

Chapter 2

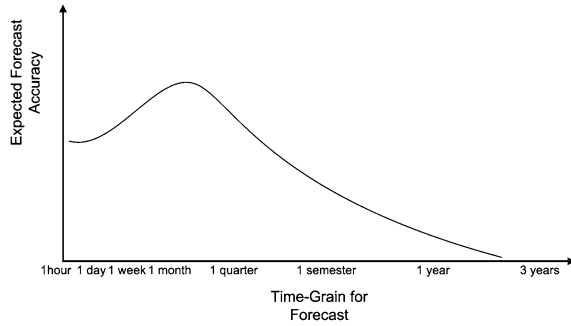
Forecasting

Since the beginning of civilizations, the ability to predict future events has been one of the most important abilities and capacities of the human mind, greatly assisting in its survival. The ability to foretell the future has always been a major source of power. On the other hand, the example of Cassandra, the ancient princess who could clearly see and prophesize catastrophic near-future events but was dismissed as insane by her people, underscores the importance of the fact that the forecaster must not only be able to make accurate forecasts, but also convince others of the accuracy of her/his forecasts. In today's world, the ability to accurately forecast near-term as well as medium-term events such as demand for existing or new products is among the most crucial capacities of an enterprise. In general, the use of forecasts falls under one of three major types: (a) economic forecasts, which attempt to measure and predict macro-economic quantities such as business cycles, inflation rates, money supply and currency exchange rates, (b) technological forecasts, whose main purpose is to predict imminent and upcoming technological break-through and innovation, and to a lesser degree market penetration of completely new products and (c) demand forecasts, whose main purpose is to predict short-and medium term sales of existing products, whose sales' history exists and is accurately recorded.

Regardless of the shift of emphasis on pull-based production models, agile manufacturing, or demand information sharing among supply chain partners, forecasting remains an invaluable tool for planning the activities of any organization—manufacturing, financial, or otherwise. In this chapter, we examine the most successful forecasting techniques available today. These include classical statistical quantitative methods such as time-series analysis that attempt to decompose a signal into a number of components such as trend, seasonality, cycle, and random noise and determine each component as accurately as possible, causal methods such as regression, as well as new techniques including Artificial Neural Networks (ANN) and Prediction Markets (PMs). All of them are statistical tools at heart.

Before discussing the above-mentioned methods in detail, it is worth noting that regardless of the forecasting method used, a few things are always true:

Fig. 2.1 Forecasting accuracy of an aggregate variable of time as a function of the time-grain used



- (1) any forecast is by necessity only an estimate of the future, and therefore will always be wrong! The only question worth asking and answering is: “*how wrong?*”
- (2) any aggregate forecast (e.g., total demand of a product sold in different packages through different distribution channels) will be more accurate than a forecast for an individual item in the aggregation. This is intuitively easy to understand, as random fluctuations of the values of individual items would tend to “cancel out” each other, making the variability of the aggregation less than the variability of its constituents. Indeed, this intuition is statistically correct: assume n items $x_1, \dots, x_n$ make up together an aggregate y whose value we wish to estimate; further assume each item’s value is a random variable with the *same* expected value μ_x and standard deviation σ_x . Then, the random variable $y = x_1 + \dots + x_n$ has expected value $\mu_y = n\mu_x$ and standard deviation $\sigma_y = \sqrt{n}\sigma_x$ and therefore its coefficient of variation (c.v.) is $1/\sqrt{n}$ times the c.v. of the individual items (Halikias 2003), which means that as n gets larger, the variable y is much more predictable than the individual variables x_i and the percentage error of the forecast tends to zero (its relative dispersion around the mean is much less than that of the constituent variables). This also holds true about aggregations in the time dimension: the forecast of *total* demand for a particular item during the next month is likely to be more accurate than the best estimate for tomorrow’s demand, *assuming of course that each day in the month is statistically the same as any other*.
- (3) any short-term forecast is in general more accurate than forecasts for events in the far future. This is also intuitively easy to understand as the farther into the future one attempts to “see”, the more uncertainty about the course of future events enters into the picture to make the variability of more distant events much greater than that of near-future events. The forecast of a day’s demand of a product 1 year from now, is likely to be much less accurate than the forecast of tomorrow’s demand of the same product.

It is interesting to notice that the combined effect of the last two observations implies that the accuracy of an aggregate forecast as a function of the time window of the aggregation must have the shape shown in Fig. 2.1: there exists an optimal length of time for which we can make accurate forecasts; attempting to aggregate

further into the future will decrease the accuracy of the forecast because the distribution of the random variables representing the far-future events will be much wider than the distribution of the variables representing the near future events. In other words, while a forecast of next quarter's sales may be more accurate than a forecast of tomorrow's sales, a forecast of the next 3 years sales will likely be much less accurate than a forecast of next quarter's sales.

The most fundamental premise, upon which all forecasting is based on, is that the future will somehow resemble the past. Therefore, as mentioned earlier, all forecasting activities will always be wrong. However, this does not mean that any forecast is useless; on the contrary, a forecast that contains a small error can be extremely useful for Supply-Chain Management purposes. Before describing the main methods of analysis of historical data for a given quantity such as monthly demand for a product, *we describe the quantities that are used to measure the accuracy of a forecast.* Suppose we have a time-series describing a particular quantity in past times, $d_i | i = 1, 2, \dots, n$ for which we are interested in predicting its value in time $n + 1$. Let F_{n+1} be the value we forecast for the quantity d_{n+1} at time n .

The quantity

$$e_t = d_t - F_t, \quad t > 1$$

defines the (one-period or instant) forecasting error. For $t > 1$, we define the following “forecasting accuracy measures” for any method that provides the forecasts F_{t+1} at time $t = 1, 2, \dots$:

- (1) Mean deviation $MD_t = \frac{\sum_{i=2}^t e_i}{t-1}$
- (2) Mean percentage deviation $MPD_t = 100 \times \frac{\sum_{i=2}^t \frac{e_i}{d_i}}{t-1} \%$
- (3) Mean absolute deviation $MAD_t = \frac{\sum_{i=2}^t |e_i|}{t-1}$
- (4) Mean absolute percentage deviation $MAPD_t = 100 \times \frac{\sum_{i=2}^t \frac{|e_i|}{d_i}}{t-1} \%$
- (5) Mean square error $MSE_t = \frac{\sum_{i=2}^t e_i^2}{t-1}$
- (6) Root mean square error $RMSE_t = \sqrt{\frac{\sum_{i=2}^t e_i^2}{t-1}}$
- (7) Tracking signal $S_t = \frac{E_t}{MAD_t} = \frac{\sum_{i=2}^t e_i}{\sum_{i=2}^t |e_i|}$
- (8) Directional symmetry $DS_t = \frac{\sum_{i=3}^t [\text{sgn}((d_i - d_{i-1})(F_i - F_{i-1}))]}{t-2}$
- (9) U-Statistics $U_1 = \frac{\sqrt{\frac{1}{t-1} \sum_{i=2}^t e_i^2}}{\sqrt{\sum_{i=2}^t d_i^2} + \sqrt{\sum_{i=2}^t F_i^2}}, \quad U_2 = \sqrt{\sum_{i=2}^{t-1} \frac{(F_{i+1} - d_{i+1})^2}{(d_{i+1} - d_i)^2}}$

The $\text{sgn}(x)$ function takes the value 1 if x is non-negative and -1 if the argument is negative. The function x_+ is defined as the $\max\{x, 0\}$. When analyzing a time-series to produce a forecast, all of the above measures are significant and should be monitored so as to get an understanding of the accuracy of the forecast.

There is no established rule to indicate what accuracy is optimal or near-optimal, but in many practical situations, a value of MAPD_t that is less than 10% is often considered excellent—although in some situations it is possible to obtain values of MAPD_t that are well below 1% in relatively stable market environments. MAPD_t values between 10 and 20% are generally considered good while values between 20 and 30% are considered moderate. Forecasts with MAPD_t scores worse than 30% are in general poor and should be discarded.

The *Directional Symmetry* metric provides an indication of the *direction* of prediction. Its value is always in the range $[0, 1]$ and a value close to 1 indicates that the forecasting procedure produces forecasts in a direction that most of the time agrees with the direction (upward or downward) of the actual time-series, and that if the time-series is about to increase in the next period, the forecast will also be higher for the next period than the forecast produced for the current period. The *tracking signal* can be useful for identifying systematic biases in the forecasts made: if it is (significantly) greater than zero, the forecast systematically underestimates the time-series d_t , whereas if it is (significantly) less than zero, the forecast overestimates the time-series. The *Root Mean Square Error* quantity (RMSE_t) is very useful as an estimate of the standard deviation of the forecast errors, which can be derived from it using the formula

$$s_t^e = \sqrt{\frac{t-1}{t-2}} \text{RMSE}_t.$$

Under the hypothesis that the errors are symmetrically distributed around zero and unbiased, the effectiveness of the forecast procedure can be affirmed by checking whether each error e_i $i = 2, \dots, t$ is in the interval $(-3s_t^e, +3s_t^e)$ to which it should lie within with a probability 99.8%. If this test fails, the forecast process needs to be revised as the errors are most likely *not* due to random noise alone.

The (Theil's) *U*-statistics have the characteristic that the more accurate the forecast, the lower their value. The U_1 -statistic is bounded in the interval $[0, 1]$, whereas the value of the U_2 -statistic provides a measure of how much better is the forecast method to the naïve method of forecasting (discussed immediately below). Values of this statistic less than 1 indicate better accuracy than the naïve method. Both *U*-statistics represent a compromise between absolute and relative measures of forecast error.

2.1 Smoothing Methods for Time-Series Analysis

2.1.1 Naïve Forecast Method

Obviously, the easiest—and most naïve—way to forecast the value of a time-series is to think that the immediate future will be the same as the immediate past, and to assume therefore that the next value of a time-series will be the same as the last one available, i.e.

$$F_{t+1} = d_t$$

This forecast, would be optimal in terms of accuracy if the values in the time-series were coming from a stochastic process that generated values according to the formula $d_{t+1} = d_t + R_t$ where R_t are independent, identically distributed (i.i.d) random variables with zero mean and constant variance σ^2 . Such a series would have a constant expected value $E[d_t] = \mu$ and a variance $\text{Var}[d_t] = t\sigma^2$ that increases linearly in time. Such a process is called a *Random Walk*, and often arises in financial-related time-series such as stock market prices, and other financial indices. More general stochastic processes where the $R_t = d_{t+1} - d_t$ variables are not necessarily independent nor have a constant variance are called *martingales* and are of great importance in financial applications as well. However, in Supply-Chain Management, such processes are quite rare, and therefore the naïve forecast method is *rarely useful* in practical situations in this domain.

2.1.2 Cumulative Mean Method

Assume that demand for a particular product is statistically constant over the time window of interest, or more generally, assume that we are interested in predicting the value of a time-series that is an instantiation of a stochastic process that generates values around a constant mean, so that the process is described by the equation $d_t = D + R_t$ where D is an unknown constant, and R_t are independent identically distributed random variables with zero mean. Under these—rather restrictive—assumptions, assuming we know the values $d_1, \dots, d_n$ the cumulative mean method computes the *best* estimate for the value d_{n+1} by the formula

$$F_{n+1} = \frac{\sum_{i=1}^n d_i}{n}$$

If at time n , any estimates of the more distant future $n + 2, n + 3 \dots$ are absolutely required, one must necessarily set $F_{n+i} = F_{n+1}$ for all $i > 1$ as well (but in general, one should not attempt to make long-range predictions using time-series methods as the error associated with them increases very fast, independent of the method). It should be rather easy for the reader to verify that the average of the observed values is the best estimator possible for the constant D and for the next value in the time-series. The graph in Fig. 2.2 shows how close the cumulative mean estimate is to a time-series that comes from the above-mentioned stochastic process, but also how close it comes to estimating the values of a time-series that is not generated from a stationary stochastic process.

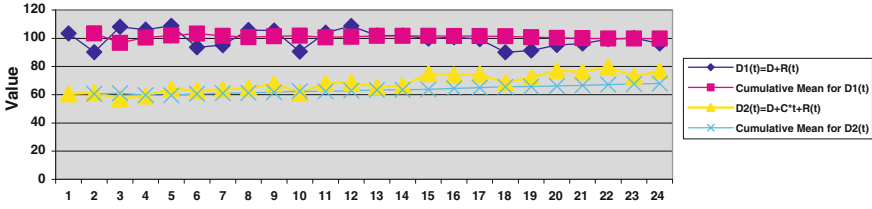


Fig. 2.2 Forecasting using the cumulative mean method: the forecast is optimal for time-series $D1(t)$, but not nearly as good when the generating stochastic process does not have a constant mean

2.1.3 Moving Average Method

One of the most well-known and established methods for demand forecasting among supply chain managers is the moving average method. The method initially looks as a computational optimization of the Cumulative Mean method, in that to compute the forecast of the next period we compute the average of a small number of the immediate past values in the time-series, thus avoiding the computation of the summation of the entire time-series. However, using only the most recent values in the time-series rather than the entire history has significant advantages when the generating stochastic process does not generate values that hover around a constant mean. If the time-series is generated from a stochastic process with a mean value that somehow drifts in time, or if the time-series is a realization of a stair-shaped stochastic process, i.e. a process that is governed by equation such as

$$d_{t+1} = D + E_{k_t} + R_t, \quad t \geq k_t$$

where D , and E_{k_t} are unknown constants, k_t is a sequence of numbers, and R_t are independent identically distributed random variables with zero mean, then the Moving Average method (with an appropriate choice of the parameter M) is a better estimator for the next value in the time-series. The Moving Average method with parameter M computes the next value in a time-series as the average of the last M observed values:

$$F_{t+1} = \frac{\sum_{i=1}^M d_{t-i+1}}{M}$$

with $M < t + 1$ of course. If estimates are needed for more distant future times, the value $F_{t+m} = F_{t+1}$ (for $m > 1$) is used as well. The method is attractive because it is very easy to understand, very easy to implement, and requires only maintaining a history of the previous M periods for each data-item to forecast. For this reason, in many surveys (Sanders and Manrodt 1994), it was shown that it ranked first among professional supply-chain managers for use in short-term and medium-term forecasts.

The graph in Fig. 2.3 shows how close the forecasts generated by the Moving Average method come to the actual values of a time-series that is generated from the stochastic process formulated above, for different values of the parameter M . However, notice in Fig. 2.4, how all the forecasts of the Moving Averages

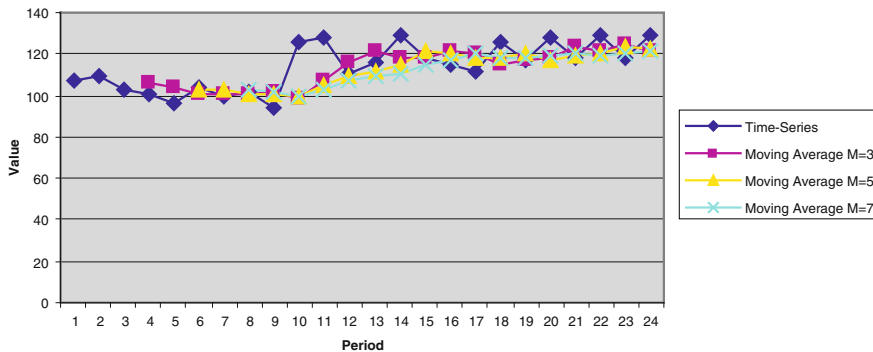


Fig. 2.3 Forecasting using the moving averages method for 3 different values of the parameter M : the generating process is a single stair-step whose values hover around a mean that changes from the value 100 to the value 120 in period 10

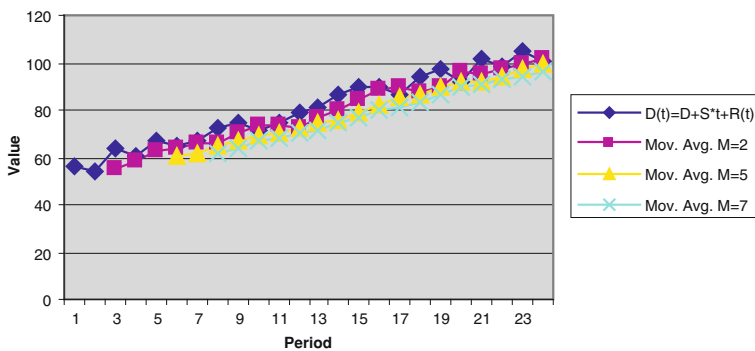


Fig. 2.4 Forecasting a time-series with a non-zero trend using the moving averages method leads to systemic errors

method consistently under-estimate a time-series that is inherently increasing (has a *trend* component). This is a major disadvantage of the method, in that by averaging the previous history, its predictions will always be in the range $[v_{\min}, v_{\max}]$ where $v_{\min}$ and $v_{\max}$, are respectively the minimum and maximum value attained in the time-series in the previous M periods.

Even if we extend the Moving Averages Method to produce a forecast as the weighted average of the last M observations, using M non-negative weights $w_1, \dots, w_M$ as follows:

$$F_{t+1} = \frac{\sum_{i=1}^M w_i d_{t-i+1}}{\sum_{i=1}^M w_i}$$

this disadvantage still persists.

Nevertheless, it is obvious from the graph that the Moving Average depicts the increasing trend in the data, despite its inability to “catch up” with the trend in the

actual forecasts. The smoothing provided by the Moving Average allows the analyst to visually determine in an instant if there exists any trend in the data even in so-called high-frequency data, i.e. time-series that fluctuates widely with a high frequency around a base trend line. The larger the value of the parameter M , the larger the “smoothing” effect that eliminates high-frequency oscillations in the time-series.

Two similar ways to improve the forecasting accuracy of the Moving Average method in the presence of trends in the data is the Moving Average with Trends method, and the Double Moving Average, both presented next.

2.1.4 Moving Average with Trends Method

To avoid the systemic problem of underestimating an inherently increasing time-series (or vice versa, over-estimating an inherently decreasing one), the following set of recursive equations is used to predict at time t , the value of a time-series at any time $t + h$ for any integer $h > 0$ given the values d_i $i = 1, \dots, t$:

$$m_t = \frac{\sum_{i=1}^M d_{t-i+1}}{M}, \quad t > M$$

$$F'_t = \begin{cases} d_t, & t \leq M \\ F'_{t-1} + \frac{6}{M(M^2-1)} ((M-1)d_t + (M+1)d_{t-M} - 2Mm_{t-1}), & t > M \end{cases}$$

$$F_{t+h} = m_t + \left(h + \frac{M-1}{2} \right) F'_t, \quad h \geq 1$$

The set of equations above (easily implementable in a spreadsheet computer program) represents a correction to the prediction provided by the Moving Average (denoted by m_t) in which the forecast for the next time-period is enhanced by a term that is analogous to the forecast for the current period which in turn is the forecast of the previous time-period plus a weighted average of the extreme points of the window considered in the time series and the average of that window. This weighted average is the best estimator of the slope of the series in a least-squares optimization sense, in which it provides the best estimate of the slope of a time-series that is linearly increasing according to the equation $d_t = D + Ct + R_t$ with R_t being independent normal random variables with zero mean. In Fig. 2.5 we show the results of the application of this method on a trended time-series, as the one in Fig. 2.4. The method fits the data better than the Moving Average method can.

2.1.5 Double Moving Average Method

Similar to the Moving Average with Trends method, the Double Moving Average method is an improvement over the classical Moving Average method that attempts to fit data generated from a linear model obeying the equation

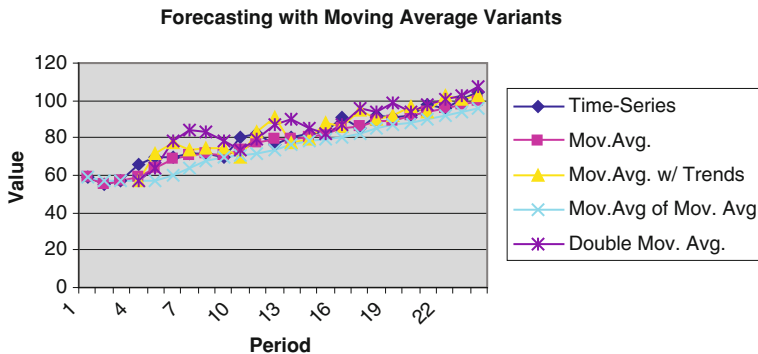


Fig. 2.5 Forecasting a time-series with a non-zero trend using the moving average, moving averages with trend method and double moving average, all with $M = 3$. the moving average of moving average assumes the time-series values for $t = 1, 2, 3$ to initialize the double moving average method. the moving average with trends method avoids systemic over- or under- estimations

$d_t = D + Ct + R_t$. The Double Moving Average computational scheme therefore evolves an estimate of the current level of the time-series and the current slope of the time-series according to the equations

$$\begin{aligned}
 F_{t+m} &= a_t + mb_t, \quad m \geq 1 \\
 a_t &= 2\gamma_t - \eta_t \\
 b_t &= \frac{2}{r-1}(\gamma_t - \eta_t) \\
 \gamma_t &= \begin{cases} \frac{1}{r} \sum_{i=0}^{r-1} d_{t-i}, & t \geq r \\ d_t, & t < r \end{cases} \\
 \eta_t &= \frac{1}{r} \sum_{i=0}^{r-1} \gamma_{t-i}, \quad t \geq r
 \end{aligned}$$

F_{t+m} is the forecast for the value of the time-series d_i at points $t + m$ for any $m > 0$ given the values d_i $i = 1, \dots, t$. Clearly, γ_t is a Moving Average with parameter r , whereas η_t is the moving average of the moving average time-series. A graphical representation of the time-series γ_t , η_t , and F_t offers some initial insight into the workings of the Double Moving Average method. The illustration in Fig. 2.5 shows how the method compares to the Moving Average with Trends method. As it turns out, the Moving Average with Trends method is superior to the Double Moving Average which tends to oscillate more than the former method.

In the following table we compare the Moving Average method with the two variants of the Moving Average that directly attempt to handle trends in the data. The Moving Average method clearly introduces systemic errors since the Tracking Signal value for period 24 is 16.92 (larger than the values produced by both variants of the method). The Moving Average with Trends method has an acceptable Tracking Signal value of -5.28 , obtains the best Mean Deviation value, and has a very good MAPD_{24} score of less than 6%.

Method ($M = 3$)	MD ₂₄	MAPD ₂₄ (%)	RMSE ₂₄	S_{24}
Mov. Avg.	2.20	3.3	3.31	16.92
Mov. Avg. w/Trends	-1.17	5.8	5.83	-5.28
Double Mov. Avg.	-3.33	7.6	7	-11.7

As we shall see in the next sections, more advanced methods (Holt's method for trended data, and Holt-Winters' method for trended and seasonal data) directly compute any trend—or periodic component—present in the series, and then synthesize the time-series as the composition of their constituents. It is interesting to notice however a common theme that appears in most of the methods to be discussed below: components, or the whole signal itself are computed as a composition of two basic estimates, an estimate of the signal itself, plus an adjustment for the previous error in the estimate, which leads to a convex combination of the estimate and the previous value of the data-series.

2.1.6 Single Exponential Smoothing Method

The easiest way to incorporate a feedback loop into the prediction system is to add to the last forecast made, a small fraction of the last forecast *error* made. The formula for Single (sometimes referred to as Simple) Exponential Smoothing Forecast (SES) is therefore the following

$$F_{t+1} = F_t + ae_t, \quad a \in (0, 1)$$

Feedback loops are extremely important in most scientific and engineering processes as they allow a system to be controlled and stabilized by making continuous adjustments to its state based on its previous performance. Fundamentally, a system that does not take into account its past performance, cannot know what adjustments to make to its processes so as to improve in the future. The Single Exponential Smoothing Method (and all Exponential Smoothing Methods to be discussed below) applies a small correction to the previous forecast if it was good and large corrections if the forecast was bad, in the direction of minimizing the error. Taking into account the definition of the error e_t , we get

$$F_{t+1} = F_t + a(d_t - F_t) = ad_t + (1 - a)F_t$$

which tells us that the next forecast is a convex combination of the last observed value in the data-series and the last forecast for the data-series. The name Exponential Smoothing derives from the fact that if we expand the formula in its convex combination form, it becomes

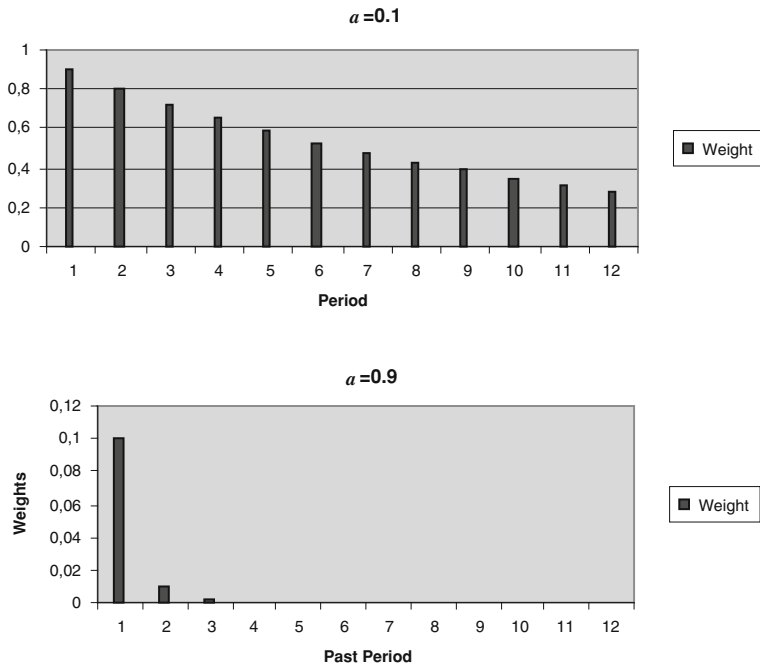


Fig. 2.6 Impact of the parameter a on the past values of the time-series in computing the SES forecast

$$\begin{aligned}
 F_{t+1} &= ad_t + (1-a)(ad_{t-1} + (1-a)F_{t-1}) = \dots \\
 &= a \sum_{i=0}^{t-1} (1-a)^i d_{t-i} + (1-a)^t F_1
 \end{aligned}$$

which shows that the Single Exponential Smoothing Method is similar to a weighted average method applying *exponentially decreasing weights* (the $(1-a)^i$ terms) to the past values of the time-series. (The sum of the exponential weights does *not* sum to one though as in weighted average methods, as the reader can easily verify.) Higher values of the parameter a imply more rapid depreciation of the past, where values of a near zero make the forecasting process behave more like the Cumulative Mean Method. Initialization of the SES method requires an initial forecast value F_1 . Usually, the choice $F_1 = d_1$ is made (other options, such as averaging demand prior to the first period and defining it as initial forecast value F_1 are also possible). In Fig. 2.6 we show how much a discounts the past values of the data-series as a function of the past.

SES methods were devised as general tools for forecasting, but work best for time-series that do *not* have inherent trends or seasonal (or business cycle) fluctuations. This is because, assuming the time-series is stationary and is generated from a process of the form $d_t = D + R_t$, it is not hard to show that the SES method

computes forecasts in such a way so that the following sum of *discounted* residual squares is minimized:

$$S' = \sum_{j=0}^{\infty} (1-a)^{j+1} (d_{t-j} - F_t)^2$$

This fact provides the theoretical justification for the method *when the time-series is generated from a (wide-sense) stationary stochastic process*. In the graphs in Fig. 2.7, we show how the value of the parameter a affects the quality of the forecast for a (trended) real-world time-series.

It is easy to see in Fig. 2.7 that higher values of parameter a allow the forecast to “catch-up” to sudden increases or decreases in the time-series much faster. To select an “optimal” value for the parameter a , one must first define a criterion which they wish to optimize. Minimizing the Mean Square Error of forecast (MSE) is often utilized as it can be expressed as a smooth polynomial of the parameter a (with degree $2t$), and local search algorithms for nonlinear optimization will easily locate a local minimizer for the criterion.

The parameter a can also be dynamically and automatically adjusted to reflect changing patterns in the data.

The *Adaptive Response Rate Single Exponential Smoothing (ARRSES) method* extends SES by assigning larger a values during periods of highly-fluctuating time-series values, and lowering the value of the parameter during periods of steady values.

The set of equations describing the application of ARRSES method are as follows

$$\begin{aligned} F_{t+1} &= a_t d_t + (1 - a_t) F_t \\ a_t &= \left| \frac{A_t}{M_t} \right| \\ A_t &= \beta e_t + (1 - \beta) A_{t-1}, \quad t > 0 \\ M_t &= \beta |e_t| + (1 - \beta) M_{t-1}, \quad t > 0 \\ A_0 &= 0, \quad M_0 = 0, \quad F_1 = 0, \quad \beta \in (0, 1) \end{aligned}$$

The method *extends* SES in that the basic SES formula is modified to include a changing a_t parameter, which is the absolute value of the ratio of a feedback loop estimate of the error and the same estimate for the absolute forecast error. The method still requires selecting a value for the single parameter β . If the error $e_i = d_i - F_i$ is consistently large in absolute value, then a_i will tend to the value 1, thus making the method more responsive to changes in the time-series whereas if the forecast errors are consistently small, the parameter values a_i will tend to zero, making the method “smooth out” random variations in the signal. In Fig. 2.8, we show how the ARRSES method forecasts the time series used in Fig. 2.7.

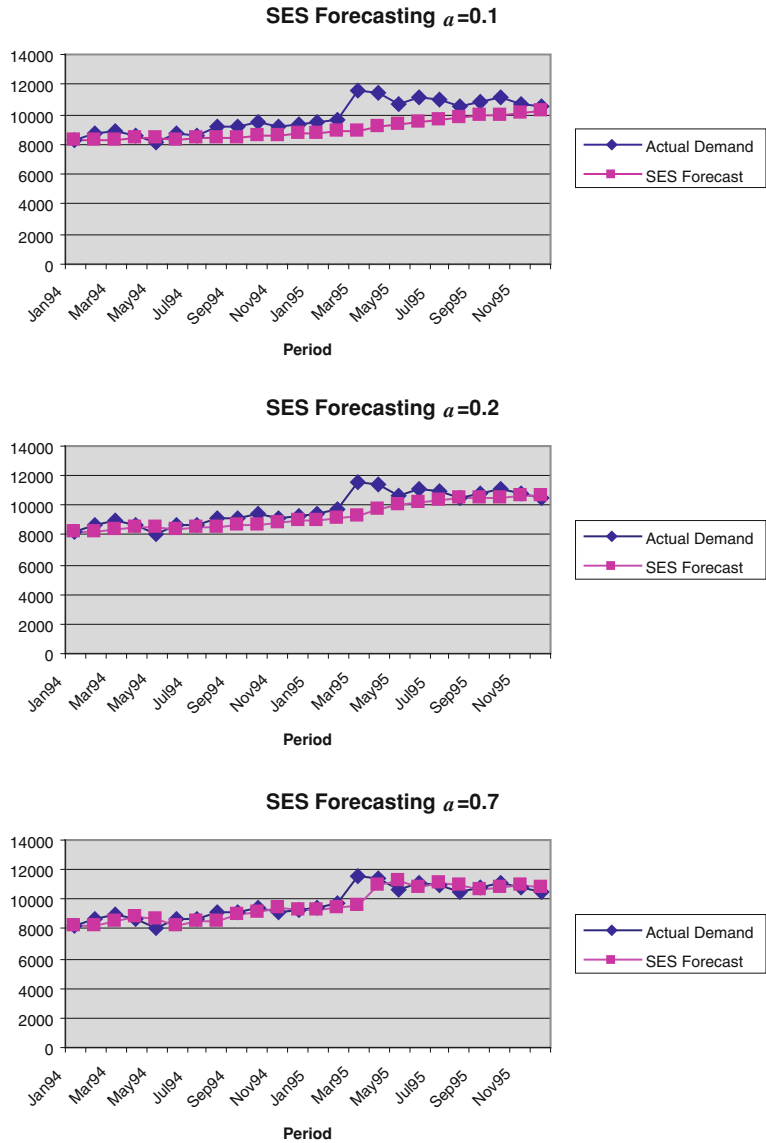


Fig. 2.7 Forecasting using the single exponential smoothing method with $\alpha = 0.1, 0.2, 0.7$. The time-series represents the monthly avicultural meat exports (in hundreds of kg) from Italy to Germany during 1994–1995 (Source: Ghiani et al. 2004)

At this point we repeat that the Single Exponential Smoothing method, independent of how the parameter α is chosen or adapted, *being a special case of a weighted average method with non-negative weights*, has the same fundamental problem of weighted averaging methods: it produces forecasts that are

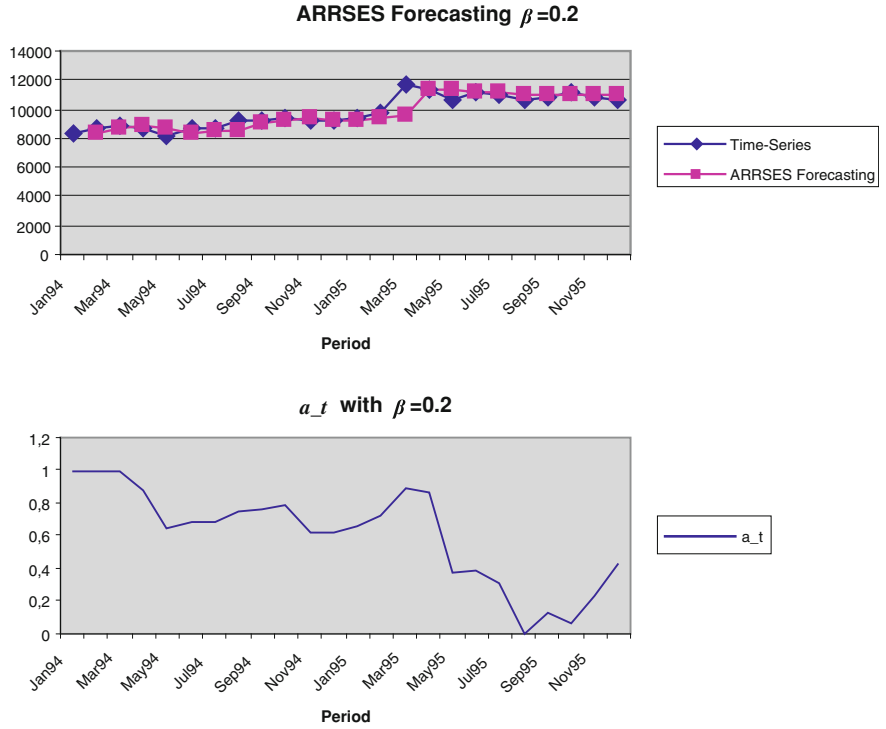


Fig. 2.8 Forecasting using the adaptive response rate single exponential smoothing method (ARRSES) with $\beta = 0.2$ and the corresponding adaptation of the values a_t . The time-series is the same as the one in Fig. 2.7

systematically under-estimating the time-series when there is an inherent positive trend, and vice versa, the forecast errors are systematically over-estimating demand when the time-series is inherently declining.

The experiment shows that the parameters can fluctuate wildly as the time-series changes its “pattern” even in the short term. In Fig. 2.9, we show that smaller values of the parameter β dampen the rate of change of the parameters a_i .

Despite the intuitive appeal of the ARRSSES method, there has been a significant body of evidence that suggests that the method does not outperform simple SES in the long run, and that, on the contrary, careless use of the ARRSSES method in practice can lead to significant forecast errors, due to long-term instabilities that the method sometimes incurs.

For this reason, many researchers and practitioners recommend great care to be taken if ARRSSES is to be applied in practice (for example, by often resetting the method, or by using it within a forecasting ensemble where other methods are used to forecast the time-series as well and a fusion scheme computes the final forecast value).

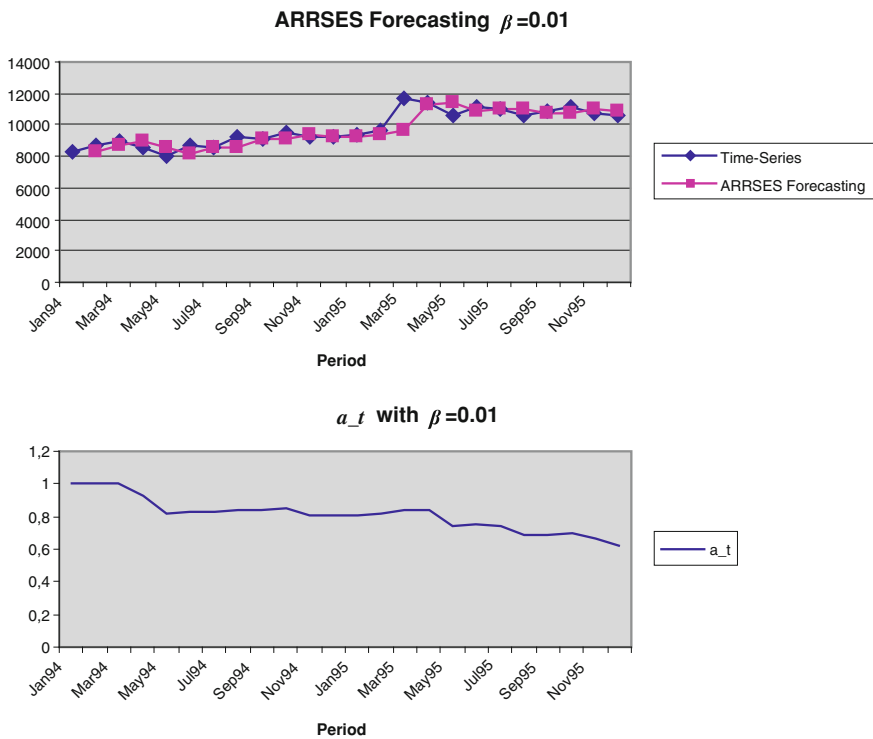


Fig. 2.9 Forecasting using the adaptive response rate single exponential smoothing method (ARRSES) with $\beta = 0.01$ and the corresponding adaptation of the values a_t . The time-series is the same as the one in Fig. 2.7

2.1.7 Multiple Exponential Smoothing Methods

Double and Triple Exponential Smoothing Methods are direct extensions of SES. The idea is to apply the same method twice or thrice on the initial time-series, in the hope that the SES method applied to a smoothed version of the initial time-series may provide better results than a single application of the method.

It can also be thought of as applying a control feedback on an already controlled forecasting system. The set of recursive equations describing the Double Exponential Smoothing Method (DES) are as follows:

$$\begin{aligned} F_{t+1} &= aF'_{t+1} + (1-a)F_t \\ F'_{t+1} &= ad_t + (1-a)F'_t \\ F_0 &= F'_0 = d_1, a \in (0, 1) \end{aligned}$$

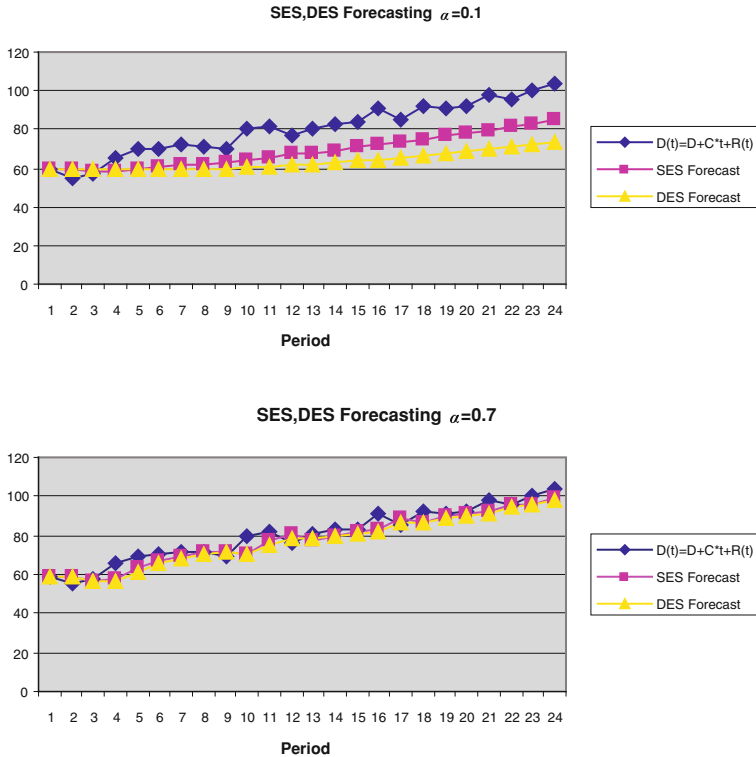


Fig. 2.10 Forecasting the next period using the double exponential smoothing method (DES) with $\alpha = 0.1, 0.7$. The original time-series the same as the one in Figs. 2.4 and 2.5

Applying this method has the same complexity as the SES method (although it requires twice as many computations as SES), and the results are still strongly affected by the choice of the parameter α , as we can see in Fig. 2.10. We check the forecasting accuracy of DES against a time-series that grows according to the equation $d_t = D + Ct + R_t$ as in the example time-series of Figs. 2.4 and 2.5. As can be seen, DES forecasts are smoothed-out versions of the smoothed-out time-series, and for this reason *lag significantly behind the original time-series when there are trends in the data*. The effect is of course less pronounced for larger values of α .

For data with a quadratic trend of the form $d_t = D + Ct + Et^2 + R_t$, Triple Exponential Smoothing can be applied. Triple Exponential Smoothing (TES) is computed via the following recursive equations:

$$F_{t+m} = 3L_t - 3L'_t + L''_t + b_tm + \frac{1}{2}c_tm^2, \quad m \geq 1$$

$$L_t = \alpha d_t + (1 - \alpha)L_{t-1}$$

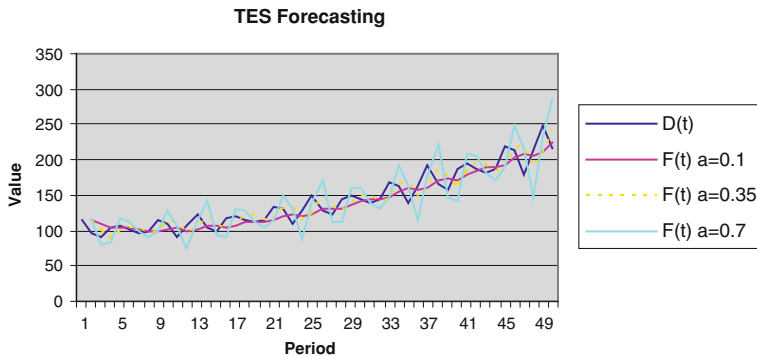


Fig. 2.11 Forecasting the next period using the triple exponential smoothing method for three different a values. the forecast with a up to 0.35 in general follows the increasing trends of the time-series, which exhibits a slow quadratic increase in time accompanied with sinusoidal oscillations. the one-period lead-time forecast using triple exponential smoothing for quadratic trends exhibits a “jerky” behavior for higher values (0.7) of the smoothing constant a

$$L'_t = aL_t + (1 - a)L'_{t-1}$$

$$L''_t = aL'_t + (1 - a)L''_{t-1}$$

$$b_t = \frac{a^2}{2(1 - a)^2} [(6 - 5a)L_t - (10 - 8a)L'_t + (4 - 3a)L''_t]$$

$$c_t = \frac{a^2}{(1 - a)^2} [L_t - 2L'_t + L''_t]$$

$$a \in (0, 1), \quad t \geq 1$$

$$L_0 = L'_0 = L''_0 = d_0$$

As mentioned already, one should not attempt to make forecasts for values F_{t+m} for values of m significantly greater than 1, as the error increases rapidly to unacceptable levels.

In Supply-Chain Management and many other real-world domains, time-series exhibiting quadratic trends are rather rare and therefore forecasting a time-series using Triple Exponential Smoothing for Quadratic Trends should never be used without extreme care, as it is likely to eventually give forecasts that are highly over-estimating the time-series. In the graph of Fig. 2.11 we show the effect of TES on a high-frequency oscillating time-series with a slow quadratic trend.

It is easy to see the “dampening” effect that the repeated applications of Exponential Smoothing have on the forecasts, but it is not easy to judge by the graphs of Fig. 2.10 alone the accuracy of the methods, mainly due to the high contribution of the random chance in the time-series. Some forecast accuracy scores for each method are shown in the table below

Method	MD ₂₄	MAPD ₂₄ (%)	RMSE ₂₄	S ₂₄
SES(a = 0.1)	6.685	12.2	19.129	9.2
DES(a = 0.1)	8.171	13.3	22.055	10.03
SES(a = 0.7)	0.805	9.6	16.691	1.4
DES(a = 0.7)	1.580	9.5	16.388	2.8

From this table, it is very easy to see that even though the MAPD_t score is reasonably good (below 15% in all cases), systemic errors are present in the forecasts when $a = 0.1$, as evidenced by the unacceptably very large values of the Tracking Signal S_t . This bias disappears when $a = 0.7$, allowing models to follow the data more closely.

2.1.8 Double Exponential Smoothing with Linear Trends Method

In the same fashion that was used to extend the Moving Average method to handle linear trends in the data, the Double Exponential Smoothing with Linear Trends method extends the DES method to take into account any linear trends in the data. The method estimates the two sequences that together implement the DES method, but provides a final forecast value that is a weighted average of the last values of the two sequences, with both positive and negative weights.

$$\begin{aligned}
 F_{t+1} &= 2F'_{t+1} - F''_{t+1} + \frac{a}{1-a} (F'_{t+1} - F''_{t+1}) \\
 F''_{t+1} &= aF'_{t+1} + (1-a)F''_t \\
 F'_{t+1} &= ad_t + (1-a)F'_t \\
 F''_0 &= F'_0 = d_1, \quad a \in (0, 1)
 \end{aligned}$$

If required to predict further data points in time, the method would obviously use the only available option $F_{t+m} = F_{t+1}$ for any $m > 1$. The method's results when forecasting the next period are graphically shown in Fig. 2.12 for different values of a , corresponding to high—or low—dampening effects. While for $a = 0.1$ the method consistently underestimates the actual signal, for a value equal to 0.7 the method responds more quickly to changes but as the random component causes high fluctuations in the data, the method consistently overshoots or undershoots the signal. Indeed, the error measures indicate that for $a = 0.7$, the MD₂₄ value is -1.779, MAPD₂₄ = 12.5% and RMSE₂₄ = 20.91. On the other hand, for $a = 0.1$, the method follows the data with MD₂₄ = 5.034, MAPD₂₄ = 11%, and RMSE₂₄ = 17.35. For $a = 0.25$, the error measures are even better, exhibiting MD₂₄ = 0.63, MAPD₂₄ = 9.3%, RMSE₂₄ = 16.8, and a tracking signal value $S_{24} = 1.17$, well within the limits of control for the forecasting process.

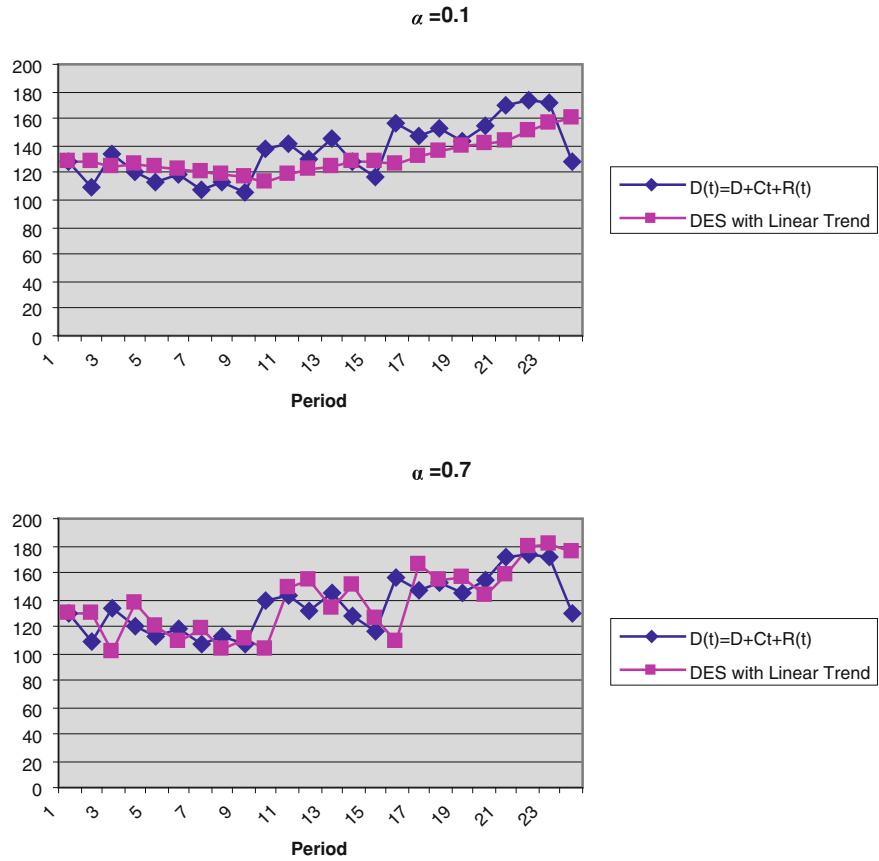


Fig. 2.12 Forecasting using the double exponential smoothing with linear trend method with $\alpha = 0.1$ and 0.7

2.1.9 The Holt Method

The DES with Linear Trends method that was discussed above is an example of an adaptation of the general idea of exponential smoothing to explicitly account for linear trends in the time-series. A better method is known as Holt's method for Forecasting; the underlying model assumes data that are generated from a stochastic process of the form $D_t = D + Ct + R_t$ so they exhibit a linear relationship with time, and attempts to estimate both the level of the data-series at each time as well as the slope of the series at that point, using a feedback loop in the spirit of exponential smoothing.

The forecast is then computed as the sum of the estimated level of the time-series at the current time L_t , plus the estimated slope b_t of the series at the current point. Estimates of the time-series at further points in time are computed as $F_{t+m} = L_t + mb_t$

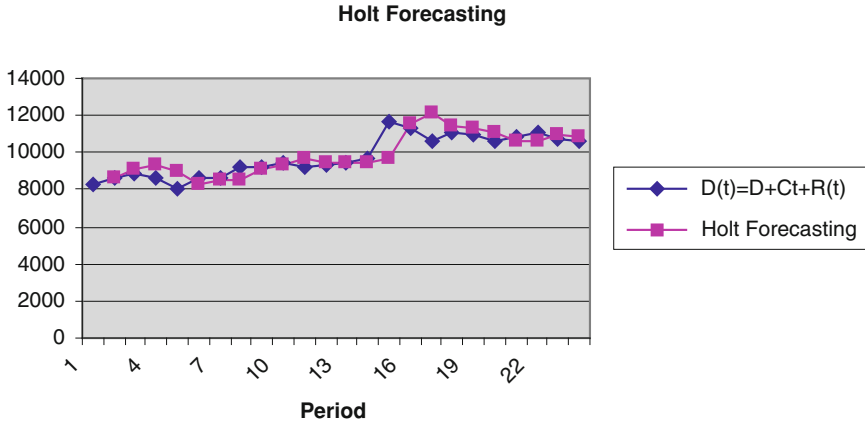


Fig. 2.13 Forecasting using the holt method with $a = 0.61$ and $\beta = 0.5$. the values for the parameters were chosen via grid search to optimize the MAPD criterion

The Holt forecasting method is given by the following set of recursive equations:

$$\begin{aligned}
 L_t &= ad_t + (1 - a)F_{t-1} \\
 b_t &= \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1} \\
 F_{t+1} &= L_t + b_t \\
 L_1 &= d_1, b_1 = d_2 - d_1 \\
 a, \beta &\in (0, 1)
 \end{aligned}$$

Estimating the level at which the time-series is done via standard single exponential smoothing, as is the estimation of the slope of the time-series. The values of the parameters a and β are usually chosen after some kind of search so as to optimize a specific criterion, which is more often than not the MSE. In Fig. 2.13 we show the results of the application of Holt's method in the time-series used in Fig. 2.12. The application of the method gives an optimal MAPD_{24} score of 11.1% which is considered good, but is nevertheless inferior to that obtained by SES or DES with linear trends methods with optimized parameters. The RMSE_{24} score for the Holt method has a value 20.61. The reason is that the particular time-series, even though generated from a stochastic process that has a linear trend, also includes a very high-noise component (the random component), which prevents the method from operating optimally.

The power of Holt's method is more evident when forecasting trended data that do not suffer from *significant* inherent random fluctuation. If we apply Holt's method to the (real-world) time-series used in Fig. 2.7, setting $a = 0.7$ and $\beta = 0.2$ (found via grid search) the results are much better: the MAPD_{24} score becomes 4.02% which indicates very good accuracy, and the RMSE_{24} score is 595.3. This score is almost exclusively due to the presence of three single large forecast errors for periods 5, 15, and 17. The errors for periods 15 and 17 are in turn caused by a large spike of the time-series value at period 15 which however

was then returned to normal, while Holt's method expected the time-series to lower their values more slowly.

The equation to estimate the slope of the data-series does not use the actual last two values of the time-series but rather the estimates of the level for the last two periods. An obvious question could be, why not use the actual values and set

$$b_t = \beta(d_t - d_{t-1}) + (1 - \beta)b_{t-1}$$

The answer is that the actual values have in them the random component which is essentially "factored out" when applying the Holt method. If one used the actual time-series values to forecast the slope at each point, setting $a = 0.5$ and $\beta = 0.55$, the ME_{24} score would be 1.82, the $MAPD_{24}$ score would be 11.2%, and the $RMSE_{24}$ metric would be 20.4. These values are only marginally worse than the Holt method.

2.1.10 The Holt–Winters Method

In Supply-Chain Management, the time-series to forecast often exhibit *cyclical fluctuations*. For example, demand for air-conditioning units typically increases significantly during summer months and decreases as the fall and then winter sets in. Such repeated cycles in demand that are directly related to the seasons of the Earth are known as the seasonal component of a time-series. A time-series can of course exhibit cyclic variations of much lower frequency, known as business cycles, but such low-frequency oscillations are useful in long-term predictions with horizons spanning several years. In any case, the models for time-series forecasting that were discussed so far are inadequate when applied to data that have inherent periodic oscillations.

The Holt–Winters' method is a direct extension of Holt's method for trended data that models the seasonal variation in a time-series as either a multiplicative component, or an additive component. *Both models assume the length of the "season" is known* and provided as input parameter s . In the *multiplicative seasonality model*, there are four equations used to obtain the forecast and they are as follows:

$$L_t = a \frac{d_t}{S_{t-s}} + (1 - a)(L_{t-1} + b_{t-1})$$

$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1}$$

$$S_t = \gamma \frac{d_t}{L_t} + (1 - \gamma)S_{t-s}$$

$$F_{t+m} = (L_t + b_t m)S_{t-s+m}, \quad t \geq s, m \geq 1$$

$$a, \beta, \gamma \in (0, 1)$$

$$b_{1 \leq i \leq s} = d_{i+1} - d_i, L_{1 \leq i \leq s} = d_i, S_{1 \leq i \leq s} = d_i s / \sum_{j=1}^s d_j$$

The first equation computes a smoothed estimate of the level of the time-series at time t , in the same way as Holt's method does except that the time-series value d_t at time t is "seasonally adjusted" by dividing it with the (multiplicative) seasonal index S_{t-s} . Therefore, the series L_t estimates the "seasonally adjusted" level of the original time-series, taking into account the trend b_t in the data, which is estimated exactly as in Holt's method by the second equation. The third equation computes the estimated seasonality index S_t of the time-series, again in the spirit of exponential smoothing methods. A rough estimate of the (multiplicative) seasonality index would be of course the value d_t/L_t so the third equation smoothes these estimates taking into account the season length. Finally, the forecast value for next period is the Holt-based estimate of the time-series multiplied by the most recent estimate available for the seasonality index of that period (S_{t-s+m}). The initialization of the procedure requires an estimate for the seasonality index for each period of the first seasonal cycle. This estimate is usually computed as simply the demand of the corresponding period divided by the mean demand throughout the first season cycle. The initial level and slope estimates are by default initialized as in the Holt method.

A special-case of the Holt–Winters' method that can be used to *forecast seasonal data that exhibit **no** trend* is easily derived by simply removing the second equation from the Holt–Winters' model and the variables b_t from the model, to obtain:

$$\begin{aligned} L_t &= a \frac{d_t}{S_{t-s}} + (1-a)L_{t-1} \\ S_t &= \gamma \frac{d_t}{L_t} + (1-\gamma)S_{t-s} \\ F_{t+m} &= L_t S_{t-s+m}, \quad t \geq s, \quad m \geq 1 \\ a, \gamma &\in (0, 1), \\ S_{1 \leq i \leq s} &= d_i, S_{1 \leq i \leq s} = d_i s / \sum_{j=1}^s d_j \end{aligned}$$

To test the multiplicative Holt–Winters method, we first apply the method to a time-series generated by a stochastic process described by the equation $D_t = (D + Ct)(2 + \sin(\omega t))(1 - R_t)$ where $R_t \sim N(0, \sigma^2)$ is a random normal variable with zero mean. With $\omega = \pi/2$, the season length equals 4. The results of applying the Holt–Winters' method to this time-series are shown in Fig. 2.14.

For this synthetic time-series example, the Holt–Winters method achieves a Mean Error $MD_{36} = -1.4995$, a $MAPD_{36}$ score of 7.82%, and an $RMSE_{36} = 12.56$. Notice the particularly good fit of the forecast after period 21, also witnessed in the reduction of the Tracking Signal S_t . These numbers are much better than Holt's method for trended data could achieve.

When the time-series oscillates in more than one frequency, the Holt–Winters method gives results that are, as expected, less favorable. Consider for example another synthetic time-series generated by an equation of the form $D_t = (D + Ct)(2 + c_1 \sin(\omega_1 t) + c_2 \sin(\omega_2 t))(1 - R_t)$ where R_t is, as before, small random noise. This time-series clearly oscillates in more than one frequency.

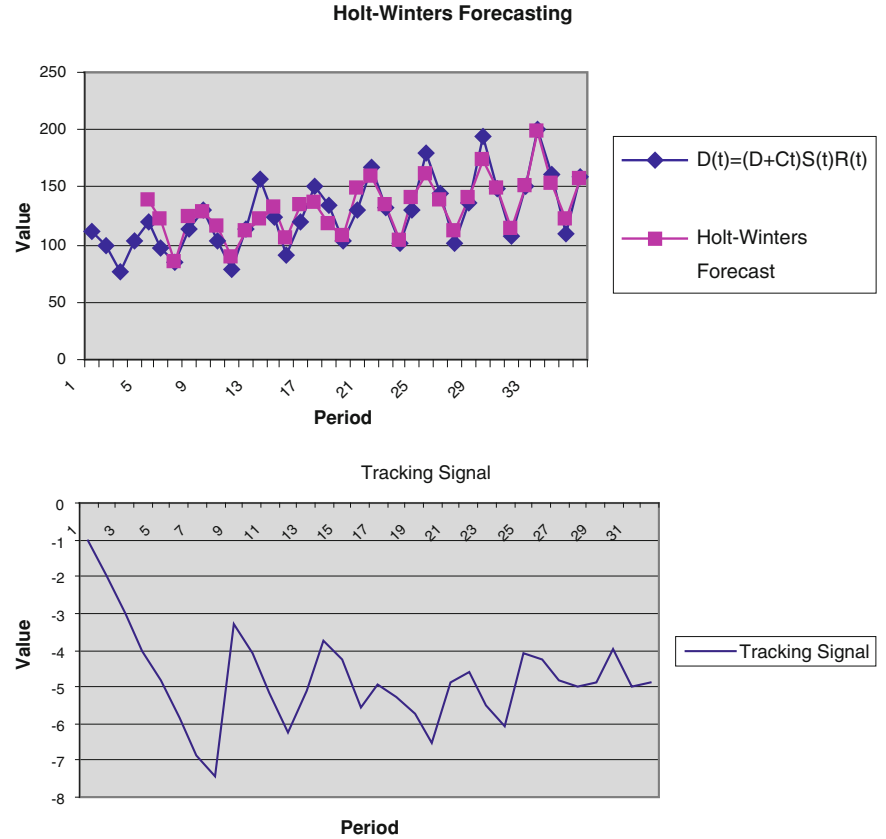


Fig. 2.14 Forecasting using the multiplicative Holt–Winters method with $s = 4$, $a = 0.79$, $\beta = 0.25$, $\gamma = 0.999$. The values for the parameters were chosen via grid search to optimize the MAPD criterion while maintaining a tracking signal within a reasonable range. The tracking signal is also plotted in the second plot

Setting $\omega_1 = \pi/2$, $\omega_2 = 2\pi/3$, and a total season length equal to 6, and carrying out a grid-search on the parameters α , β , γ to minimize the MAPD score, the parameter values are set to $a = 0.13$, $\beta = 0.3$, and $\gamma = 0.1$.

The best MAPD_{50} value attained is 17.2% which is not useless, but is less than ideal. In order to compare the Holt–Winters’ method with the Holt method, we apply Holt’s method to this time-series; the best MAPD_{50} score found by grid search when setting the parameters $a = 0.4$ and $\beta = 0.45$, is approximately 29.7% which is on the border of being useless for practical purposes, and far worse than the forecast provided by the Holt–Winters’ method that directly takes into account periodic fluctuations in the data. Figure 2.15 illustrates how the method follows the data.

The **additive seasonality Holt–Winters method** assumes that the time-series fluctuates according to a model of the form $D_t = D + Ct + S_t + R_t$

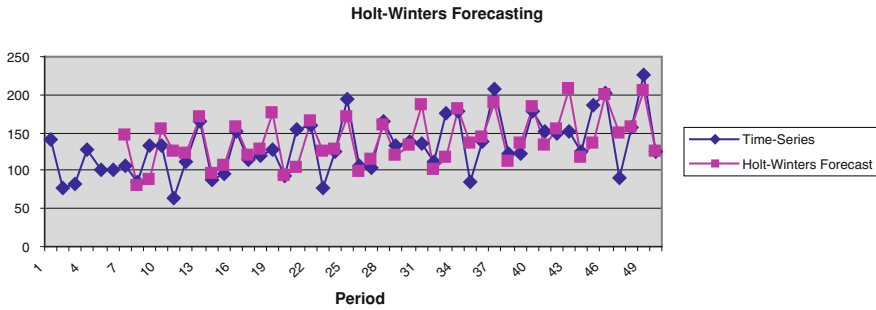


Fig. 2.15 Forecasting an oscillating time-series with more than one frequency using the multiplicative Holt–Winters method with $s = 6$, $\alpha = 0.13$, $\beta = 0.3$, $\gamma = 0.1$. The values for the parameters were chosen via grid search to optimize the MAPD criterion

The forecasting equations are as follows:

$$L_t = \alpha(d_t - S_{t-s}) + (1 - \alpha)(L_{t-1} + b_{t-1}), \quad t > s$$

$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta)b_{t-1}, \quad t > 1$$

$$S_t = \gamma(d_t - L_t) + (1 - \gamma)S_{t-s}, \quad t > s$$

$$F_{t+m} = L_t + mb_t + S_{t-s+m}, \quad m \geq 1, t \geq s$$

$$L_1 \leq i \leq s = d_i, b_1 = d_2 - d_1, S_1 \leq i \leq s = sd_i / \sum_{j=1}^s d_j$$

$$\alpha, \beta, \gamma \in (0, 1)$$

The justification of these equations follows the same rationale that was used to justify the multiplicative seasonality Holt–Winters method, with the single modifications to account for and reflect the fact that the seasonality component does not multiply the sum of level and trend component of the time-series but is rather added to it. The additive seasonality Holt–Winters method is not used very often in practice for the simple reason that the time-series that arise in real-world Supply-Chain Management problems are best modeled via a multiplicative seasonality component.

2.2 Time-Series Decomposition

Besides smoothing methods, a widely used methodology for time-series analysis and prediction is based on a decomposition of the time-series into its constituent parts, namely, (*T*) trend, (*S*) seasonality, (*C*) cycles, and (*R*) random variations. As mentioned in Sect. 2.1, the two major models for time-series analysis assume that the time-series is either the sum of the 4 parts (additive model), or alternatively, the product of the four parts (multiplicative). In the additive model, expressed by the equation

$$d_t = T_t + S_t + C_t + R_t$$

all constituent time-series are expressed in the same measurement unit. For example, if the time-series represents the mass of bananas imported to UK in a month in metric tons, each of the series T , S , C , and R express values measured in metric tons. On the other hand, in the multiplicative model, expressed by the equation

$$d_t = T_t C_t S_t R_t$$

the trend component T_t is the only time-series that is expressed in the same measurement unit as the original time-series d_t . The components S , C , and R are pure indices, i.e. simple numbers that do not express any quantity in some measurement unit. In the multiplicative model of time-series decomposition therefore, the influence of the S , C , and R components is measured as percentages and not as absolute numbers.

The idea behind time-series decomposition is to estimate and forecast as accurately as possible each of the contributing components of a time-series and then obtain a final forecast of the time-series by composing the components together again (by adding them together in an additive model, or multiplying them in case of the multiplicative model). Before discussing the technique in detail, it is important to make two observations:

- (1) the distinction between trend and cyclic components in the time-series is somewhat artificial, and for this reason, most decomposition techniques do not separate the two, but instead directly estimate a single trend-cycle component.
- (2) the time-series decomposition method, while intuitively appealing, has significant theoretical weaknesses. Despite this fact, practitioners have applied the method with significant success in a wide range of business-oriented forecasting problems, and this fact alone more than makes up for its theoretical issues.

2.2.1 Additive Model for Time-Series Decomposition

In the additive model for time-series decomposition, the trend-cycle component is first estimated. The most widely-used method for trend-cycle estimation in this model is the *centered* Moving Average method with parameter k

$$A_t = \begin{cases} \frac{\sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} d_{t+i}}{k} & t \geq \lfloor k/2 \rfloor, k \text{ odd} \\ \frac{d_{t-k/2/2} + \sum_{i=-k/2+1}^{k/2-1} d_{t+i} + d_{t+k/2/2}}{k} & t \geq k/2, k \text{ even} \end{cases}$$

Note that, in contrast to [Sect. 2.1.3](#), the value corresponding to period t is obtained by considering $k/2$ past values and $k/2$ future values of the time-series d_t .

At this point, the usual method is to consider a linear regression of the series A_t (to be discussed in detail in the next section) to compute the line which is the best mean square error estimator of the time-series. This line $T_t = a + bt$ forms the trend, and the difference $C_t = A_t - T_t$ would form the business cycle component. The slope b of the line and the intercept a are given by the formulae

$$b = \frac{N' \sum_{t=1}^{N'} t A_t - \sum_{t=1}^N t \sum_{t=1}^{N'} A_t}{N' \sum_{t=1}^N t^2 - \left(\sum_{t=1}^{N'} t \right)^2}, a = \frac{\sum_{t=1}^{N'} A_t - b \sum_{t=1}^{N'} t}{N'}$$

where

$$N' = N - \left\lfloor \frac{k}{2} \right\rfloor.$$

and N is the total length of the time-series where its values are known.

Next, the estimate of the seasonal components S_t are easily computed from the de-trended time-series $SR_t = d_t - A_t$ which must represent the seasonal component plus the random variations. Assuming that the seasonal length s is known, the seasonal components are estimated as the average of all the *homologous* values of the de-trended series SR_t at a point $t = 1, \dots, s$ in the period. The formula for computing the seasonality component is therefore as follows:

$$S'_t = \frac{\sum_{i=0}^{\lfloor N/s \rfloor} SR_{t\%s + is + 1}}{\lfloor \frac{N}{s} \rfloor + 1}, \quad t = 1, \dots$$

where $t\%s$ denotes the remainder of the *integer* division of t by s . Because the seasonality indices are assumed to complete a cycle in exactly s periods, their sum over this cycle must equal zero. However, the random white noise will almost always prevent this sum from equaling zero, and for this reason, the indices are further adjusted to sum to zero using the following formula to provide the final estimates S_t :

$$S_t = S'_t - \frac{\sum_{i=1}^s S'_i}{s}$$

Forecasting the time-series then reduces to adding back together the future estimates for the trend component T_t and the seasonal component S_t at a future time t :

$$F_t = T_t + C_{N'} + S_t, \quad t \geq N$$

Note that the estimate does not include the random variation which of course cannot be predicted, as it is assumed to be essentially white noise. However, also note that the cyclic component C_t is assumed to be constant. This is because it is very hard to predict the fluctuations of business cycles; however, by their nature business cycles oscillate in very low frequencies and therefore, in the short range, the estimate $C_t \simeq C_{N'}$ is valid for values of t close to N .

2.2.2 Multiplicative Model for Time-Series Decomposition

In an analogous fashion to the additive model for time-series decomposition, in the multiplicative model, it is assumed that $d_t = T_t C_t S_t R_t$. The procedure is almost identical to the procedure used for the additive model, with the only difference being divisions being made instead of subtractions, and thus the name “*ratio-to-moving-averages*” often used for this method.

The estimate for the trend-cycle component is identical to the additive model (Sect. 2.2.1). The formula

$$A_t = \begin{cases} \frac{\sum_{i=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} d_{t+i}}{k}, & t \geq \lfloor k/2 \rfloor, k \text{ odd} \\ \frac{d_{t-k/2/2} + \sum_{i=-k/2+1}^{k/2-1} d_{t+i} + d_{t+k/2/2}}{k}, & t \geq k/2, k \text{ even} \end{cases}$$

provides the estimate of the trend-cycle component parameterized by the value k , and a linear regression provides the optimal estimate for the trend as $T_t = a + bt$, with the parameters a, b taking on the values

$$b = \frac{N' \sum_{t=1}^{N'} t A_t - \sum_{t=1}^{N'} t \sum_{t=1}^{N'} A_t}{N' \sum_{t=1}^{N'} t^2 - \left(\sum_{t=1}^{N'} t \right)^2}, a = \frac{\sum_{t=1}^{N'} A_t - b \sum_{t=1}^{N'} t}{N'},$$

where $N' = N - \left\lfloor \frac{k}{2} \right\rfloor$

An estimate of the cyclic component is then $C_t = A_t/T_t$ for values of t up to N' .

The de-trended time-series $SR_t = d_t/A_t$ now represents the *ratio of actual to moving-averages*, and expresses the seasonal components multiplied by the random variations R_t . Assuming the season length s is known, the initial seasonal indices are computed in the same way as for the additive model, according to the formula:

$$S'_t = \frac{\sum_{i=0}^{\lfloor N/s \rfloor} SR_{t\%s+is+1}}{\left\lfloor \frac{N}{s} \right\rfloor + 1}, \quad t = 1, \dots$$

However, since in the multiplicative model, the seasonal indices are interpreted as percentages rather than absolute numbers, their sum should equal 1 (100%). Therefore the final seasonal indices are adjusted according to:

$$S_t = S'_t / \sum_{i=1}^s S'_i$$

The final forecast for the original time-series is then given as

$$F_t = T_t C_{N'} S_t, \quad t > N$$

The rationale behind setting the cyclic component's value to the computed value for time N' is the same as in the previous section.

As a final note on time-series decomposition, sometimes it may be beneficial to replace the global homologous de-trended series values averaging by a moving average, i.e. to compute the seasonal components S_t according to an equation of the form

$$S_t = \frac{\sum_{i=\lfloor N/s \rfloor - m}^{\lfloor N/s \rfloor} SR_{t\%s+is+1}}{m+1}, \quad t = 1, \dots$$

where m is a small integer representing the number of previous seasons over which the homologous time-periods will be averaged. This can be advantageous in cases where the seasonal component is not stable but changes significantly within the length of the time-series $[1, \dots, N]$.

2.2.3 Case Study

The following table is the monthly demand for electricity throughout Greece between 2002 and 2004 in MWh. We shall use this real-world time-series to draw some conclusions about the demand for energy in Greece in this time-period.

Year	Month	Extracte energy (MW h)
2002	1	3876335.96
I	2	3288467.95
	3	3485829.77
	4	3350508.41
	5	3282427.63
II	6	3737809.103
	7	4152938.09
	8	3764802.17
III	9	3336105.56
	10	3448071.245
IV	11	3218141.1
	12	3487888.82
2003	1	3369915.94
I	2	3343150
	3	3652744.9
	4	3486522.03
	5	3665734.19
II	6	3802478.11
	7	4226681.92
	8	4038016.82
III	9	3610959.24

(continued)

(continued)

Year	Month	Extracte energy (MW h)
IV	10	3538465.09
	11	3440257.73
	12	3875609.42
2004	1	4028306.21
I	2	3656400.53
	3	3640526.75
	4	3432973.48
II	5	3363492.61
	6	3713392.58
	7	4428404.18
III	8	4165200.75
	9	3813872.16
	10	3699573.23
IV	11	3590192.74
	12	

The data in the above table are plotted in Fig. 2.16 .

The data as given are in monthly granularity. Since it is well established that electrical energy demand has a strong seasonal component that is directly related to the Earth’s seasons, we shall first aggregate the data in a time-decomposition fashion in quarters. The quarterly data are presented in the following table

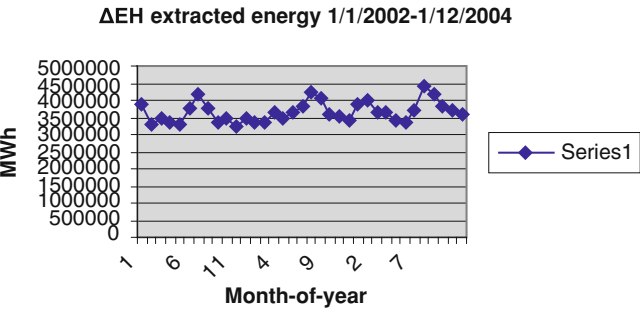


Fig. 2.16 Total electrical energy demand in Greece

Quarter	Quarterly energy
2002	
I	10650633.68
II	10370745.14
III	11253845.82
IV	10154101.17
2003	
I	10365810.84
II	10954734.33
III	11875657.98
IV	10854332.24
2004	
I	11325233.49
II	10509858.67
III	12407477.09
IV	

Next, in the aggregate, quarterly data time-series, we compute the Moving Average with period four, thus covering an entire year and all the seasons in the year, resulting in a series that has smoothed-out any seasonal variations. In the table below, we also show the centered-moving-average as the mean of two consecutive aggregate quarterly time-series values.

Quarter	Quarterly energy	Moving average of quarterly energy with $M = 4$	Centered moving average of 4 quarters
2002			
I	10650634		
II	10370745	10607331.45	10571728.6
III	11253846	10536125.74	10609124.39
IV	10154101	10682123.04	
2003			10759849.56
I	10365811	10837576.08	10925104.96
II	10954734	11012633.85	11132561.68
III	11875658		

(continued)

(continued)

Quarter	Quarterly energy	Moving average of quarterly energy with $M = 4$	Centered moving average of 4 quarters
		11252489.51	
IV	10854332		11196880.05
		11141270.6	
2004			11207747.98
I	11325233	11274225.37	
			5637112.686
II	10509859		
III	12407477		

The deviations of the actual quarterly demands from the centered 12-month Moving Average are shown in the table below.

Quarters	Centered moving average of 4 quarters	Deviations from moving average	Quarterly energy	Seasonally adjusted data
2002				
I			10650633.68	10885754.96
II			10370745.14	10255755.77
	10571729			
III		682117.2231	11253845.82	10638083.73
	10609124			
IV		-455023.2253	10154101.17	10649731.35
2003	10759850			
I		-394038.7188	10365810.84	10600932.12
	10925105			
II		29629.36688	10954734.33	10839744.96
	11132562			
III		743096.3013	11875657.98	11259895.89
	11196880			
IV		-342547.8125	10854332.24	11349962.43
2004	11207748			
I		117485.5063	11325233.49	11560354.77
	5637113			
II		4872745.984	10509858.67	10394869.3
III			12407477.09	11791715

The seasonal adjustments were made using the formulae in Sect. 2.2.1, and the operations are shown in tabulated form in the following table:

	Seasonal index estimation			
	I	II	III	IV
2002			682117.2	−455023
2003	−394039	29629.37	743096.3	−342548
2004	117485.5	394038.7		
<i>Average</i>	−138277	211834	712606.8	−398786
<i>Seasonal indices adjusted</i>	−235121	114989.4	615762.1	−495630

A plot of the seasonally adjusted quarterly electricity demand data is shown in Fig. 2.17.

The plot clearly shows an upward trend in demand during the period 2002–2004, with a spike in the third quarter of 2004 (the last point in the plot), which is partly due to the Olympic Games that took place at the time.

Finally, in Fig. 2.18, we plot the best Holt–Winters forecast on the monthly energy demand.

2.3 Regression

Consider a set of ordered pairs of observations $\{(t_1, d_1), (t_2, d_2), \dots, (t_n, d_n)\}$, where $t_i < t_j$ whenever $i < j$. If the observations $d_i = d(t_i)$ are the result of a process $d(\cdot)$ applied to the points t_i that is expected to be linear in its argument but there is the possibility of some noise to interfere with the measurement of the observations, one valid question is how to obtain the “best” line describing the data. If the noise interfering with the measurements is “white noise”, i.e. follows a Gaussian distribution with zero mean, then the line that optimally describes the observations is the line that minimizes the ℓ_2 norm of the errors, or the square errors of the observation points from that line. The optimal line that best describes the observations is the line $y = ax + b$ where the real coefficients a and b minimize the error function $\varphi(a, b) = \|at + be - d\|^2$ where $t = [t_1 \ t_2 \ \dots \ t_n]^T$, $d = [d_1 \ d_2 \ \dots \ d_n]^T$ and $e = [1 \ 1 \ \dots \ 1]^T$. The method is known as the *least squares method*. Using Fermat’s theorem, taking the partial derivatives of the function $\varphi(a, b)$ with respect to both a and b and setting them to zero in order to locate the unique (and thus global) minimum of the function, we obtain the following

$$\begin{aligned} \frac{\partial \varphi}{\partial a} &= \sum_{k=1}^n 2(at_k + b - d_k)t_k = 0 \\ \frac{\partial \varphi}{\partial b} &= \sum_{k=1}^n 2(at_k + b - d_k) = 0 \end{aligned}$$

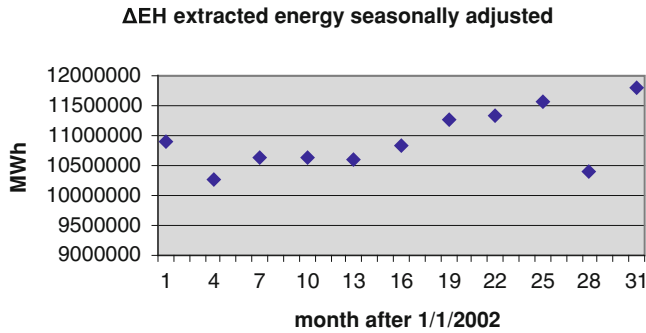


Fig. 2.17 Plot of quarterly demand for electrical energy in Greece, seasonally adjusted

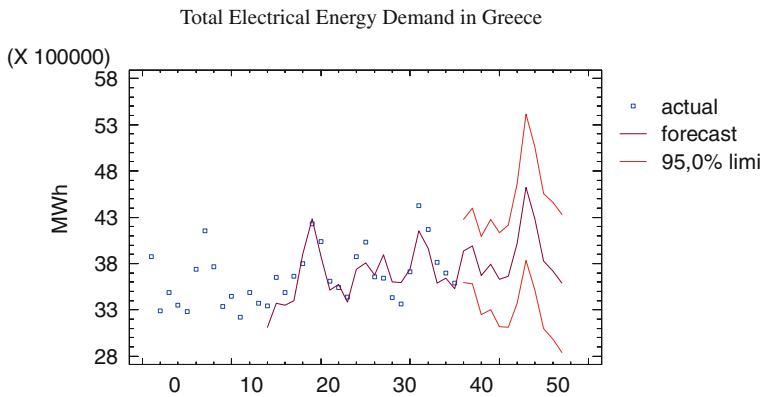


Fig. 2.18 Best Holt-Winters method forecast of monthly demand for electrical energy in Greece between January 2002 and December 2004. The figure is produced from the statistical software package StatGraphics

and rearranging terms, we obtain a system of two linear equations in the two unknowns a and b :

$$\left(\sum_{k=1}^n t_k^2 \right) a + \left(\sum_{k=1}^n t_k \right) b = \sum_{k=1}^n t_k d_k$$

$$\left(\sum_{k=1}^n t_k \right) a + nb = \sum_{k=1}^n d_k$$

Solving this 2×2 linear system yields the values for the parameters of the best line interpolating the data in the ℓ_2 norm:

$$\begin{aligned}
 a &= \frac{1}{D} \left(n \sum_{k=1}^n t_k d_k - \sum_{k=1}^n t_k \sum_{k=1}^n d_k \right) \\
 b &= \frac{1}{D} \left(\sum_{k=1}^n t_k^2 \sum_{k=1}^n d_k - \sum_{k=1}^n t_k \sum_{k=1}^n t_k d_k \right) \\
 D &= n \sum_{k=1}^n t_k^2 - \left(\sum_{k=1}^n t_k \right)^2
 \end{aligned}$$

It is trivial to verify that the unique point where $\nabla \varphi(a, b) = 0$ is a minimum for the function $\varphi(\cdot, \cdot)$ and so it is left as an exercise for the reader.

The technique of finding the best line that fits a set of observations is known as “linear regression”, and one possible use in forecasting should be immediately clear. Assume the observations form a time-series $\{(1, d_1), (2, d_2), \dots, (n, d_n)\}$. If the data seem to agree to a large degree with the best line found by linear regression, or in other words if the minimum value of the function φ is small enough, then a reasonable estimate of the value of the time series at any time $n + m$ $m > 0$ is given as $F_{n+m} = a(n + m) + b$.

A somewhat more subtle use of linear regression for forecasting purposes is as follows. Define the *correlation coefficient* between the variables t_i and d_i as

$$r = \frac{\sum_{i=1}^n (t_i - \bar{t})(d_i - \bar{d})}{\sqrt{\sum_{i=1}^n (t_i - \bar{t})^2 \sum_{i=1}^n (d_i - \bar{d})^2}}, \quad \bar{t} = \frac{\sum_{i=1}^n t_i}{n}, \quad \bar{d} = \frac{\sum_{i=1}^n d_i}{n}.$$

The correlation coefficient takes on values in the interval $[-1, 1]$ and is 1 when there is a perfect positive linear relationship between the variables t and d meaning the two are increasing or decreasing together, is -1 when there is a perfect negative linear relationship between them (i.e. whenever t increases, d decreases) and is zero if statistically there is no relationship between the two variables. Assume that

- (1) the correlation coefficient r between t and d is close to 1.
- (2) the value of the variables t_{n+i} $i > 0$ can be accurately predicted.

Then, the value of the quantity d_{n+i} for $i > 0$ can be predicted according to the formula $F_{n+i} = at_{n+i} + b$.

Even when the correlation coefficient between two variables is high, it is not necessarily true that the two variables are truly related. Spurious correlations arise when the correlation coefficient is high, but is not statistically significant—but instead, presumably it may be high because of sampling errors. To double check the validity of the hypothesis of the strong relationship between the two variables, which is essential in order to have any confidence on the forecasting results based on the regression analysis of the two variables, the following test statistic is often used:

$$t_{n-2} = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}}$$

If the value $\|t_{n-2}\| > \|t_{n-2, \alpha/2}\|$ then the hypothesis that the correlation coefficient is different from zero and is accepted at confidence level $1 - \alpha$ (Halikias 2003). The value $\|t_{n-2, \alpha/2}\|$ is simply the t-student criterion, and its value can be found in standard statistical tables for most values of interest for the parameter α ; it is also available in every statistical software package or spreadsheet software as well.

2.3.1 Generalized Regression

The method of least squares is also applicable when one wishes to compute the linear combination of a set of given functions that optimally—in the sense of least squares—matches a set of observations, i.e. a set of points in $\mathbb{R}^2$ ordered in the first dimension. If we are given a set of *basis functions* $g_j(x)$ $j = 0, \dots, m$ then the following optimization problem provides the least-squares approximation of any linear combination of the basis functions to the data $\{(t_1, d_1), (t_2, d_2), \dots, (t_n, d_n)\}$:

$$\min_{c_0, \dots, c_m} \varphi(c_0, c_1, \dots, c_m) = \sum_{k=1}^n \left[\sum_{j=0}^m c_j g_j(t_k) - d_k \right]^2$$

As before, applying Fermat's theorem to determine the minimizing point of the error function $\varphi(c)$, where c is the $m + 1$ dimensional column vector collecting the c_j parameters, we obtain

$$\frac{\partial \varphi}{\partial c_i} = \sum_{k=1}^n 2 \left[\sum_{j=0}^m c_j g_j(t_k) - d_k \right] g_i(t_k) = 0, \quad i = 0, \dots, m$$

which can be re-written as a system $Ac = b$ of $m + 1$ linear equations in $m + 1$ unknown coefficients $c_0, \dots, c_m$:

$$\sum_{j=0}^m \left[\sum_{k=1}^n g_i(t_k) g_j(t_k) \right] c_j = \sum_{k=1}^n d_k g_i(t_k), \quad i = 0, \dots, m$$

Solving the above linear system yields the optimal approximation in a least squares sense of the data using the basis functions.

2.3.2 Non-linear Least Squares Regression

It is also possible to apply the idea of least squares to optimally fit observations to any model function with parameters $\beta = [\beta_1 \ \beta_2 \ \dots \ \beta_k]^T$. Suppose we want to compute the parameters β so that the observations $\{(x_i, y_i) \ i = 0, \dots, n\}$ fit the

known function $g(x, \beta)$ as best as possible in a least squares sense. The objective is therefore to solve the following unconstrained optimization problem:

$$\min_{\beta} f(\beta) = \sum_{i=0}^n (y_i - g(x_i, \beta))^2$$

According to the First Order Necessary Conditions, the optimal β^* will satisfy the following conditions:

$$\nabla f(\beta^*) = 0 \Leftrightarrow \frac{\partial f}{\partial \beta_j}(\beta^*) = - \sum_{i=0}^n 2[y_i - g(x_i, \beta^*)] \frac{\partial g}{\partial \beta_j}(x_i, \beta^*) = 0, \quad j = 1 \dots k$$

The above is a system of k nonlinear equations in k unknowns and can be written as

$$\sum_{i=0}^n g(x_i, \beta^*) \nabla_{\beta} g(x_i, \beta^*) = \sum_{i=0}^n y_i \nabla_{\beta} g(x_i, \beta^*)$$

Obtaining a solution to the above system may be of the same complexity as the original least squares optimization problem; also, as we have seen in [Chap. 1](#), a solution to the above problem is not guaranteed to be the global minimum to the original problem. In fact, depending on convexity properties of the function $g(x, \beta)$, the solution of the nonlinear system of equations may not even be a local minimum, and therefore checking the Second Order Sufficient Conditions is required to eliminate the possibility of having located a local maximum or a saddle point of the original objective function $f(\beta)$. As discussed in the first chapter, there is in general no guarantee that a local minimizer β^* of the objective function f is the actual global minimizer, and for this reason, usually an algorithm is chosen for nonlinear optimization and is repeatedly run with a number of starting points β_0 to increase confidence in the belief that the best minimizer found is actually the global optimum. The algorithms discussed in [Sect. 1.1.1.3](#) are particularly useful in this context.

2.3.3 Exponential Model Regression

The method of least squares regression can also be applied when the data are generated from an *exponential* curve. Such data are sometimes found in economics and business-related time-series and especially when studying macroeconomic data. A highly successful model in such cases is the curve $y(x) = c_1 c_2^x$. To find the optimal fit of data on such a curve, one works with the equivalent equation $\ln y = \ln c_1 + x \ln c_2$ which is linear in the unknowns $c'_1 = \ln c_1, c'_2 = \ln c_2$, and therefore is amenable to standard linear regression. The optimal values for the parameters are given by the equations:

$$c_1 = \exp\left(\frac{1}{D}\left(\sum_{k=1}^n t_k^2 \sum_{k=1}^n \ln d_k - \sum_{k=1}^n t_k \sum_{k=1}^n t_k \ln d_k\right)\right)$$

$$c_2 = \exp\left(\frac{1}{D}\left(n \sum_{k=1}^n t_k \ln d_k - \sum_{k=1}^n t_k \sum_{k=1}^n \ln d_k\right)\right)$$

$$D = n \sum_{k=1}^n t_k^2 - \left(\sum_{k=1}^n t_k\right)^2$$

To illustrate the point, let us fit the Greek GDP in million (1980-adjusted) Euro values between the years 1948 and 1998 to the curve $y(x) = c_1 c_2^x$. The data (Halikias 2003) are tabulated in the following table

Year	GDP	Year	GDP	Year	GDP
1948	678	1965	2243	1982	5056
1949	803	1966	2380	1983	5093
1950	846	1967	2509	1984	5246
1951	920	1968	2675	1985	5424
1952	926	1969	2942	1986	5511
1953	1054	1970	3176	1987	5482
1954	1086	1971	3397	1988	5701
1955	1168	1972	3698	1989	5869
1956	1266	1973	3965	1990	5660
1957	1348	1974	3823	1991	5851
1958	1412	1975	4061	1992	5897
1959	1464	1976	4317	1993	5869
1960	1526	1977	4459	1994	6030
1961	1695	1978	4763	1995	6161
1962	1720	1979	4937	1996	6277
1963	1894	1980	5021	1997	6467
1964	2050	1981	5028	1998	6690

The plot in Fig. 2.19 shows the actual GDP data versus the best fit curve of the form $y(x) = c_1 c_2^x$.

Applying the same procedure to the data for the time-period 1948–1980, we obtain the optimal values for $c_1 = \exp(-116.1)$ and $c_2 = \exp(0.06) \approx 1.06503$ and the plot in Fig. 2.20 gives a visualization of the fit of the data to the exponential model for that period. The exponential model regression is a much better fit for the data of this period. This analysis shows clearly that Greek GDP time-series essentially “changed” its pattern after 1980.

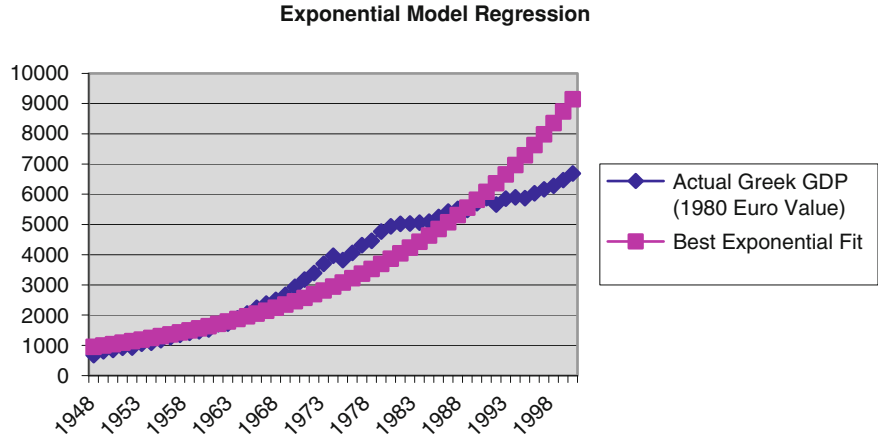


Fig. 2.19 The optimal exponential regression model to fit the Greek GDP data. Although it has a very high-correlation coefficient ($r = 0.965$) with time, it shows a clear deviation from the real data after 1968

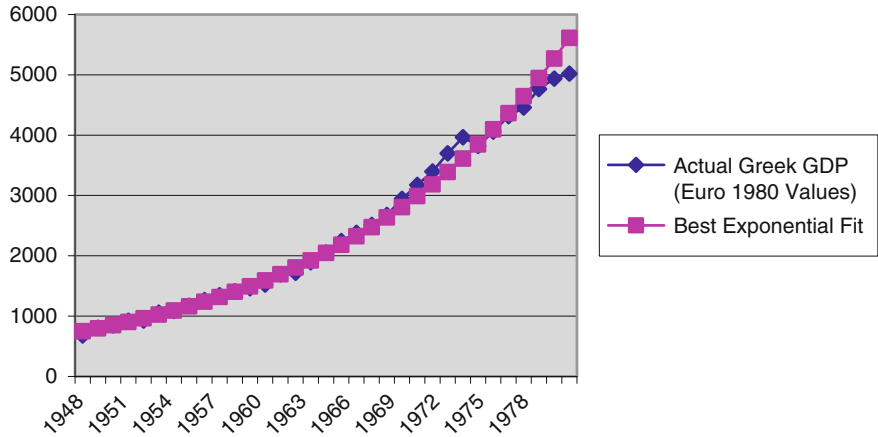


Fig. 2.20 The optimal exponential regression model to fit the Greek GDP data for the time-period 1948–1980

2.4 Auto-Regression-Based Forecasting

The regression method presented in Sect. 2.3 computes the line $y = at + b$ that minimizes the sum of the square differences of the data-points in a given time-series from the points of the line at each particular time. Exponential smoothing methods on the other hand use a number of previous data-points to forecast the immediately next values. Combining the two ideas with this ultimate goal of

forecasting the time-series in mind, a new method can be derived as the answer to the question “how to optimally, in a least squares sense, combine the last p points in the time-series so as to predict the next value”. In this Auto-Regressive (AR) model of a time-series therefore, the current value of the series d_n is expressed as a linear combination of a constant number of the values from the immediate past plus an error term (that is assumed to have zero mean and constant variance). The time-series in such a case is obviously wide-sense stationary. If the time-series is not stationary, i.e. its mean drifts in time, then an AR model cannot be expected to provide good forecasts; in such cases *integrating* the time-series by applying an appropriate differencing operator to produce a new time series $\Delta d_t = d_t - d_{t-1}$, perhaps multiple times, as in $\Delta^m d_t = \Delta(\Delta^{m-1})d_t$ may help. If the time-series $\Delta^m d_t$ for some value of $m > 0$ is stationary, the AR model can be applied on this integrated series, and then forecasts for the original time-series can be easily calculated.

To formalize the idea of AR modeling, let $d(n) = [d_n \ d_{n-1} \ \dots \ d_{n-p}]^T$ denote the column vector of the last $p + 1$ observations of the time-series $d = \{d_0, d_1, \dots, d_{N-1}\}$ at time n . We can always write $d_n = -(a_1 d_{n-1} + a_2 d_{n-2} + \dots + a_p d_{n-p}) + e_n$ where e_n is an error term that we hope to render almost always as small as possible via the optimal selection of the parameters a_i , $i = 1, \dots, p$. Therefore, we would like to compute a vector of p coefficients $a = [a_1 \ \dots \ a_p]^T$ so that the inner product $[1 \ a^T]d(n)$ is minimized in a least squares sense. Therefore, the following *least squares error problem* must be solved:

$$\min_a \sum_{i=p}^{N-1} ([1 \ a^T] d(i))^2 = \min_a [1 \ a^T] \left[\sum_{i=p}^{N-1} d(i) d(i)^T \right] \begin{bmatrix} 1 \\ a \end{bmatrix}$$

The quantity $d(i) d(i)^T$ is a $(p + 1) \times (p + 1)$ matrix whose (r, c) element is the quantity $d_{i-r+1} d_{i-c+1}$ so that the matrix $R_{p+1} = \sum_{i=p}^{N-1} d(i) d(i)^T$ is also a $(p + 1) \times (p + 1)$ matrix and has at its (k, l) position the quantity $\sum_{i=p}^{N-1} d_{i-k+1} d_{i-l+1}$. If $p \ll N$ then the elements of the matrix along *any* diagonal (i.e. the elements whose indices have a constant difference $k - l = \text{const}$) should be essentially the same. This is because the difference

$$\sum_{i=p}^{N-1} d_{i-k+1} d_{i-k+c+1} - \sum_{i=p}^{N-1} d_{i-k+2} d_{i-k+c+2} = d_{p-k+1} d_{p-k+c+1} - d_{N-k+1} d_{N-k+c+1}$$

will be much smaller than the value $\sum_{i=p}^{N-1} d_{i-k+1} d_{i-k+c+1}$ when $p \ll N$. Notice that the value $\sum_{i=p}^{N-1} d_{i-k+1} d_{i-l+1} = r_{|k-l|}$ and is independent of p or N when the time-series d_n is generated from a wide-sense stationary process.

Since the elements along the diagonals of the matrix R_{p+1} are the same, by definition the matrix is a Toeplitz matrix. From the arguments above, the matrix is also symmetric. Therefore denote the elements of the matrix R_{p+1} as follows:

$$R_{p+1} = \begin{bmatrix} r_0 & r_1 & \cdots & r_p \\ r_1 & r_0 & \cdots & r_{p-1} \\ \vdots & \cdots & \cdots & \vdots \\ r_p & r_{p-1} & \cdots & r_0 \end{bmatrix}$$

where $r_i = \sum_{j=0}^{N-i-1} d_j d_{j+i}$. The optimization problem becomes the unconstrained minimization of the function $f(a) = [1 \ a^T] R_{p+1} [1 \ a^T]^T$ which is a smooth function. So, we can rewrite the least-squares optimization problem as follows:

$$\min_{a=[a_1 \dots a_p]^T} [1 \ a^T] \begin{bmatrix} r_0 & \tilde{r}_p^T \\ \tilde{r}_p & R_p \end{bmatrix} \begin{bmatrix} 1 \\ a \end{bmatrix} = r_0 + 2a^T \tilde{r}_p + a^T R_p a$$

where $a = [a_1 \dots a_p]$, $\tilde{r}_p = [r_1 \ \dots \ r_p]^T$ and

$$R_p = \begin{bmatrix} r_0 & r_1 & \cdots & r_{p-1} \\ r_1 & r_0 & \cdots & r_{p-2} \\ \vdots & \vdots & \cdots & \vdots \\ r_{p-1} & r_{p-2} & \cdots & r_0 \end{bmatrix}$$

From Fermat's theorem, setting the gradient to zero, we obtain $\nabla f(a) = 2\tilde{r}_p + 2R_p a = 0$ or, equivalently,

$$R_p a = -\tilde{r}_p \quad (2.1)$$

The linear system (2.1) of p linear equations in p unknowns, namely the values of the coefficients $a_1, \dots, a_p$, is known as the *Yule-Walker equations*, also known as the *Wiener-Hopf equations* in the theory of linear prediction. Solving this system yields the optimal in the least squares sense predictor of the time-series d_n using a linear combination of the p last values of the signal. The minimum total error is then $E_{\min} = r_0 + a^T \tilde{r}_p$. The optimal prediction in the least-squares sense for the new value d_N as a linear combination of the last p values is given as

$$F_N = -(a_1 d_{N-1} + a_2 d_{N-2} + \cdots + a_p d_{N-p}).$$

Even though it is possible to solve the system $R_p a = -\tilde{r}_p$ using any standard algorithm for linear systems of equations, the special structure of this particular linear system of equations allows its solution via a very elegant and fast *order-recursive algorithm*, known as the *Levinson-Durbin algorithm*. Time-recursive methods also exist in the form of extensions of the ideas to be presented next. Let $\tilde{a}_m$ be the solution of the m th order system $R_m \tilde{a}_m = -\tilde{r}_m$ for any m less than or equal to p . We will develop a recursive set of equations that compute the solution of the system $R_{m+1} \tilde{a}_{m+1} = -\tilde{r}_{m+1}$ given the solution $\tilde{a}_m$. First, consider the matrix

$$J_m = \begin{bmatrix} 0 & \dots & 0 & 1 \\ \vdots & \ddots & 1 & 0 \\ 0 & 1 & \ddots & \vdots \\ 1 & 0 & \dots & 0 \end{bmatrix}$$

of dimensions $m \times m$ that when it multiplies any $m \times 1$ vector, it *reverses the order of the vector's elements*. Notice that $J_m^T = J_m$, $J_m J_m = I_m$ and that $J_m R_m = R_m J_m$ where I_m is the unit $m \times m$ matrix. Further, let $L_m = [I_m | 0]$. The matrix L_m has dimensions $m \times (m + 1)$ and has the property that when it multiplies an $(m + 1) \times 1$ vector it results in an $m \times 1$ vector that is identical to the vector that was multiplied except that the last element of the original vector is dropped. The matrix R_{m+1} can be written as

$$R_{m+1} = \left[\begin{array}{c|c} R_m & J_m \tilde{r}_m \\ \hline \tilde{r}_m^T J_m & r_0 \end{array} \right]$$

and the system $R_{m+1} \tilde{a}_{m+1} = -\tilde{r}_{m+1}$ can now be written in a decomposed form as

$$\left[\begin{array}{cc} R_m & J_m \tilde{r}_m \\ \tilde{r}_m^T J_m & r_0 \end{array} \right] \left[\begin{array}{c} L_m \tilde{a}_{m+1} \\ a_{m+1} \end{array} \right] = - \left[\begin{array}{c} \tilde{r}_m \\ r_{m+1} \end{array} \right]$$

where a_{m+1} is the last element of the vector $\tilde{a}_{m+1}$. The above system can now be written as

$$\begin{aligned} R_m L_m \tilde{a}_{m+1} + a_{m+1} J_m \tilde{r}_m &= -\tilde{r}_m \\ \tilde{r}_m^T J_m L_m \tilde{a}_{m+1} + r_0 a_{m+1} &= -r_{m+1} \end{aligned}$$

Multiplying the first equation above by J_m keeping in mind the properties of the matrix, that R_m is invertible, and the fact that $\tilde{r}_m = -R_m \tilde{a}_m$ we obtain

$$R_m (J_m L_m \tilde{a}_{m+1}) = R_m (J_m \tilde{a}_m + a_{m+1} \tilde{a}_m) \Leftrightarrow L_m \tilde{a}_{m+1} = \tilde{a}_m + a_{m+1} J_m \tilde{a}_m$$

The last equation expresses the first m components of the vector $\tilde{a}_{m+1}$ as a product of the vector $\tilde{a}_m$ and a coefficient that is linear in the value a_{m+1} . Substituting this quantity to the last equation in our system of equations we then obtain:

$$\tilde{r}_m^T (J_m \tilde{a}_m + a_{m+1} \tilde{a}_m) + r_0 a_{m+1} = -r_{m+1} \Leftrightarrow a_{m+1} = -\frac{r_{m+1} + \tilde{r}_m^T J_m \tilde{a}_m}{r_0 + \tilde{r}_m^T \tilde{a}_m}$$

At this point the $m + 1$ dimensional column vector $\tilde{a}_{m+1}$ can be expressed as a linear combination of the $m \times 1$ vector $\tilde{a}_m$. The recursive equation that obtains the solution of the $(m + 1)$ st order system of equations from the m th order solution is given by:

$$\begin{aligned}\tilde{a}_{m+1} &= \begin{bmatrix} \tilde{a}_m \\ 0 \end{bmatrix} + k_{m+1} \begin{bmatrix} J_m \tilde{a}_m \\ 1 \end{bmatrix} \\ k_{m+1} &= -\frac{r_{m+1} + \tilde{r}_m^T J_m \tilde{a}_m}{r_0 + \tilde{r}_m^T \tilde{a}_m} = -\frac{b_{m+1}}{\alpha'_m}, \quad b_{m+1} = r_{m+1} + \tilde{r}_m^T J_m \tilde{a}_m, \quad \alpha'_m = r_0 + \tilde{r}_m^T \tilde{a}_m\end{aligned}\quad (2.2)$$

It is not hard to verify that the quantities α'_m can be computed recursively as well as they obey the equation

$$\alpha'_m = \alpha'_{m-1} + k_m b_m = \alpha'_{m-1} (1 - k_m^2)$$

The above equations lead to a very fast order-recursive algorithm for the computation of the parameters of the optimal forward linear predictor of the time-series d_n . The following is a formal description of the Levinson-Durbin algorithm.

Algorithm Levinson-Durbin

Input: A time-series $d = \{d_0, \dots, d_{N-1}\}$ and the order p of the optimal least-squares linear predictor.

Output: Optimal Coefficients vector $a = [a_1 \dots a_p]^T$ of dimension $p \times 1$ to be used in forward time-series prediction $F_N = -(a_1 d_{N-1} + \dots + a_p d_{N-p})$

Begin

/*Initialization*/

1. Set $r_0 = \sum_{i=0}^{N-1} d_i^2$, $\alpha_0 = r_0$, $b_1 = r_1 = \sum_{i=0}^{N-2} d_i d_{i+1}$, $k_1 = -r_1/r_0$, $a_1^{[1]} = k_1$,

$m = 1$.

/* Loop over the orders */

2. while $m < p$ do:

a. Set $a^{[m]} = [a_1^{[m]} \dots a_m^{[m]}]^T$.

b. Set $r_{m+1} = \sum_{i=0}^{N-m} d_i d_{i+m+1}$, $\tilde{r}_m = [r_1 \dots r_m]^T$

c. Set $\alpha_m = \alpha_{m-1} + b_m k_m$.

d. If $\alpha_m \leq 0$ then ERROR (' R_{m+1} matrix is not symmetric Toeplitz').

e. Set $b_{m+1} = a^{[m]T} J_m \tilde{r}_m + r_{m+1}$.

f. Set $k_{m+1} = -b_{m+1}/\alpha_m$.

g. Set $a^{[m+1]} = \begin{bmatrix} a^{[m]} \\ 0 \end{bmatrix} + k_{m+1} \begin{bmatrix} J_m a^{[m]} \\ 1 \end{bmatrix}$.

h. Set $m=m+1$.

3. end-while

4. return $a^{[p]}$.

End

The algorithm runs in $O(p)$ time, as can be easily verified from the description of the algorithm's loop.

It is also possible to efficiently update the optimal auto-regression coefficients vector a as new points in the time-series become available. The update is to be done continuously, as soon as each new point of the time-series becomes known. The algorithm minimizes an exponential smoothing-based variant of the squares of forecasting errors. In particular, the algorithm minimizes the following error function

$$\min E_m(t) = \sum_{j=M}^t \lambda^{t-j} (e_m^f(j))^2$$

where

$$\begin{aligned} e_m^f(j) &= d_j + \tilde{a}_m(j)^T \tilde{d}_m(j-1) \\ \tilde{d}_m(j) &= [d_j d_{j-1}, \dots, d_{j-m+1}]^T \\ \tilde{a}_m(j) &= [a_1(j), \dots, a_m(j)]^T \end{aligned}$$

and λ in $(0,1]$ plays a role analogous to the α factor in SES. Following the same steps in the analysis of the equations resulting from the First Order Necessary Conditions for the optimization of the new objective function, we obtain the following linear system of "normal equations":

$$\begin{aligned} R_m(t-1) \tilde{a}_m(t) &= -\tilde{r}_m(t) \\ \tilde{r}_m(t) &= \sum_{j=M}^t \lambda^{t-j} d_j \tilde{d}_m(j-1) \\ R_m(t) &= \sum_{j=M}^t \lambda^{t-j} \tilde{d}_m(j-1) \tilde{d}_m(j-1)^T \end{aligned}$$

The optimum total error will then be $E_m^f(t) = r_{om}^f(t) + \tilde{a}_m(t)^T \tilde{r}_m(t)$ with the first term $r_{om}^f(t) = \sum_{j=M}^t \lambda^{t-j} d_j^2$. By decomposing the $R_m(t)$ matrix and the vectors $\tilde{d}_{m+1}(t) = [\tilde{d}_m(t)^T \quad d_{t-m}]^T = [d_t \quad \tilde{d}_m(t-1)^T]^T$ and $\tilde{r}_m(t), \tilde{a}_m(t)$ we finally obtain the following discrete time-recursions:

$$\begin{aligned} \tilde{a}_m(t+1) &= \tilde{a}_m(t) + e_m^f(t+1) \tilde{w}_m^*(t) = \tilde{a}_m(t) + e_m^f(t+1) \tilde{w}_m(t) \\ e_m^f(t+1) &= d_{t+1} + \tilde{a}_m(t)^T \tilde{d}_m(t) \\ e_m^f(t+1) &= d_{t+1} + \tilde{a}_m(t+1)^T d_m(t) \\ R_m(t) \tilde{w}_m^*(t) &= -\tilde{d}_m(t) \\ \lambda R_m(t) \tilde{w}_m(t+1) &= -\tilde{d}_m(t+1) \end{aligned}$$

The quantities $e_m^f(t)$ are known as the a priori errors whereas the quantities $e_m^f(t)$ are known as the a posteriori errors of the prediction process. The vectors $\tilde{w}_m^*(t)$ are known as the *Kalman gain vectors*. Now, the Sherman-Morrison equality from

linear algebra states that if x is a vector of n components, the matrix xx^T is of rank 1 and the matrix R is invertible, then for every non-zero λ it holds that

$$(\lambda R + xx^T)^{-1} = \frac{1}{\lambda} R^{-1} - \frac{\lambda^{-2} R^{-1} xx^T R^{-1}}{1 + \lambda^{-1} x^T R^{-1} x}$$

Using this fact, we obtain a formula to compute recursively the inverse of the matrix $R_m(t)$ as follows:

$$P_m(t+1) = R_m(t+1)^{-1} = \lambda^{-1} R_m(t)^{-1} - \tilde{w}_m^*(t+1) \tilde{w}_m(t+1)^T$$

From this, the classical time-recursive optimal linear prediction algorithm follows.

Algorithm Time-Recursive Optimal Linear Predictor

Input: A time-series $d = \{d_0, \dots, d_t\}$, the order p of the optimal least-squares linear predictor, the smoothing factor λ , the quantities $P_p(t)$, $\tilde{w}_p^*(t)$, $\tilde{a}_p(t)$ from the previous iteration, and the latest time-series point d_{t+1} .

Output: Updated Optimal Coefficients vector $\tilde{a}_p(t+1)$ of dimension $p \times 1$ to be used in forward time-series prediction $F_{t+2} = -(a_1(t+1)d_{t+1} + \dots + a_p(t+1)d_{t+2-p})$

Begin

0. If $t=0$ then Set $P_p(0) = \sigma^{-2}I$, $0 < \sigma < 1$ /* initialization */

1. Set $\tilde{w}_p(t+1) = \lambda^{-1} P_p(t) \tilde{d}_p(t+1)$

2. Set $\alpha_p(t+1) = 1 - \tilde{d}_p(t+1)^T \tilde{w}_p(t+1)$

3. Set $\tilde{w}_p^*(t+1) = \frac{1}{\alpha_p(t+1)} \tilde{w}_p(t+1)$

4. Set $P_p(t+1) = \lambda^{-1} P_p(t) - \tilde{w}_p^*(t+1) \tilde{w}_p(t+1)^T$

5. Set $e_p^f(t+1) = d_{t+1} + \tilde{a}_p(t)^T \tilde{d}_p(t)$

6. Set $\tilde{a}_p(t+1) = \tilde{a}_p(t) + e_p^f(t+1) \tilde{w}_p^*(t)$

7. return $\tilde{a}_p(t+1)$.

End

This algorithm runs in $O(p^2)$ complexity since Step 4 is of this complexity. A faster time-recursive algorithm that runs in $O(p)$ iterations was first developed in 1978, and is known as the *Fast Kalman Algorithm*. Details of this algorithmic scheme can be found in (Karagiannis 1988).

Auto-regressive models are special cases of the more general case where a time-series d_n is well approximated by a model of the form

$$d_n = - \sum_{k=1}^p a_k d_{n-k} + \sum_{k=0}^q b_k u_{n-k}. \quad (2.3)$$

where the input driving sequence u_n is white noise process that is inherent in the model (and cannot be attributed to some “observation error”). This model is known

as an *Auto-regressive Moving Average model* (ARMA (p, q) process) with parameters p and q . Clearly, the AR model of order p is a special case of the ARMA $(p, 0)$ model with $q = 0$ and $b_0 = 1$. As with AR models, an ARMA model can be applied only to wide-sense stationary processes generating the sequence d_n so if the time-series is not wide-sense stationary, the differencing operators should be applied to the time-series until the resulting time-series appears to be stationary in the wide sense. If application of the differencing operator is necessary, then the model is called an ARIMA (p, d, q) model, where the parameter d refers to the number of times the difference operator had to be applied to turn the time-series into a wide-sense stationary time-series. In time-series where noise plays a significant role in the signal values, application of an ARMA model for prediction may yield much superior results than AR-based models, but the downside of the application of an ARMA model is the fact that it requires the optimization of a highly nonlinear objective function; this in turn implies that standard gradient-descent-based methods of nonlinear optimization can only guarantee convergence of the model parameters a_k and b_k to a saddle point (or at most a local minimum, see [Chap. 1](#)).

Another special case worth noting concerns the development of a forecasting algorithm assuming an MA (Moving Average) process of order q . Assuming that the time-series is generated from a process of the form

$$d_n = \sum_{k=0}^q b_k u_{n-k}$$

where u_n is white noise, it is also possible to model the time-series as an infinite-order Auto-Regressive-based time-series according to the model $AR(\infty)$:

$$d_n = \sum_{k=1}^{\infty} a_k d_{n-k} + u_n$$

(assuming the time series exists or can be extended to negative infinite time). Durbin (1960) showed that by considering a sufficiently large order, an $AR(p)$ model optimized via the Levinson-Durbin algorithm can approximate very well a time-series arising from a $MA(q)$ process. Having obtained the vector a of length p that must satisfy $q \ll p \ll N$, the length of the time-series, the optimal estimation for the parameters $b_1, \dots, b_q$ are then obtained as the solution to the linear system

$$\begin{aligned} Rb &= -\tilde{r} \\ R_{i,j} &= \frac{1}{p+1} \sum_{n=0}^{p-|i-j|} a_n a_{n+|i-j|}, \quad i, j = 1, \dots, q \\ \tilde{r}_i &= \frac{1}{p+1} \sum_{n=0}^{p-i} a_n a_{n+i}, \quad i = 1, \dots, q \end{aligned}$$

It must be noted that the b vector thus obtained contains the optimal parameters in a Maximum Likelihood Estimation sense.

2.5 Artificial Intelligence-Based Forecasting

Among the many tools that Artificial Intelligence researchers developed during the past half-century, Artificial Neural Networks, ANNs for short, by far have been the most popular tools for forecasting financial and other time-series. Many different factors contributed to the ANNs' popularity; the ability to predict continuous-valued outputs (or classes) for a given input vector ranks certainly high among those factors. Because of the enormous popularity of ANNs as forecasting tools in stock-market data and other business and financial data, a short description of the ANN architecture as well as the most popular algorithm for training ANNs is given below.

The discussion on ANNs will follow a fairly standard approach where an ANN will be considered as a system that can be trained to fit a finite data set $X = \{x^{[1]}, \dots, x^{[N]}\} \subset \mathbb{R}^n$ to an accompanying value set $V = \{v_1, \dots, v_N\}$ of real numbers by constructing a function $g(x)$ such that the sum of squared errors $\sum_{i=1}^N (v_i - g(x^{[i]}))^2$ is minimized.

We shall restrict attention to *feed-forward, multi-layer perceptron models* (MLP). A MLP ANN is a network of individual nodes, called perceptrons organized in a series of layers as shown in Fig. 2.21. Each node (perceptron) in this network has some inputs and outputs as shown in more detail in Fig. 2.22, and represents a processing element that implements a so-called activation function. This activation function is a function of one real variable that is the weighted sum of the values of the node's inputs, so that

$$v = \varphi\left(\sum_{i=0}^k w_k u_k\right).$$

The most often used function for the hidden and output layers nodes is the sigmoid function:

$$\varphi(x) = (1 + e^{-x})^{-1}$$

which has the property that is differentiable everywhere, and its derivative is $\varphi'(x) = \varphi(x)(1 - \varphi(x))$, plus near the origin the function is almost linear. For the input layers, the activation function most often used is the identity function, $\varphi(x) = x$. Finally, a less often used activation function is the threshold function:

$$\varphi(x) = \text{sgn}(x_+), \quad \text{sgn}(0) = 0$$

where the sign function $\text{sgn}(x)$ is defined in the beginning of this chapter. The interest in MLP networks stems from a fundamental theorem due to Kolmogorov stating that a MLP with just one hidden layer and threshold nodes can approximate any function with any specified precision, and from an algorithmic technique developed in the 1980s that became known as the "BackPropagation Algorithm" that could train an MLP network to classify patterns given a input training set.

The (Error) BackPropagation algorithm (BP) is a form of gradient descent-based optimization algorithm that attempts to adjust the weights of each edge

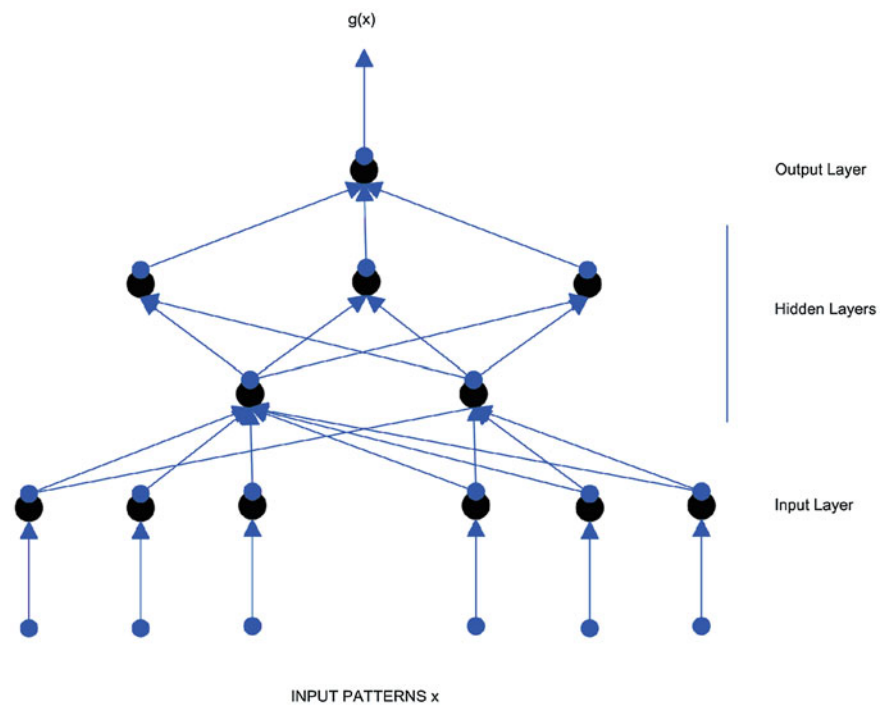


Fig. 2.21 A feed-forward multi-layer perceptron artificial neural network. notice how each node accepts inputs only from the immediate layer of nodes beneath it, and transmits the same value to the nodes in the layer immediately above it. Edges connecting two nodes have associated weights to them that multiply the value outputted by the node at the lower end of the edge

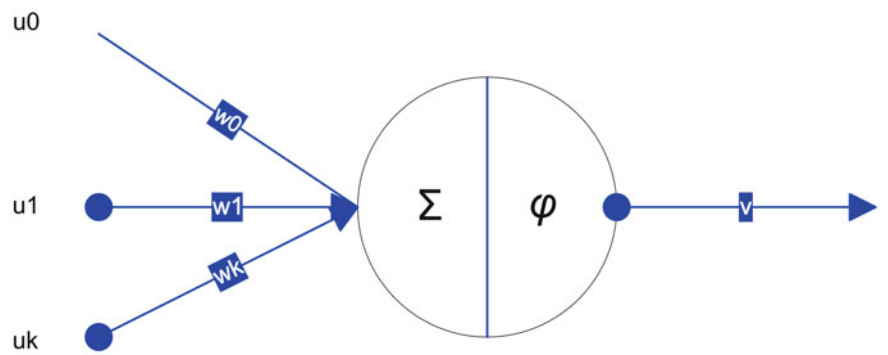


Fig. 2.22 Schematic representation of an individual perceptron. The node has associated weights $w_0, \dots, w_k$ that multiply each of its input values $u_0, \dots, u_k$ and an activation function $\phi()$. Its output is the value $v = \phi(w_0 u_0 + w_1 u_1 + \dots, w_k u_k)$. By default, w_0 represents the bias of the node and the corresponding variable value u_0 is constantly set to -1

connecting two nodes in the MLP to minimize the square error of the ANN's predicted outputs for each vector input pattern x it receives. The objective function to be *minimized* by the ANN—whose architecture, namely number of hidden layers and number of nodes in each layer is assumed to have been fixed somehow—is therefore the following:

$$E(W) = \frac{1}{2} \sum_{j=1}^N \left(g(x^{[j]}, W) - v_j \right)^2$$

The variables in this function are the weights of each node that form the matrix W . Given the above discussion, it should be now clear that given the topology of the network, the activation function for each node, and the weights W that essentially define the network, the value $g(x, W)$ is trivial to compute for any pattern x fed to the ANN.

BP employs a standard gradient descent-based method (see [Sect. 1.1.1](#)) in order to obtain a *saddle point* of the objective function $E(W)$ (i.e. a point where the derivative is zero). The rule that BP employs to update the i th node's weight w_{ij} is therefore the following

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \quad (2.4)$$

where η is a user-defined parameter. The partial derivative $\partial E / \partial w_{ij}$ can be computed using the chain rule of differential calculus. Let the node's input sum be denoted by ξ , having accepted input values $u_{i0}, \dots, u_{ik}$ with weights for each input $w_{i0}, \dots, w_{ik}$. The partial derivative of $E(W)$ with respect to w_{ij} by the chain rule is

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \xi} \frac{\partial \xi}{\partial w_{ij}} = \frac{\partial E}{\partial \xi} u_{ij}$$

Now, the partial derivative $\partial E / \partial \xi = \delta$ is called *the error*. Given an input pattern x , and its associated value v , we can calculate the output value of the ANN with given weights for each edge by forward propagating the values of the nodes at the lowest layer to the nodes in higher layers, until the output node's value is determined. Once the output node's value is computed, the derivative of $E(W)$ with respect to the *output* is simply $g_o(x, W) - v$. If the net sum of the inputs of this output node is denoted by ξ^o and assuming that the node is a sigmoid activation node so that $g(x, W) = \phi(\xi^o)$, we obtain

$$\begin{aligned} \delta^o &= \partial E / \partial \xi^o = \partial E / \partial g(x, W) \times \partial g(x, W) / \partial \xi^o \\ &= [g(x, W) - v] \phi'(\xi^o) = [g(x, W) - v] g(x, W) [1 - g(x, W)] \end{aligned}$$

This quantity is the error at the output layer and can be used to update the weights at the *most upper hidden layer* using the BP rule:

$$w_{ko} \leftarrow w_{ko} - \eta \delta^o v_k$$

where w_{ko} are the weights of the edges connecting the k th most upper hidden layer node and the output node, and v_k is the output of that k th node. Now, to update the weights of the edges between the previous hidden layer and the upper-most hidden layer, the chain rule can be applied again. As before, let ξ^k denote the net input sum of the k th upper-most hidden layer node, so that $v_k = \varphi(\xi^k)$, having assumed sigmoid activation hidden nodes (except at the input layer nodes where the activation function is assumed to be the identity function). Now, $\delta^k = \partial E / \partial \xi^k = \partial E / \partial v_k \times \partial v_k / \partial \xi^k = \partial E / \partial v_k \varphi'(\xi^k)$. The partial derivative of E with respect to v_k can be easily computed via the chain rule again as $\partial E / \partial v_k = \partial E / \partial \xi^o \times \partial \xi^o / \partial v_k = \delta^o w_{ko}$. Substituting this expression in the previous equation for the error δ^k we obtain the following Back-Propagation of errors equation:

$$\delta^k = \delta^o w_{ko} \varphi'(\xi^k) = \delta^o w_{ko} v_k [1 - v_k] \quad (2.5)$$

Applying the BP rule, we see that the weights of the edges connecting the upper-most hidden layer node i and node k on the layer immediately below it are updated according to

$$w_{ki} \leftarrow w_{ki} - \eta \delta^i v_k$$

The detailed BP algorithm for MLP ANNs can now be stated.

Algorithm Online BackPropagation MLP Training

Input: A dataset $S = \{x^{[1]}, \dots, x^{[N]}\}$ of vectors in $\mathbb{R}^n$, together with associated real values $r_1, \dots, r_N$, number of hidden layers L , number of nodes at each layer M , learning rate η , maximal number of epochs T and maximum acceptable error tolerance ε (optionally an activation function for the hidden and output layers other than the sigmoid).

Output: matrix of ANN adjusted weights.

Begin

0. Set the weights of each edge in the network to a small random value, Set $E = +\infty, t = 1, j = 1$ /* initialization */
1. while ($E > \varepsilon$ AND $t \leq T$) do
 - a. Pass $x^{[j]}$ as input, and compute the output v of every node in the ANN. /* Forward Propagation */
 - b. Set $\delta^o = [g(x^{[j]}, W) - r_j]g(x^{[j]}, W)[1 - g(x^{[j]}, W)]$ /* output layer error */
 - c. Set $\delta^k = \delta^o w_{ko} v_k [1 - v_k]$ for each node k at the upper-most hidden layer. /* upper-most hidden layer error */
 - d. for each layer l below the upper-most hidden layer do /* BackPropagation */
 - i. for each edge (m, p) connecting layer l with upper layer $l+1$ do
 1. Set $\delta^m = \delta^p w_{mp} v_m [1 - v_m]$.
 - ii. end for

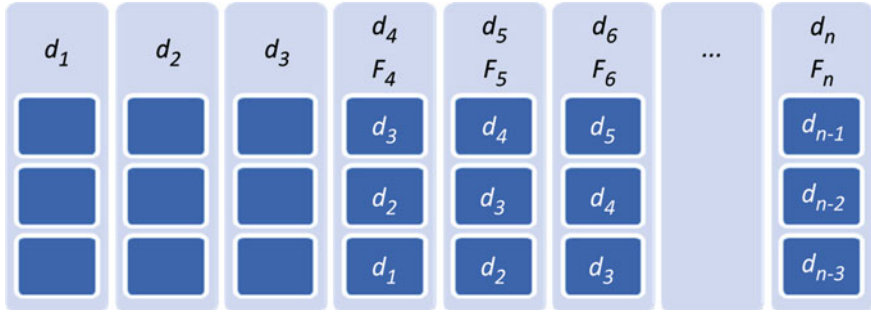


Fig. 2.23 Example break-down of time-series for use with ANN

- e. end-for.
 - f. for each edge (m,o) connecting upper-most hidden layer node m to output node o do
 - i. Set $w_{mo} = w_{mo} - \eta \delta^o v_m$.
 - g. end-for.
 - h. for each edge (k,i) connecting node k at hidden layer l to node i at hidden layer $l+1$ do
 - i. Set $w_{ki} \leftarrow w_{ki} - \eta \delta^i v_k$.
 - i. end-for
 - j. Set $E = \frac{1}{2} \sum_{j=1}^N (g(x^{[l]}, W) - v_j)^2$ using current weights W .
 - k. if $j = N$ then Set $t = t + 1$, Set $j = 0$ else Set $j = j + 1$. /* epoch update */
 3. end-while
 4. return W .
- End.

The Algorithm is known as the *online* version of BackPropagation because weights are updated as soon as each pattern $x^{[j]}$ is fed to the algorithm. In the *batch* version of BackPropagation, weights are not updated until a full epoch has passed, meaning all instances have been fed to the ANN. In such a case, as patterns are fed to the system, the system maintains and stores the changes that should be made to each weight, but does not actually modify the edge weights and applies the cumulative changes as soon as all patterns have been fed to the system.

To apply MLP ANNs in forecasting, a trivial procedure may have to be applied first. Assume the time-series d_i $i = 1, \dots, N$ is available. Then, one can obtain the data-set of patterns S by first choosing a time-window w that will serve as the dimensionality of the input pattern data-set, and define the $N-w$ patterns $x_i = [d_i \dots d_{i+w-1}]^T$ in R^w with associated values $v_i = d_{i+w}$. Figure 2.23 illustrates this procedure for $w = 3$.

Several points should be made regarding MLP ANNs. The most important point has to do with the generalization capacity of an ANN. It is true that the more hidden nodes one selects as input network architecture the more likely it is for the system to be able to minimize the training error function $E(W)$. The same holds true for the maximum number of epochs T the system is allowed to run for. However, minimization of the training error E does not necessarily guarantee good performance. Indeed, ANNs are often plagued by a phenomenon known as *overfitting*, whereby the ANN simply “memorizes” the training input patterns and so even though its performance on the training set is very good, its performance on previously unseen instance patterns is unacceptably bad. For this reason, ANNs are usually trained using a (reasonably large) subset S' of the training set S , and at the end of each epoch, the performance of the ANN is evaluated on $S - S'$ and when this (test-set) performance stops improving, the algorithm stops.

A last point to be made about ANNs is their inherent instability. By instability we mean that small differences in the inputs to the algorithm can sometimes result in significantly differently trained ANNs. This observation does not have only bad consequences. A good consequence is that combining different ANNs (that are obtained starting from different random initializations of the network's weights and applying the same BP algorithm) in a classifier ensemble can lead to better performance due to the “diversity” of the base classifiers involved in the scheme, caused precisely by the ANN's inherent instability.

2.5.1 Case Study

We illustrate the use of ANN-based forecasting in the case of stock-market forecasting. We try to forecast the daily closing price of a common stock that trades in the Athens Stock Exchange. The time-series contains data for approximately 15 months (290 data points) between January 2009 and March 2010. These data points were broken down in patterns of 10 points, where we hypothesized that the closing prices of two weeks should provide a reasonable indicator of the price of the stock for the next day. The ANN architecture was set to have a single hidden layer with 20 nodes, and one output node. Using the MATLAB Neural Network Toolbox, we trained the system using cross-validation as the criterion to be optimized. The historical performance of the training of the system is shown in Fig. 2.24.

The results of the training are very good as the trained ANN can forecast the next day closing price of the stock with reasonable accuracy, certainly much better than the accuracy obtainable from exponential smoothing methods. Figure 2.25 shows how closely the ANN can match the historical data.

The accuracy of the forecast provided is more evident in the following regression plot (Fig. 2.26)—output by the matlab toolbox—that plots the forecast values versus the target values for the time-series in our case-study.

The performance of the trained ANN is also shown in Fig. 2.27, where the *Percentage Deviations* of the forecasts from the actual values are shown. Notice

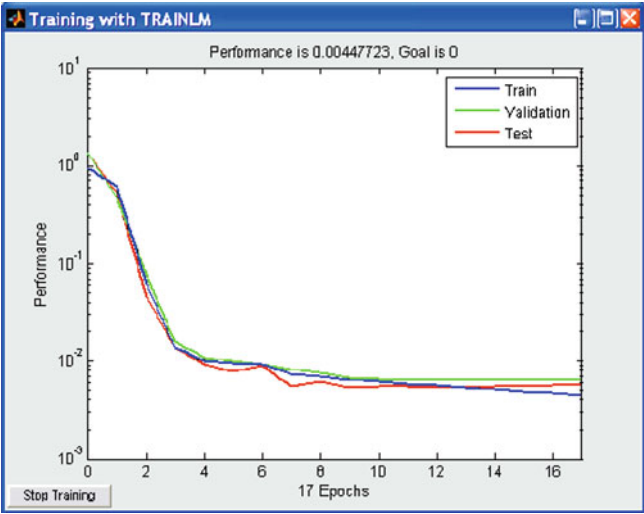


Fig. 2.24 ANN training using matlab NN-toolbox

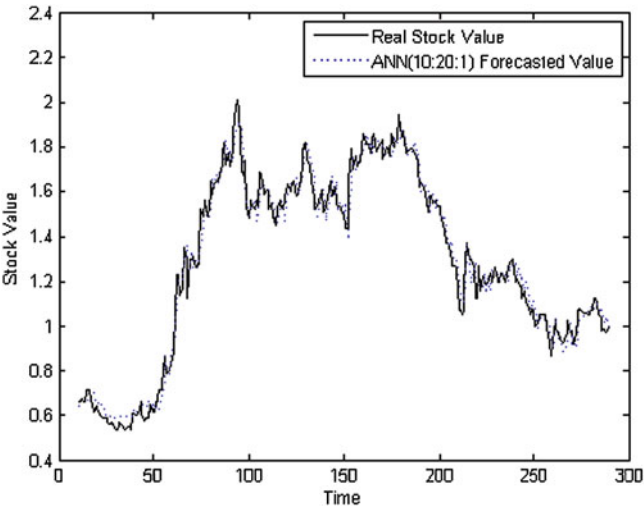


Fig. 2.25 Forecasting performance of the trained ANN on closing price of common stock

that the daily percentage fluctuations of a common stock are usually in a comparable range to the fluctuations observed in the figure.

The Mean Square Error of the Neural Network’s forecasts is $MSE_{289} = 0.0032$. This value is actually better (but *not* significantly better) than the Mean Square Error of the forecasts obtained by the Naïve Forecast Method forecasting $F_{t+1} = d_t$, which obtains an MSE_{289} value of 0.0038. In Fig. 2.28 we show a plot of the Percentage Forecast Deviations obtained by applying the Naïve Forecast method.

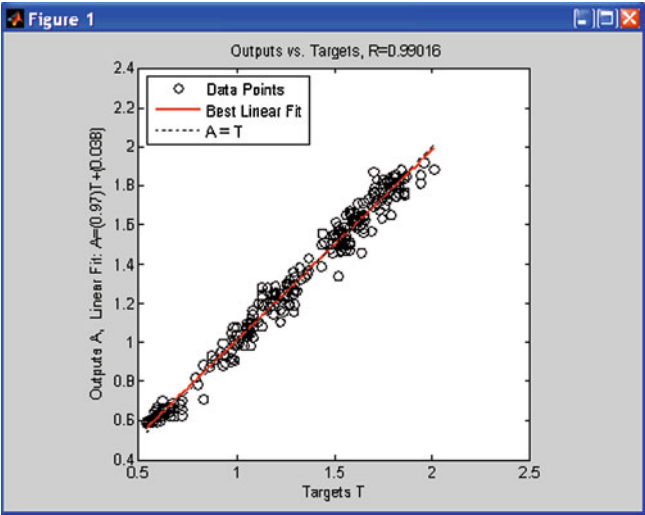


Fig. 2.26 Forecasting performance of the trained ANN as regression line between actual and forecasted value pairs

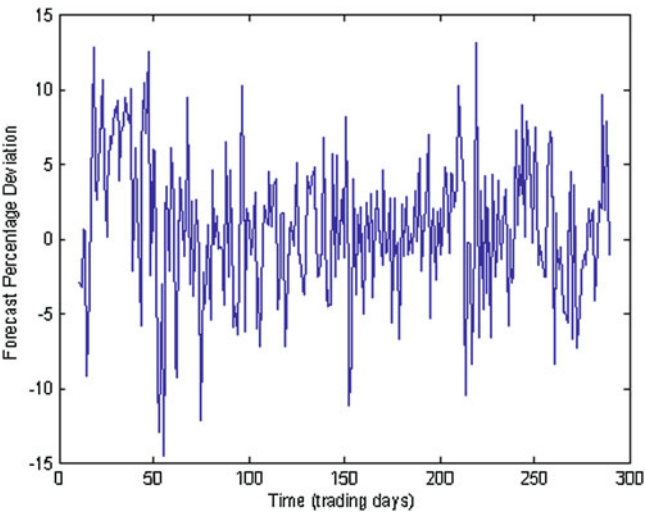


Fig. 2.27 Forecast percentage deviation of the trained ANN on the closing value of a common stock time-series

The predictions made by the trained ANN are reasonable as one can easily check by looking at Figs. 2.25 and 2.26. However, this does *not* mean that one can make money by speculating on the stock’s price as predicted by the ANN. Indeed, suppose we use the ANN’s predictions to predict whether buying that stock on any particular day is likely to be a good investment in that within the next 5 days

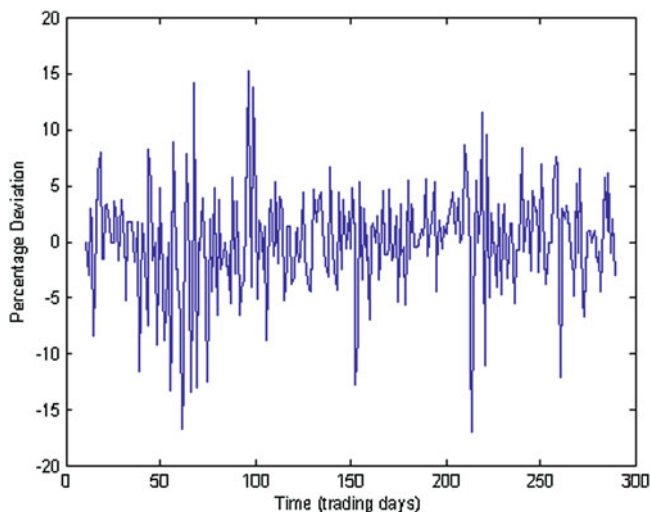


Fig. 2.28 Forecasting percentage deviation of the Naïve forecasting method on the closing value of a common stock time-series. This figure essentially represents the daily percentage fluctuations of the closing price of the stock

the stock will increase its value (speculative buying behavior). It turns out that the particular ANN will be wrong more than 50% of the time in its trend prediction. Nevertheless, this does not imply that stock short-term trend prediction is impossible. Indeed, when a Support Vector Machine (Vapnik 1995) is trained with input data a vector comprising 10 continuous observations of the twice integrated (i.e. differentiated) closing stock price time-series labeled, each vector labeled as “positive” if the twice-differentiated time-series increases its value within the next 5 observations, and “negative” otherwise, the results show a testing accuracy (on unseen data) of more than 70%, meaning that it is possible to a certain extent to predict short-term trends in stock market time-series for certain stocks at least.

2.6 Forecasting Ensembles

Ensembles of forecasting systems have been used in weather forecasting for a long time with great success. In weather forecasting, a system of highly non-linear differential equations is solved many times, each time with a slightly perturbed initial condition. Because of the nonlinearity of the system dynamics involved, even slight perturbations to the initial conditions of the system can lead to widely-differing solutions within a short amount of time. Because the initial conditions are not exactly known, the system is solved many times to see in a statistical sense how it will likely evolve. The results of the differently-initialized system are then combined to produce a final prediction about the weather. The predictions of an

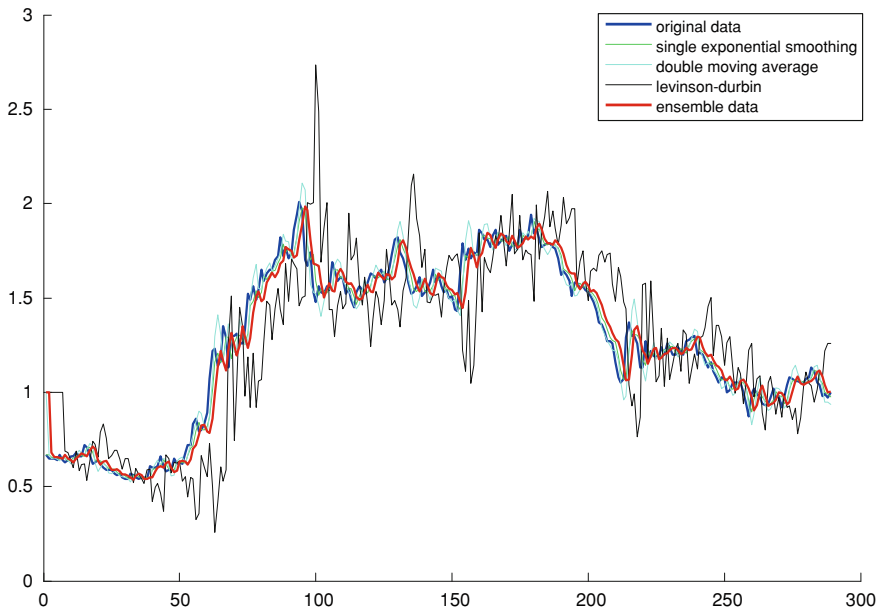


Fig. 2.29 Forecasting performance of an ensemble of three different methods on closing price of common stock. The x -axis represents time (in trading days)

event are given a probability that is often equal to the frequency of appearance of the event in the solutions of the system in the ensemble.

Similarly to forecasting ensembles for weather forecasting, general time-series forecasting ensembles (sometimes also known as committee forecasting) comprise of a collection of forecasting models and algorithms that operate on the same time-series to produce a forecast for the next period. The predictions are then combined in a *fusion scheme* to produce a final forecast. The most obvious fusion method is to average the predictions of the methods to obtain the final forecast. A more intelligent but still straightforward method to combine the ensembles' forecasts would be to compute a weighted average of the forecasters' predictions, where the weight of each forecaster would be proportional to its accuracy as measured by one of the metrics discussed at the beginning of this chapter. More formally, let $\mathfrak{F} = \{P_1, \dots, P_L\}$ denote a set of L forecasting algorithms. Assume that the forecasts F_{n+1}^i , $i = 1, \dots, L$ for the $(n + 1)$ st element of the time-series d_n have associated MAPD_i values $w_1, \dots, w_L$. The combined forecast according to the *weighted-average fusion scheme* is then given as

$$F_{n+1} = \frac{\sum_{i=1}^L w_i^{-1} F_{n+1}^i}{\sum_{i=1}^L w_i^{-1}}$$

For example, combining the predictions of the Single Exponential Smoothing model, the Double Moving Average model, and the Auto-Regressive model in one

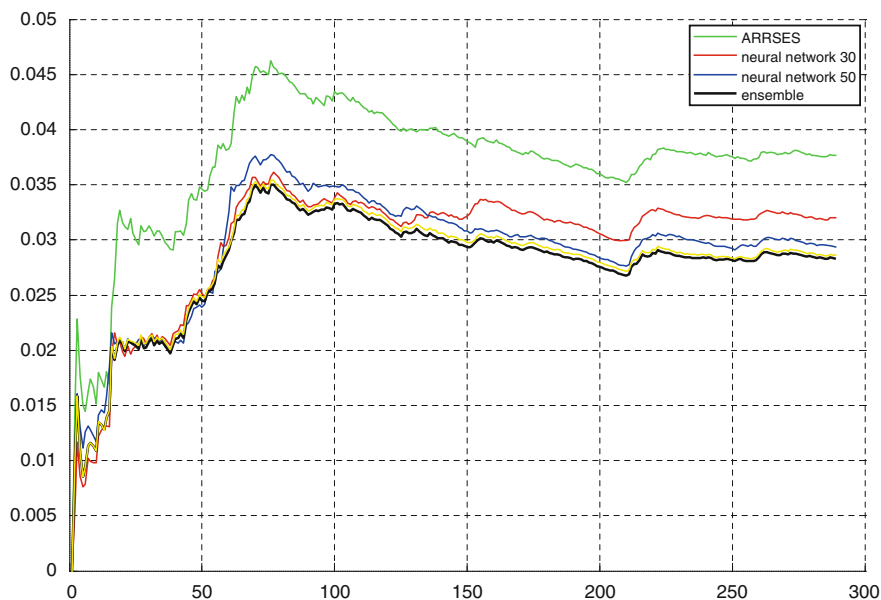


Fig. 2.30 Forecasting performance of an ensemble of three different methods on closing price of common stock. The x -axis represents time (in trading days). The y -axis represents the MAPD metric for each forecasting method

ensemble and applying it to the same (undifferentiated) time-series as that of Fig. 2.25, we get the forecasts shown in Fig. 2.29.

If we combine the predictions of ARRSES, an ANN with 30 hidden nodes, and an ANN with 50 hidden nodes forecasting the same common stock time-series as before, we get the MAPD_k measure plotted in Fig. 2.30. As can be seen, combining the forecasts of the three methods actually leads to a consistently better forecast in terms of MAPD error metric. Such results are *not* always typical of forecasting ensembles however.

Testing the ensemble idea on a much more “predictable” data-set, that of CO_2 concentrations measured at the Mauna-Loa Observatory provided by the NIST/SEMATECH e-Handbook of Statistical Methods (<http://www.itl.nist.gov/div898/handbook>), we get the graphs shown in Fig. 2.31. The ensemble consists of a SES forecaster, a DES forecaster, and an AR-based forecaster. Notice that *none* of the forecasting methods comprising the ensemble of Fig. 2.31 have any notion of seasonality (it is obvious by looking at the graph that seasonality and a trend component is inherent in the data).

Experiments with forecasting ensembles of ANNs on stock-market data have often produced poor results (the ensemble prediction being often worse than the best individual forecasting method in the ensemble.) However, when we test the performance of an ensemble of 9 ANNs, each with a different number of hidden nodes, ranging from 10, 20, ..., 90 on three stock-market time-series data sets, the

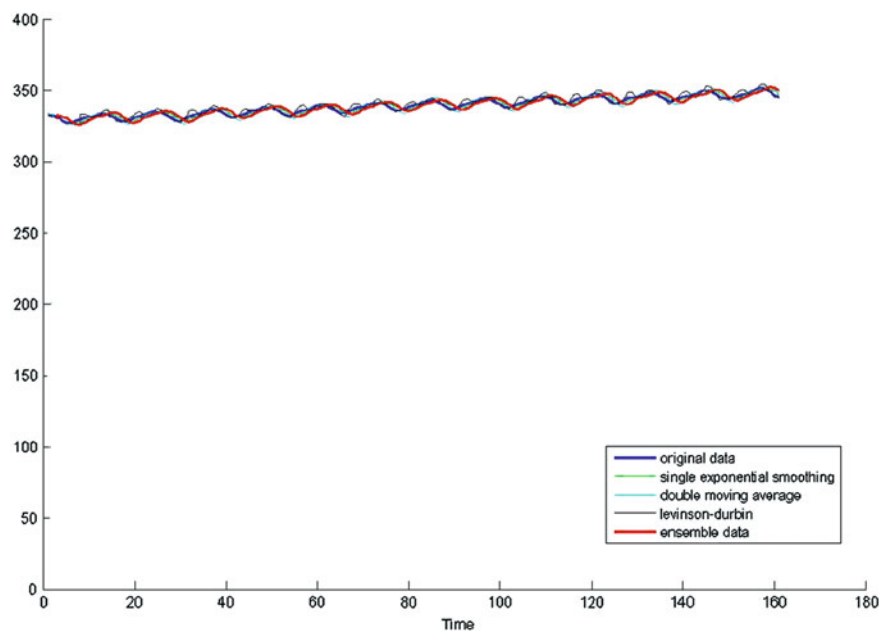


Fig. 2.31 Forecasting performance of an ensemble of three different methods on CO₂ concentrations measurements at the Mauna-Loa observatory

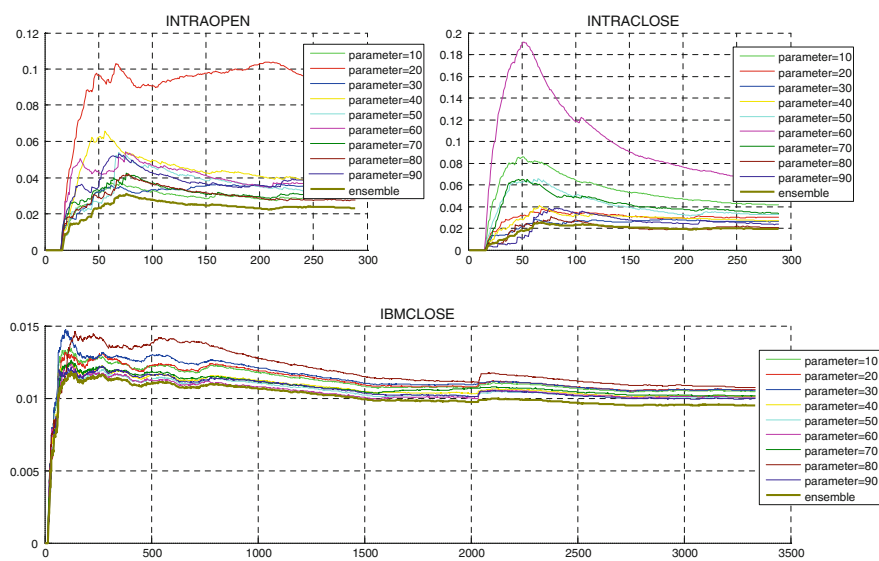


Fig. 2.32 Forecasting performance of an ensemble of 9 ANNs on three different stock market data sets. The x-axis represents trading days. The y-axis represents MAPD error. The ensemble outperforms each individual base ANN

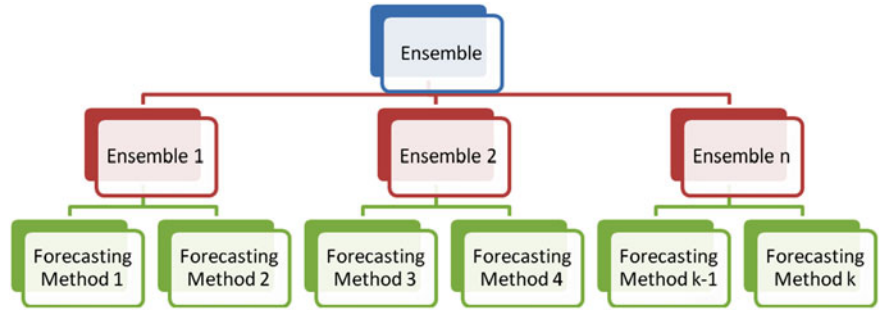


Fig. 2.33 Structure of a tree forecasting ensemble. obviously, $n = k/2$ in the figure

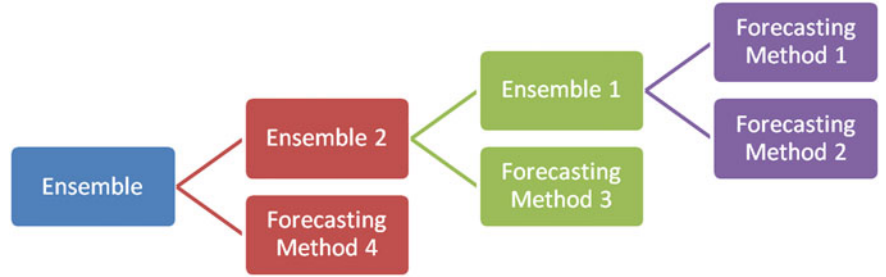


Fig. 2.34 Structure of a cascading forecasting ensemble

results shown in Fig. 2.32 show that the ensemble of ANNs combined with the simple rule discussed above outperforms every individual ANN in the ensemble.

Unfortunately, while weather forecasting ensembles have greatly improved the performance of individual models, general time-series forecasting ensembles based on ANN trained for forecasting, produce mixed results. Some times the ensemble result is clearly inferior to the predictive power of good, single models. The conditions under which such phenomena occur are still the subject of active research. Another area of research is concerned with the effect that the diversity of the models in the selection of the base forecasting algorithms has on forecasting accuracy, which is a favorite subject of research in the classifier ensemble literature within the machine learning community.

Besides the obvious weighted average method for combining individual forecasters into an ensemble forecast, other architectures are also possible. For example, in a tree architecture, forecasting methods form pairs of forecasters that are fused in one ensemble, and the result is fed to a final forecast ensemble, where all the ensemble forecasts compute their values using the weighted average method described above. This architecture is shown in Fig. 2.33.

An alternative architecture known as cascading ensemble, also “borrowed” from the classifier research community is shown in Fig. 2.34. Preliminary experiments in (Luo 2010) show that a tree ensemble classifier may perform slightly better than a standard ensemble or a cascading ensemble.

2.7 Prediction Markets

When historical data about the demand for a product exist, one can use any or all of the methods discussed in the previous sections regarding time-series analysis to predict the future demand within a short *lead-time*. However, when new products are developed, the more innovative the products are, the less clues one has as to what the demand for such products might be. In the absence of any quantitative data to work with, marketing research tools including *market surveys* could be launched to “probe” the market so as to gauge the “acceptance” of the product in the market before it is launched. Such market surveys are usually conducted using questionnaires that are filled in by participants in the survey, and are then processed using segmentation and clustering tools from classical pattern recognition that can eventually lead to some kind of forecast demand. Unfortunately, the sample population that must fill out such questionnaires so that the survey results are statistically meaningful is usually of very big size, making such tools and methods very expensive. An alternative approach, based on ideas borrowed from the mechanisms of stock-markets is known as Prediction Markets (PMs) and has been proved to be successful enough that it is currently in use by most major corporations world-wide, at least when other quantifiable data are not available for the prediction problem at hand. PMs can be thought of as a (virtual) stock market for events: player/traders place “bids” to buy/sell options on a future event: each option will pay a number of (virtual) dollars if a particular future event materializes, and pay nothing otherwise. Usually, derivative options are also allowed.

Wolfers and Zitzewitz (2007) restricted their attention to simple, binary option PMs, where traders buy and sell an all-or-nothing contract that will pay \$1 if a specific event occurs, and nothing otherwise (the \$1 may be real or “virtual” money used only inside the particular PMs). The specific event could be the re-election of the US President to office for a second term, or whether the latest smart-phone from Apple will exceed the sales of its predecessor within 3 months. The market players (traders) have in general different subjective beliefs about the outcome of the specific event, and the belief of trader j on the occurrence of the event is considered a random variable, denoted by q_j , drawn from a distribution $F(x)$. Assuming further that traders are price-takers—so there are no oligopoly dynamics—who maximize their expected utility defined to be a log function that is often assumed in economics theory, the optimization problem each trader j faces is the following:

$$\max U(x) = q_j \log(y + x(1 - \pi)) + (1 - q_j) \log(y - x\pi)$$

where π is the price for the contract, y is the trader’s wealth level, and x is the decision variable representing the number of contracts trader j should buy to maximize their expected subjective utility. Setting $dU(x)/dx = 0$, we obtain the optimal buying quantity for trader j at price π to be

$$x_j = y(q_j - \pi) / [\pi(1 - \pi)]$$

From this, one immediately sees that individual demand will be zero when the price π equals the belief of the trader q_j —as expected. Also, trader's j demand will increase linearly with their belief, and is decreasing in risk (represented by a price close to 0.5).

Now, the PMs will be in equilibrium when supply will equal demand, which can be written down mathematically as

$$\int_{-\infty}^{\pi} \int y \frac{q - \pi}{\pi(1 - \pi)} dG(y) dF(q) = \int_{\pi}^{+\infty} \int y \frac{\pi - q}{\pi(1 - \pi)} dG(y) dF(q)$$

where $G(y)$ is the Cumulative Distribution Function (cdf) of the wealth levels of all traders in the market. Denoting the Probability Density Function (pdf) of the traders' beliefs by $f(q)$, and assuming that wealth and beliefs are uncorrelated so that $E[q, y] = 0$, the above equation implies that

$$\begin{aligned} \frac{y}{\pi(1 - \pi)} \int_{-\infty}^{\pi} (q - \pi) f(q) dq &= \frac{y}{\pi(1 - \pi)} \int_{\pi}^{+\infty} (\pi - q) f(q) dq \Leftrightarrow \\ \pi &= \int_{-\infty}^{+\infty} q f(q) dq = E[q] \end{aligned}$$

The latter directly shows that under this simple but elegant and easily expandable model, *market equilibrium is achieved when the market price equals the mean belief of the population about the probability of the outcome of the event*. As Wolfers and Zitzewitz argue, the monotonicity of demand in expected returns implies that this is the only price which results in equilibrium of the market (resulting in zero aggregate demand).

Interestingly, even when the model is generalized to account for various degrees of correlation of traders' wealth levels and beliefs (which should be expected to exist), or other utility functions, it remains true that the deviation of the equilibrium market price from the mean beliefs of the traders is very small—and usually negligible. Therefore, PMs should be expected to work very efficiently as (approximate) information aggregators. Indeed, this has been verified in several experiments as well.

2.8 Bibliography

Forecasting, as mentioned in the beginning of this chapter is as old as civilization itself. However, rigorous methods for time-series analysis are not that old. While statistical methods such as regression go back to the work of mathematicians including Euler, Gauss and Sir Francis Galton, work on Auto-Regressive models

goes only back to Yule (1927) and Walker (1931), while traces of the basic ideas can be found in Schuster (1906). The definitive text on Auto-Regressive models and their expanded form, ARIMA models remains Box and Jenkins (1976); the current 4th edition of this book by Box et al. (2008) contains very useful material on outlier points detection in time-series, quality control etc. Fast algorithms for solving the Yule-Walker equations are usually the subject of signal processing courses in Electrical Engineering curricula. The Levinson-Durbin algorithm was developed in the 1960s, see Durbin (1960), and at around the same time, Kalman filters were introduced in the signal processing literature as well, e.g. Kalman (1960). Exponential smoothing and its variants (SES, DES, TES, Holt–Winters method etc.) were invented in the 1950s within the context of Operations Research and Statistics research. See for example Holt (1957), Winters (1960), and the book by Brown (1962). The combination of forecasting methods has been a subject of research since the 1980s, e.g. Winkler and Makridakis (1983).

See the book by Makridakis et al. (1998) for a detailed discussion of time-series decomposition. For more details on the regression and numerical computing aspects of the method see Cheney and Kincaid (1994).

The literature on ANNs is as impressive as that on forecasting methods. Early analysis of single layer ANNs (perceptrons) was carried out in Minsky and Papert (1969). A classical treatment on ANNs remains the two-volume set (Rumelhart et al. 1987). Computational Intelligence techniques involving ensembles of ANNs used for forecasting time-series are discussed in some detail in Shi and Liu (1993), Yong (2000), and more recently, Palit and Popovic (2005). For a case-study of the application of Computational Intelligence techniques in electric load forecasting see Tzafestas and Tzafestas (2001).

Tree forecasting ensembles and cascading forecasting ensembles are described in detail in Yonming Luo's Master Thesis (Luo 2010).

2.9 Exercises

- 1 Show that the forecasts produced by the Single Exponential Smoothing method minimize the discounted cost criterion $S' = \sum_{j=0}^{\infty} (1-a)^{j+1} (d_{t-j} - F_t)^2$.
- 2 Show that for the two-variable function φ defined as $\varphi(a, b) = \|at + be - d\|^2$ where $t = [t_1 \ t_2 \ \dots \ t_n]^T$, $d = [d_1 \ d_2 \ \dots \ d_n]^T$, and $e = [1 \ 1 \ \dots \ 1]^T$ are given n -dimensional column vectors, the unique point (a^*, b^*) where $\nabla \varphi(a^*, b^*) = 0$ is its global minimizer.

- 3 Two methods for predicting weekly sales for a product gave the following results

Period	Method 1	Method 2
1	80	80
2	83	82
3	87	84
4	90	86
5	81	86
6	82	83
7	82	82
8	85	82

The actual demand observed was the following:

Period	Actual demand
1	83
2	86
3	85
4	89
5	85
6	84
7	84
8	83

Determine the $MAPD_i$ and MSE_i value of the two methods for $i = 1, \dots, 8$, as well as the Tracking Signal S_i of the two methods. Based on this information, determine whether it is possible to safely use one method or the other.

- 4 Implement the SES and DES formulae on a spreadsheet program, and use it to determine the optimal parameter a in the SES method giving the best $MAPD_8$ error on the time-series of the previous exercise.
- 5 Implement the Levinson-Durbin algorithm.

- (a) Test the implementation with the following values of $p = 2, 3, 5$ on the following time-series:

Period i	d_i
1	20
2	18
3	17
4	17
5	19
6	22
7	20
8	20
9	19
10	18
11	19
12	22
13	20
14	21
15	22
16	24
17	23
18	25
19	27
20	26

- (b) Test the same implementation on the differentiated time-series $\Delta d_i = d_i - d_{i-1}$
- 6 For the time-series d_i of Exercise 5, compute the best estimate for the value d_{21} using the *additive* model for time-series decomposition of Sect. 2.2.1. To estimate trend in the data, use the centered-moving-average method with parameter $k = 6$. Assume that the seasonality length $s = 6$.
- 7 Assume demand for some good is a *martingale* process where $d_{t+1} = d_t + R_t$, where the R_t are independent random variables normally distributed, with zero mean, and variance $\sigma_t = \sqrt{t}$. Which of the forecast methods discussed so far would give—when optimized in its parameters— the best results in the mean square error sense?

References

- Box GEP, Jenkins GM (1976) Time series analysis: forecasting and control. Holden Day, San Francisco CA
- Box GEP, Jenkins GM, Reinsel GC (2008) Time series analysis: forecasting and control., 4th edn. Wiley, Hoboken, NJ

- Brown RG (1962) Smoothing, forecasting and prediction of discrete time series. Prentice-Hall, Englewood Cliffs, NJ
- Cheney W, Kincaid D (1994) Numerical mathematics and computing, 3rd edn. Brooks/Cole Publishing Company, Pacific Grove, CA
- Durbin J (1960) The fitting of time-series models. *Rev Int Stat Inst* 28:233–244
- Ghiani G, Laporte G, Musmanno R (2004) Introduction to logistics systems planning and control. Wiley, Chichester, UK
- Halikias I (2003) Statistics: analytic methods for business decisions, 2nd edn. Rosili, Athens, Greece (in Greek)
- Holt CC (1957) Forecasting trends and seasonals by exponentially weighted moving averages. O.N.R. Memorandum 52, Carnegie Institute of Technology, Pittsburgh
- Kalman RE (1960) A new approach to linear filtering and prediction problems. *J Basic Eng* 82:35–45
- Karagiannis G (1988) Digital signal processing. National Technical University of Athens, Athens, Greece (in Greek)
- Luo Y (2010) Time series forecasting using forecasting ensembles. M.Sc. thesis, Information Networking Institute, Carnegie-Mellon University
- Makridakis S, Wheelwright SC, Hyndman RJ (1998) Forecasting: methods and applications, 3rd edn. Wiley, Hoboken, NJ
- Minsky M, Papert S (1969) Perceptrons. MIT Press, Cambridge, MA
- Palit AK, Popovic D (2005) Computational intelligence in time series forecasting: theory and applications. Springer, Berlin, Germany
- Rumelhart DE, McClelland JL et al (1987) Parallel distributed processing: explorations in the micro-structure of cognition, volume I: foundations. MIT Press, Cambridge, MA
- Sanders NR, Manrodt KB (1994) Forecasting practices in US corporations: survey results. *Interfaces* 24(2):92–100
- Schuster A (1906) On the periodicities of sunspots. *Philos Trans R Soc A* 206:69
- Shi S, Liu B (1993) Nonlinear combination of forecasts with neural networks. In: Proceedings of the international joint conference on neural networks, Nagoya, Japan
- Tzafestas S, Tzafestas E (2001) Computational intelligence techniques for short-term electric load forecasting. *J Intell Robot Syst* 31(1–3):7–68
- Vapnik VN (1995) The nature of statistical learning theory. Springer, New York
- Walker G (1931) On periodicity in series of related terms. *Proc R Soc A* 131:195–215
- Winkler R, Makridakis S (1983) The combination of forecasts. *J R Stat Soc Ser A* 137:131–165
- Winters PR (1960) Forecasting sales by exponentially weighted moving averages. *Manag Sci* 6(3):324–342
- Wolfers J, Zitzewitz E (2007) Interpreting prediction markets as probabilities. NBER working paper #12200, The Wharton school of business, University of Pennsylvania
- Yong Y (2000) Combining different procedures for adaptive regression. *J Multivar Anal* 74:135–161
- Yule GU (1927) On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers. *Philos Trans R Soc A* 226:267–298



<http://www.springer.com/978-0-85729-765-5>

Quantitative Methods in Supply Chain Management
Models and Algorithms

Christou, I.T.

2012, XIV, 398 p., Hardcover

ISBN: 978-0-85729-765-5