
- **Period**: Time steps in seasonal period.

How to Configure Exponential Smoothing

All of the model hyperparameters can be specified explicitly.

This can be challenging for experts and beginners alike.

Instead, it is common to use numerical optimization to search for and fund the smoothing coefficients (*alpha*, *beta*, *gamma*, and *phi*) for the model that result in the lowest error.

> *[…] a more robust and objective way to obtain values for the unknown parameters included in any exponential smoothing method is to estimate them from the observed data. […] the unknown parameters and the initial values for any exponential smoothing method can be estimated by minimizing the SSE [sum of the squared errors].*

— Page 177, Forecasting: principles and practice, 2013.

The parameters that specify the type of change in the trend and seasonality, such as weather they are additive or multiplicative and whether they should be dampened, must be specified explicitly.

Exponential Smoothing in Python

This section looks at how to implement exponential smoothing in Python.

The implementations of Exponential Smoothing in Python are provided in the Statsmodels Python library.

The implementations are based on the description of the method in Rob Hyndman and George Athanasopoulos' excellent book "Forecasting: Principles and Practice," 2013 and their R implementations in their "forecast" package.

Single Exponential Smoothing

Single Exponential Smoothing or simple smoothing can be implemented in Python via the SimpleExpSmoothing Statsmodels class.

First, an instance of the *SimpleExpSmoothing* class must be instantiated and passed the training data. The *fit()* function is then called providing the fit configuration, specifically the *alpha* value called *smoothing_level*. If this is not provided or set to *None*, the model will automatically optimize the value.

This *fit()* function returns an instance of the *HoltWintersResults* class that contains the learned coefficients. The *forecast()* or the *predict()* function on the result object can be called to make a forecast.

For example:

```
# single exponential smoothing
...
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
# prepare data
data = ...
# create class
model = SimpleExpSmoothing(data)
# fit model
model_fit = model.fit(...)
# make prediction
yhat = model_fit.predict(...)
```

Double and Triple Exponential Smoothing

Single, Double and Triple Exponential Smoothing can be implemented in Python using the ExponentialSmoothing Statsmodels class.

First, an instance of the ExponentialSmoothing class must be instantiated, specifying both the training data and some configuration for the model.

Specifically, you must specify the following configuration parameters:

- **trend**: The type of trend component, as either "*add*" for additive or "*mul*" for multiplicative. Modeling the trend can be disabled by setting it to None.
- **damped**: Whether or not the trend component should be damped, either *True* or *False*.
- **seasonal**: The type of seasonal component, as either "*add*" for additive or "*mul*" for multiplicative. Modeling the seasonal component can be disabled by setting it to None.
- **seasonal_periods**: The number of time steps in a seasonal period, e.g. 12 for 12 months in a yearly seasonal structure (more here).

The model can then be fit on the training data by calling the *fit()* function.

This function allows you to either specify the smoothing coefficients of the exponential smoothing model or have them optimized. By default, they are optimized (e.g. *optimized=True*). These coefficients include:

- **smoothing_level** (*alpha*): the smoothing coefficient for the level.

- **smoothing_slope** (*beta*): the smoothing coefficient for the trend.
- **smoothing_seasonal** (*gamma*): the smoothing coefficient for the seasonal component.
- **damping_slope** (*phi*): the coefficient for the damped trend.

Additionally, the fit function can perform basic data preparation prior to modeling; specifically:

- **use_boxcox**: Whether or not to perform a power transform of the series (True/False) or specify the lambda for the transform.

The *fit()* function will return an instance of the *HoltWintersResults* class that contains the learned coefficients. The *forecast()* or the *predict()* function on the result object can be called to make a forecast.

```
# double or triple exponential smoothing
...
from statsmodels.tsa.holtwinters import ExponentialSmoothing
# prepare data
data = ...
# create class
model = ExponentialSmoothing(data, ...)
# fit model
model_fit = model.fit(...)
# make prediction
yhat = model_fit.predict(...)
```

Further Reading

This section provides more resources on the topic if you are looking to go deeper.

Tutorials

- How to Grid Search Triple Exponential Smoothing for Time Series Forecasting in Python

Books

- Chapter 7 Exponential smoothing, Forecasting: principles and practice, 2013.
- Section 6.4. Introduction to Time Series Analysis, Engineering Statistics Handbook, 2012.
- Practical Time Series Forecasting with R, 2016.

API

- Statsmodels Time Series analysis tsa
- statsmodels.tsa.holtwinters.SimpleExpSmoothing API
- statsmodels.tsa.holtwinters.ExponentialSmoothing API
- statsmodels.tsa.holtwinters.HoltWintersResults API
- forecast: Forecasting Functions for Time Series and Linear Models R package

Articles

- Exponential smoothing on Wikipedia

Summary

In this tutorial, you discovered the exponential smoothing method for univariate time series forecasting.

Specifically, you learned:

- What exponential smoothing is and how it is different from other forecast methods.
- The three main types of exponential smoothing and how to configure them.
- How to implement exponential smoothing in Python.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.