Simple Exponential Smoothing for Time Series Forecasting

Simple exponential smoothing is a simple—yet powerful—method to forecast a time series. Moreover, it is used as a building block by many other models. Let's see how it works.

This article is an extract from my book Data Science for Supply Chain Forecasting. You can read my other articles here. I am also active on LinkedIn.

Idea

A simple exponential smoothing is one of the simplest ways to forecast a time series. The basic idea of this model is to assume that the future will be more or less the same as the (recent) past. Thus, the only pattern that this model will learn from demand history is its level (you can learn about more complex models on my blog or in my book).

The level is the average value around which the demand varies over time. As you can observe in the figure below, the level is a smoothed version of the demand.

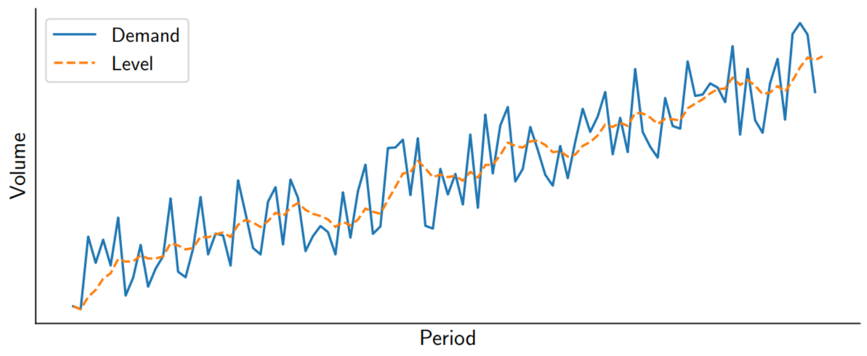

Figure 3.1: Demand level

The exponential smoothing model will then forecast the future demand as its last estimation of the level. It is essential to understand that there is no definitive mathematical definition of the level. Instead, it is up to our model to **estimate** it.

The exponential smoothing model will have some advantages compared to a simpler forecast model (such as a naïve or a moving average):

- The weight put on each observation decreases **exponentially** (we'll discuss this in more detail later) over time (the most recent observation has the highest weight). This is often better than moving average models that allocate the same weight to all the relevant historical months.
- Outliers and noise have less impact than with the naïve method.

Model

The underlying idea of an exponential smoothing model is that, at each period, the model will learn a bit from the most recent demand observation and remember a bit of the last forecast it did. The magic about this is that the last forecast populated by the model already included a part of the previous demand observation and a part of the previous forecast. And so forth. **That means that this previous forecast includes everything the model learned so far based on demand history**. The smoothing parameter (or learning rate) **alpha** will determine how much importance is given to the most recent demand observation. Let's represent this mathematically,

$$f_t = \alpha d_{t-1} + (1-\alpha) f_{t-1}$$

$$0 < \alpha \leq 1$$

What is the intuition behind this formula?

- **alpha** is a ratio (or a percentage) of how much importance the model will allocate to the most recent observation compared to the importance of demand history.
- **alpha d{t-1}** represents the previous demand observation times the learning rate. You could say that the model attaches a certain weight (alpha) to the last demand occurrence.
- **(1-alpha) f{t-1}** represents how much the model remembers from its previous forecast. Note that this is where the recursive magic happens as f{t-1} was itself defined as partially d{t-2} and f{t-2}.

There is a necessary trade-off to be made here between *learning* and *remembering,* between being reactive and being stable. If alpha is high, the model will allocate more importance to the most recent demand observation (i.e., the model will learn fast), and it will be reactive to a change in the demand level. But it will also be sensitive to outliers and noise. On the other hand, if alpha is low, the model won't notice a change in level rapidly, but will also not overreact to noise and outliers.

Future forecast

Once we are out of the historical period, we need to populate a forecast for future periods. This is simple: the last forecast (the one based on the most recent demand observation) is simply extrapolated into the future. If we define f{t*} as the last forecast that we could make based on demand history, we simply have

$$f_{t>t*} = f_{t*}$$

Model initialization

As with every model, the question comes of the initialization of the first forecast. This simple question, unfortunately, does not have a simple answer. This will often be the case in this book: the most straightforward questions won't always have definitive and absolute answers. As we will discuss over and over, only experimentation will allow us to understand which technique works best for each dataset. Let's discuss some ideas.

Simple initialization—We initialize the first forecast (period 0) as the first demand observation. We then have

$$f_0 = d_0$$

This is a fair & straightforward way to initialize the forecast.

Average—We initialize the forecast as the average of the first *n* demand occurrences.

$$f_0 = \frac{1}{n}\sum_{t=0}^{n} d_t$$

In such a case, I would advise testing different values of *n*. It could either be set as a fixed small value (3 to 5) or as the inverse of the learning rate (1/alpha). If *n* is set as the inverse of the learning rate, this allows a smoother estimation of f_0 as the learning rate decreases. This makes sense as a low value for alpha means that we want our model to react smoothly to variations.

Data leakage

If you choose an initialization method that includes information about multiple periods ahead—for example, if you define the initial forecast as the average of the first five periods—you face a **data leakage**. This means that you provide your model with pieces of information about the future. Basically, you tell it: *“Can you provide me a forecast of the next period, knowing that the average of the demand for the next five periods is 10?”*. This is a typical example of overfitting: the model will give you a good forecast accuracy for the initial periods (that’s easy— you gave it the average demand of theses periods!) but won’t be able to replicate such accuracy in the future.

Always be cautious when you initialize the different parameters of a model not to give it too much information about the future.

Insights

Impact of alpha

The figure below shows that a forecast made with a low alpha value (here 0.1) will take more time to react to changing demand, whereas a forecast with a high alpha value (here 0.8) will closely follow the demand fluctuations.

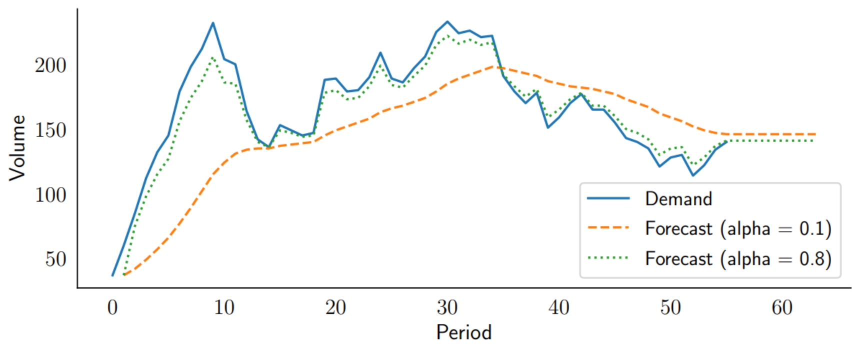

Figure 3.2: Simple smoothing

Why is it called exponential smoothing?

This model is called exponential smoothing as the weight given to each demand observation is exponentially reduced. To show this, we will start by taking back the exponential smoothing model.

$$f_t = \alpha d_{t-1} + (1 - \alpha) f_{t-1}$$

As you can see, the weight given to the most recent demand observation d{t-1} is alpha. Now let’s replace f{t-1} by its formula.

$$\begin{aligned} f_t &= \alpha d_{t-1} + (1-\alpha) f_{t-1} \\ &= \alpha d_{t-1} + (1-\alpha)\Big(\alpha d_{t-2} + (1-\alpha) f_{t-2}\Big) \end{aligned}$$

If we do a bit of algebra, we obtain the following formula:

$$f_t = \alpha d_{t-1} + \alpha(1-\alpha) d_{t-2} + (1-\alpha)^2 f_{t-2}$$

We see that the weight given to the second most recent demand observation d{t-2} is alpha(1-alpha), which is lower than the importance given to d{t-1}.

Let’s go further and replace f{t-2} by its formula.

$$\begin{aligned} f_t &= \alpha d_{t-1} + \alpha(1-\alpha) d_{t-2} + (1-\alpha)^2 \Big(\alpha d_{t-3} + (1-\alpha) f_{t-3}\Big) \\ &= \alpha d_{t-1} + \alpha(1-\alpha) d_{t-2} + \alpha(1-\alpha)^2 d_{t-3} + (1-\alpha)^3 f_{t-3} \end{aligned}$$

We see that the weight given to d{t-3} is alpha(1-alpha)², which is the weight given to d{t-2} multiplied by (1-alpha). From here, we deduce that the weight given to each further demand observation is reduced by a factor (1-alpha). This is why we call this method **exponential** smoothing.

	Moving Average					Exponential Smoothing				
	n					α				
Period	5	4	3	2	1	0.2	0.4	0.6	0.8	1
$t-1$	0.2	0.25	0.33	0.5	1	0.20	0.40	0.60	0.80	1
$t-2$	0.2	0.25	0.33	0.5		0.16	0.24	0.24	0.16	
$t-3$	0.2	0.25	0.33			0.13	0.14	0.10	0.03	
$t-4$	0.2	0.25				0.10	0.09	0.04	0.01	
$t-5$	0.2					0.08	0.05	0.02		

Table 3.1: Weight allocated to each historical period

Limitations

This simple exponential smoothing model is slightly smarter than the moving average model thanks to its smarter weighting of the historical demand observation. But it has many limitations:

- It does not project trends. We will solve this with our next model: the exponential smoothing with trend, otherwise known as **double exponential smoothing**.
- It does not recognize any seasonal pattern. We will solve this with the **triple exponential smoothing model**.
- It cannot use any external information (such as pricing or marketing expenses).

In conclusion, this first exponential smoothing model will be most likely too simple to achieve good results, but it is a good foundation block to create more complex models later.

A brief history of Holt-Winters models

The exponential smoothing models are often called "Holt-Winters," based on the names of the researchers who proposed these models. An early form of exponential smoothing forecast was initially proposed by R.G. Brown in 1956. His equations were refined in 1957 by Charles C. Holt—a US engineer from MIT and the University of Chicago—in his paper *"Forecasting Trends and Seasonals by Exponentially Weighted Averages."* The exponential smoothing models were again improved three years later by Peter Winters. Their two names were remembered and given to the different exponential smoothing techniques that we sometimes call *"Holt-Winters."*

Holt & Winters proposed different exponential smoothing models (simple, double, and triple) that can also understand & project a trend or a seasonality. This ensemble of models is then quite robust to forecast any time series. And, as Holt and Winters already explained in 1960, these forecasts only require a modest use of computation power.

Do it yourself

You can make your own simple exponential smoothing in Excel (here) or Python (here).

About the author

Nicolas Vandeput — Consultant, Founder — SupChains | LinkedIn

View Nicolas Vandeput's profile on LinkedIn, the

world's largest professional community. Nicolas ...

www.linkedin.com

Nicolas Vandeput is a supply chain data scientist specialized in demand forecasting and inventory optimization. He founded his consultancy company SupChains in 2016 and co-founded SKU Science —a fast, simple, and affordable demand forecasting platform—in 2018. Passionate about education, Nicolas is both an avid learner and enjoys teaching at universities: he has taught forecasting and inventory optimization to master students since 2014 in Brussels, Belgium. Since 2020, he is also teaching both subjects at CentraleSupelec, Paris, France. He published *Data Science for Supply Chain Forecasting* in 2018 (2nd edition in 2021) and *Inventory Optimization: Models and Simulations* in 2020.