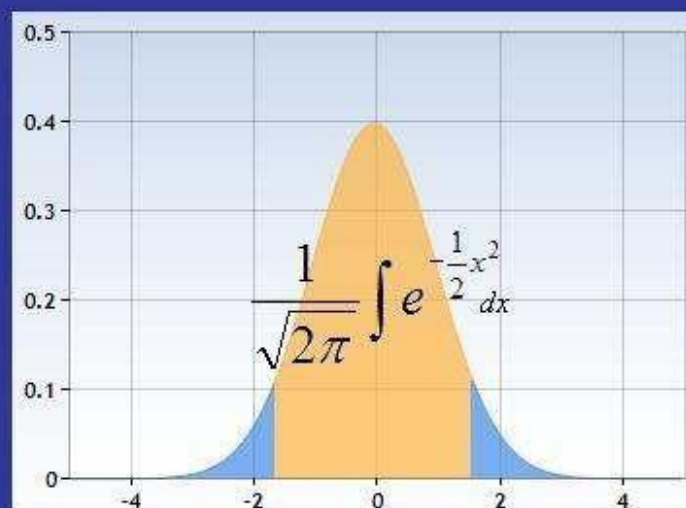


## Powerful Forecasting With MS Excel



Joe Choong

Readers will be provided a link to download the software and Excel files that are used in the book after payment. Please visit <http://www.xlpert.com> for more information on the book. The Excel files are compatible with Excel 2003, 2007 and 2010.

## **Chapter 1: Basic Forecasting Methods**

p/g 1

**Moving Average** – Forecast a country farm production

**Exponential Smoothing** – Forecast a country farm production

**Holts Method** – Forecast a winter clothing sales

**Holts Winters** – Forecast fishing rods sales

## **Chapter 2: ARIMA / Box Jenkins**

p/g 24

**Arima Modeling**

**Subjective Method**

Example 1: Daily Sales Forecasting

**Objective Method**

Example 2: Forecast daily electricity production

## **Chapter 3: Monte Carlo Simulation**

p/g 81

Example 1: Coin Tossing

Example 2: Sales Forecasting

Example 3: Modeling Stock Prices

## **Chapter 4: K Nearest Neighbors**

**p/g 103**

Example 1: KNN for classification of home ownership

Example 2: KNN for time series prediction with stock market data

Example 3: Cross Validation With MSE

## **Chapter 5: Neural Network With MS Excel**

**p/g 117**

Example 1: Credit Approval Model

Example 2: Sales Forecasting

Example 3: Predicting Stock Prices (DJIA)

Example 4: Predicting Real Estate Value

Example 5: Classify Type Of Flowers

## **Chapter 6: Markov Chain with MS Excel**

**p/g 218**

Example 1: Predict delivery time for a courier company

Example 2: Predict brand loyalty of soft drink customers

## **Chapter 7: Bayes' Theorem with MS Excel**

**p/g 231**

Example 1: Predict rainy or sunny day

Example 2: Predict stock market trend

## **Appendix A**

p/g 239

**How to use nn\_Solve**

## **Appendix B**

p/g 241

**Simple Linear Regression For Forecasting**

## **Appendix C**

p/g 251

**How to access Excel Solver 2007 and 2010**

# Chapter 1:

## Basic Forecasting Methods

### Introduction

Forecasting is the estimation of the value of a variable (or set of variables) at some future point in time. In this book we will consider some methods for forecasting. A forecasting exercise is usually carried out in order to provide an aid to decision-making and in planning the future. Typically all such exercises work on the premise that if we can predict what the future will be like we can modify our behaviour now to be in a better position, than we otherwise would have been, when the future arrives. Applications for forecasting include:

- inventory control/production planning - forecasting the demand for a product enables us to control the stock of raw materials and finished goods, plan the production schedule, etc
- investment policy - forecasting financial information such as interest rates, exchange rates, share prices, the price of gold, etc. This is an area in which no one has yet developed a reliable (consistently accurate) forecasting technique (or at least if they have they haven't told anybody!)
- economic policy - forecasting economic information such as the growth in the economy, unemployment, the inflation rate, etc is vital both to government and business in planning for the future.

### Types of forecasting problems/methods

One way of classifying forecasting problems is to consider the timescale involved in the forecast i.e. how far forward into the future we are trying to forecast. Short, medium and long-term are the usual categories but the actual meaning of each will vary according to the situation that is being studied, e.g. in forecasting energy demand in order to construct power stations 5-10 years would be short-term and 50 years would be long-term, whilst in forecasting consumer demand in many business situations up to 6 months would be short-term and over a couple of years long-term. The table below shows the timescale associated with business decisions.

<b>Timescale</b>	<b>Type of decision</b>	<b>Examples</b>
Short-term up to 3-6 months	Operating	Inventory control, Production planning, distribution
Medium-term up to 3-6 months to 2 years	Tactical	Leasing of plant and equipment, Employment changes
Long-term above 2 years	Strategic	Research and development Acquisitions and mergers, Product changes

The basic reason for the above classification is that different forecasting methods apply in each situation, e.g. a forecasting method that is appropriate for forecasting sales next month (a short-term forecast) would probably be an inappropriate method for forecasting sales in five years time (a long-term forecast). In particular note here that the use of numbers (data) to which quantitative techniques are applied typically varies from very high for short-term forecasting to very low for long-term forecasting when we are dealing with business situations. Some areas can encompass short, medium and long term forecasting like stock market and weather forecasting.

Forecasting methods can also be classified into several different categories:

- qualitative methods - where there is no formal mathematical model, often because the data available is not thought to be representative of the future (long-term forecasting)
- quantitative methods - where historical data on variables of interest are available—these methods are based on an analysis of historical data concerning the time series of the specific variable of interest and possibly other related time series and also examines the cause-and-effect relationships of the variable with other relevant variables
- time series methods - where we have a single variable that changes with time and whose future values are related in some way to its past values.

This chapter will give a brief overview of some of the more widely used techniques in the rich and rapidly growing field of time series modeling and analysis. We will deal with qualitative and quantitative forecasting methods on later chapters.

Areas covered in this Chapter are:

- a) Simple Moving Average
- b) Weighted Moving Average
- c) Single Exponential Smoothing
- d) Double Exponential Smoothing (Holt's Method)

e) Triple Exponential Smoothing (Holt Winters Method)

These methods are generally used to forecast time series. A time series is a series of observations collected over evenly spaced intervals of some quantity of interest. e.g. number of phone calls per hour, number of cars per day, number of students per semester, daily stock prices...

Let

$y_i$  = observed value  $i$  of a time series ( $i = 1, 2, \dots, t$ )

$\hat{y}_i$  = forecasted value of  $y_i$

$e_i$  = error for case  $i = y_i - \hat{y}_i$

Sometimes the forecast is too high (negative error) and sometimes it is too low (positive error).

The accuracy of the forecasting method is measured by the forecasting errors. There are two popular methods for assessing forecasting accuracy:

*Mean Absolute Deviation (MAD)*

**MAD** =  $(\sum |e_i|)/n$   $\Rightarrow$  sum of errors divided by the number of periods in the forecast

Where  $n$  is the number of periods in the forecast;  $e_i$  = error

Units of MAD are same as units of  $y_i$

*Mean Squared Error (MSE)*

**MSE** =  $(\sum e_i^2)/n$   $\Rightarrow$  sum of squared errors divided by the number of periods in the forecast

**a) Single Moving Average**

This simplest forecasting method is the moving average forecast. The method simply averages of the last  $n$  observations. It is useful for time series with a slowly changing mean. Use an average of the  $N$  most recent observations to forecast the next period. From one period to the next, the average “moves” by replacing the oldest observation in the average with the most recent observation. In the process, short-term irregularities in the data series are “smoothed out”.

The general expression for the moving average is



$$\hat{y}_t = [y_t + y_{t-1} + \dots + y_{t-N+1}] / N$$

How do you choose N? We want the value of N that gives us the best forecasting accuracy. (i.e. minimizes MAD or MSE). Let's look at an example. Open workbook TimeSeries in the folder Chapter 1 and goto to *Worksheet (Milk Data)*. Here we have the daily record of milk production of a country farm.

	A	B	C	D	E	F
1	<b>Moving Average Example for Milk Production</b>					
2	<b>Use N = 3</b>					
3						
4		<b>yi</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>	
5	<b>Day</b>	<b>Gallons</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>	
6	1	56				
7	2	58				
8	3	45				
9	4	50				
10	5	52				
11	6	55				
12	7	57				
13	8	53				
14	9	54				
15	10	48				
16	11	51				
17	12	54				
18	13	50				
19	14	48				
20	15	53				
21	16	54				
22	17	52				
23	18	55				
24	19	53				
25	20	49				
26						
27			<b>MAD</b>			<b>MSE</b>
28						
29			<b>se</b>			

Figure 1.1

Next, goto *Worksheet (ma solution)*. We will use N = 3, so we enter the formula =AVERAGE(B6:B8) in cell C9. This formula is simply  $[y_t + y_{t-1} + y_{t-2}] / 3$ . (see Fig 1.2 below)


	A	B	C	D	E	F
1	<b>Moving Average Example for Milk Production</b>					
2	<b>Use N = 3</b>					
3						
4		<b>yi</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>	
5	<b>Day</b>	<b>Gallons</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>	
6	1	56				
7	2	58				
8	3	45				
9	4	50	53.0	3.0	9.0	
10	5	52	51.0	1.0	1.0	
11	6	55	49.0	6.0	36.0	
12	7	57	52.3	4.7	21.8	
13	8	53	54.7	1.7	2.8	
14	9	54	55.0	1.0	1.0	
15	10	48	54.7	6.7	44.4	
16	11	51	51.7	0.7	0.4	
17	12	54	51.0	3.0	9.0	
18	13	50	51.0	1.0	1.0	
19	14	48	51.7	3.7	13.4	
20	15	53	50.7	2.3	5.4	
21	16	54	50.3	3.7	13.4	
22	17	52	51.7	0.3	0.1	
23	18	55	53.0	2.0	4.0	
24	19	53	53.7	0.7	0.4	
25	20	49	53.3	4.3	18.8	
26	21		52.3			
27			<b>MAD</b>	2.7	10.7	<b>MSE</b>
28						
29			<b>se</b>	3.4		

Figure 1.2

After that we fill down the formula until C25. We take the absolute error at column D and squared error at column E. See all the formula enter in Fig 1.3 below.

We have the MAD = 2.7 in cell D27 and MSE = 10.7 in cell E27. If at day 20 we have 49 gallons, how do you forecast the production at day 21? To forecast simply fill down the formula in cell C25 to cell C26 or enter the formula =AVERAGE(B23:B25). Here the result is 52.3 gallons. (see Figure 1.2)

As you can see with this example, the moving average method is very simple to build. You can also experiment with N = 4 or 5... to see whether you can get a lower MAD and MSE

	A	B	C	D	E	
1	<b>Moving Average Example</b>					
2	<b>Use N = 3</b>					
3						
4		<b>y<sub>i</sub></b>	<b>y<sub>hat</sub><sub>i</sub></b>	<b> e<sub>i</sub> </b>	<b>e<sub>i</sub><sup>2</sup></b>	
5	<b>Day</b>	<b>Gallons</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>	
6	1	56				
7	2	58				
8	3	45				
9	4	50	=AVERAGE(B6:B8)	=ABS(B9-C9)	=D9^2	
10	5	52	=AVERAGE(B7:B9)	=ABS(B10-C10)	=D10^2	
11	6	55	=AVERAGE(B8:B10)	=ABS(B11-C11)	=D11^2	
12	7	57	=AVERAGE(B9:B11)	=ABS(B12-C12)	=D12^2	
13	8	53	=AVERAGE(B10:B12)	=ABS(B13-C13)	=D13^2	
14	9	54	=AVERAGE(B11:B13)	=ABS(B14-C14)	=D14^2	
15	10	48	=AVERAGE(B12:B14)	=ABS(B15-C15)	=D15^2	
16	11	51	=AVERAGE(B13:B15)	=ABS(B16-C16)	=D16^2	
17	12	54	=AVERAGE(B14:B16)	=ABS(B17-C17)	=D17^2	
18	13	50	=AVERAGE(B15:B17)	=ABS(B18-C18)	=D18^2	
19	14	48	=AVERAGE(B16:B18)	=ABS(B19-C19)	=D19^2	
20	15	53	=AVERAGE(B17:B19)	=ABS(B20-C20)	=D20^2	
21	16	54	=AVERAGE(B18:B20)	=ABS(B21-C21)	=D21^2	
22	17	52	=AVERAGE(B19:B21)	=ABS(B22-C22)	=D22^2	
23	18	55	=AVERAGE(B20:B22)	=ABS(B23-C23)	=D23^2	
24	19	53	=AVERAGE(B21:B23)	=ABS(B24-C24)	=D24^2	
25	20	49	=AVERAGE(B22:B24)	=ABS(B25-C25)	=D25^2	
26	21		=AVERAGE(B23:B25)			
27			<b>MAD</b>	=AVERAGE(D9:D25)	=AVERAGE(E9:E25)	<b>MSE</b>
28						
29			<b>se</b>	=1.25*D27		

Figure 1.3

## b) Weighted Moving Average

In the moving averages method above, each observation in the average is equally weighted whereas the weighted moving averages method, typically the most recent observation carries the most weight in the average. The general expression for weighted moving average is

$$\hat{y}_i = [w_t * y_t + w_{t-1} * y_{t-1} + \dots + w_{t-N+1} * y_{t-N+1}]$$

Let  $w_i$  = weight for observation  $i$

$\sum w_t = 1 \Rightarrow$  The sum of the weights is 1.

To illustrate, let's rework the milk example. Goto *Worksheet (wma solution)*. Using a weight of .5 enter in cell H2 for the most recent observation, a weight of .3 in cell H3 for the next most recent observation, and a weight of .2 in cell H4 for the oldest observation. (see Figure 1.4)

Again using  $N=3$ , we enter the formula  $=(B8*\$H\$2+B7*\$H\$3+B6*\$H\$4)$  in cell C9 and fill down to C25. (Note: we use the \$ sign to lock the cell in H2, H3 and H4).

	A	B	C	D	E	F	G	H
1	<b>Weighted Moving Average Example for Milk Production</b>							
2	<b>Use <math>H = 3</math></b>						w(I-1)	0.50
3							w(I-2)	0.30
4		<b>yi</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>		w(I-3)	0.20
5	<b>Day</b>	<b>Gallons</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>			<b>1</b>
6	1	56						
7	2	58						
8	3	45						
9	4	50	51.1	1.1	1.2			
10	5	52	50.1	1.9	3.6			
11	6	55	50.0	5.0	25.0			
12	7	57	53.1	3.9	15.2			
13	8	53	55.4	2.4	5.8			
14	9	54	54.6	0.6	0.4			
15	10	48	54.3	6.3	39.7			
16	11	51	50.8	0.2	0.0			
17	12	54	50.7	3.3	10.9			
18	13	50	51.9	1.9	3.6			
19	14	48	51.4	3.4	11.6			
20	15	53	49.8	3.2	10.2			
21	16	54	50.9	3.1	9.6			
22	17	52	52.5	0.5	0.3			
23	18	55	52.8	2.2	4.8			
24	19	53	53.9	0.9	0.8			
25	20	49	53.4	4.4	19.4			
26			51.4					
27			<b>MAD</b>	2.6	9.5	<b>MSE</b>		
28								
29			<b>se</b>	3.3				

Figure 1.4

The absolute error and squared error are entered in column D and E respectively. Cell D27 show the MAD and cell E27 is the MSE.

Could we have used a better set of weights than those above in order to have a better forecast? Let use Excel Solver to check for you. Choose **Tools, Solver** from the menu.

*(See Appendix C on how to access Excel 2007 and Excel 2010 Solver)*

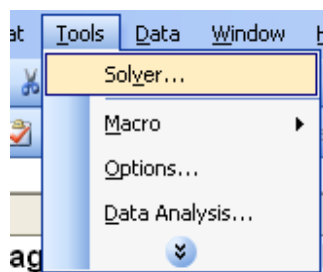


Figure 1.5

Upon invoking Solver from the Tools menu, a dialogue box appears as in Fig 1.6 below

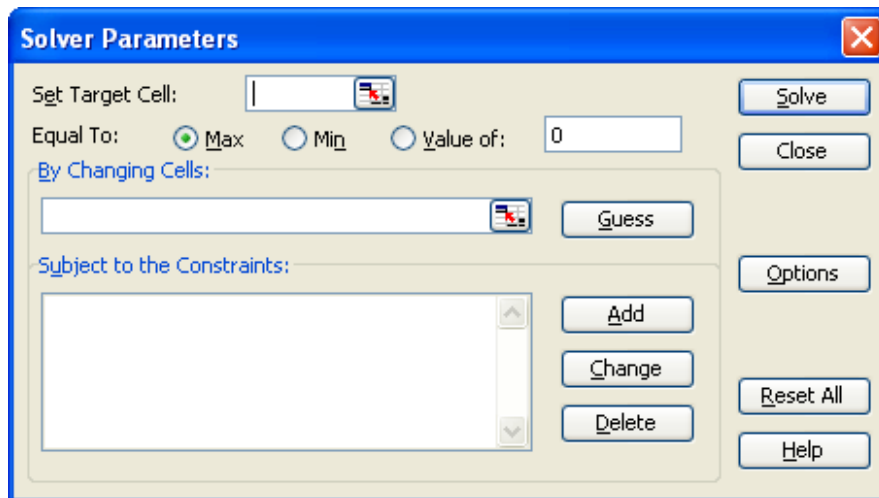


Figure 1.6

Next we will enter all the parameters in this dialogue box. It will look like Fig 1.7 below after all the parameters are entered.

Set Target Cell: MAD (D27)

Equal to: Min

By Changing Cells: the weights at H2:H4

Constraints are that the weights that must add up to 1 at H5

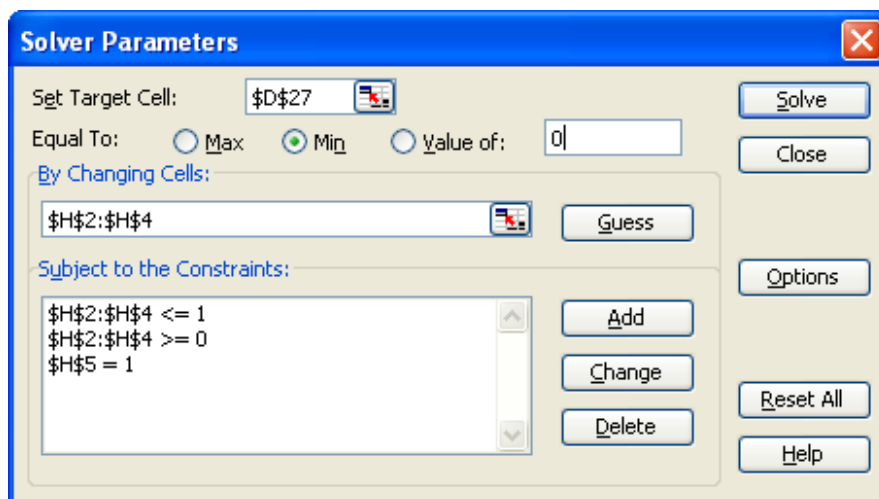


Figure 1.7

Click the Solve button. Solver will start to optimize.

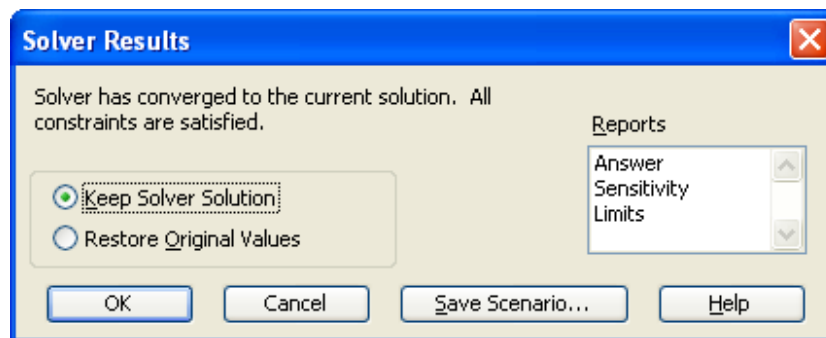


Figure 1.8

And Keep the Solver solution

Initially we have the  $MAD = 2.6$  in cell D27 (see Figure 1.4 above). After optimizing with Excel Solver the  $MAD$  is  $= 2.3$  and the weights are 0.57, 0.11, 0.32 in cell H2:H4 respectively. (see Fig 1.9 below) So we have improved our model using Excel Solver.

If at day 20 we have 49 gallons, how do you forecast the production at day 21? To forecast simply fill down the formula in cell C25 to cell C26 or enter the formula  $= (B25 * \$H\$2 + B24 * \$H\$3 + B23 * \$H\$4)$ . Here the result is 51.4 gallons. (see Fig 1.9 below)

	A	B	C	D	E	F	G	H
1	<b>Weighted Moving Average Example for Milk Production</b>							
2	<b>Use <math>n = 3</math></b>						w(l-1)	0.57
3							w(l-2)	0.11
4		<b>yi</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>		w(l-3)	0.32
5	<b>Day</b>	<b>Gallons</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>			<b>1</b>
6	1	56						
7	2	58						
8	3	45						
9	4	50	50.0	0.0	0.0			
10	5	52	52.0	0.0	0.0			
11	6	55	49.5	5.5	29.9			
12	7	57	53.1	3.9	15.6			
13	8	53	55.2	2.2	4.7			
14	9	54	54.1	0.1	0.0			
15	10	48	54.8	6.8	46.9			
16	11	51	50.3	0.7	0.5			
17	12	54	51.6	2.4	5.7			
18	13	50	51.7	1.7	3.0			
19	14	48	50.8	2.8	7.7			
20	15	53	50.2	2.8	8.1			
21	16	54	51.5	2.5	6.4			
22	17	52	52.0	0.0	0.0			
23	18	55	52.5	2.5	6.0			
24	19	53	54.3	1.3	1.8			
25	20	49	52.9	3.9	15.3			
26			51.4					
27			<b>MAD</b>	2.3	8.9	<b>MSE</b>		
28								
29			<b>se</b>	2.9				

Figure 1.9

As you can see with this example, the weighted moving average method is more complicated to build and will give us better result.

One disadvantage of using moving averages for forecasting is that in calculating the average all the observations are given equal weight (namely  $1/L$ ), whereas we would expect the more recent observations to be a better indicator of the future (and accordingly ought to be given greater weight). Also in moving averages we only use recent observations, perhaps we should take into account all previous observations.

One technique (which we will look at next) known as exponential smoothing (or, more accurately, single exponential smoothing) gives greater weight to more recent observations *and* takes into account all previous observations.

### c) Single Exponential Smoothing

Exponential Smoothing is a very popular scheme to produce a smoothed time series. Whereas in Single Moving Averages the past observations are weighted equally, Exponential Smoothing assigns *exponentially decreasing weights* as the observation get older. In other words, *recent observations are given relatively more weight in forecasting than the older observations*.

In the case of moving averages, the weights assigned to the observations are the same and are equal to  $1/N$ . In exponential smoothing, however, there are one or more *smoothing parameters* to be determined (or estimated) and these choices determine the weights assigned to the observations.

Let's look at Single Exponential Smoothing first. Recall that  $y_i$  = observed value  $i$  and  $\hat{y}_i$  = forecasted value  $i$ . The general expression is

$$\hat{y}_{i+1} = \alpha y_i + (1 - \alpha) \hat{y}_i$$

Which says

forecast for the next period = forecast for this period + smoothing constant \* error for this period

where  $0 \leq \alpha \leq 1$

The forecast for the current period is a weighted average of all past observations. The weight given to past observations declines exponentially. The larger the  $\alpha$ , the more weight is given to recent observations.

So you can see here that the exponentially smoothed moving average takes into account all of the previous observations, compare the moving average above where only a few of the previous observations were taken into account.

Don't worry above the equation above. I'll show you how it is easily implemented in Excel.

Again, this method works best when the time series fluctuates about a constant base level. Simple exponential smoothing is an extension of weighted moving averages where the greatest weight is placed on the most recent value and then progressively smaller weights are placed on the older values.

To start the process, assume that  $\hat{y}_1 = y_1$  unless told otherwise. Do not use this observation in your error calculations though.

Let's now rework the MILK problem. Goto the *Worksheet (exp solution)*. We use a simple exponential smoothing with an  $\alpha$  of 0.3 as initial value in cell G4.

x

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> more information on the book

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x



It is customary to assume that  $\hat{y}_1 = y_1$  so we enter  $C6 = B6$ . After that starting at period 2 in cell C7 we enter the formula  $=G\$4*B6+(1-G\$4)*C6$  and fill down the formula until cell C25. (Note: we use the \$ sign to lock the cell in G4). This is how the formula  $\hat{y}_{i+1} = \hat{y}_i + \alpha (y_i - \hat{y}_i)$  is entered as Excel formula.

The absolute error and squared error are entered in column D and E respectively. Cell D27 show the MAD and cell E27 is the MSE. We will use Excel Solver to find a better value for  $\alpha$  and also minimize the MAD and MSE.

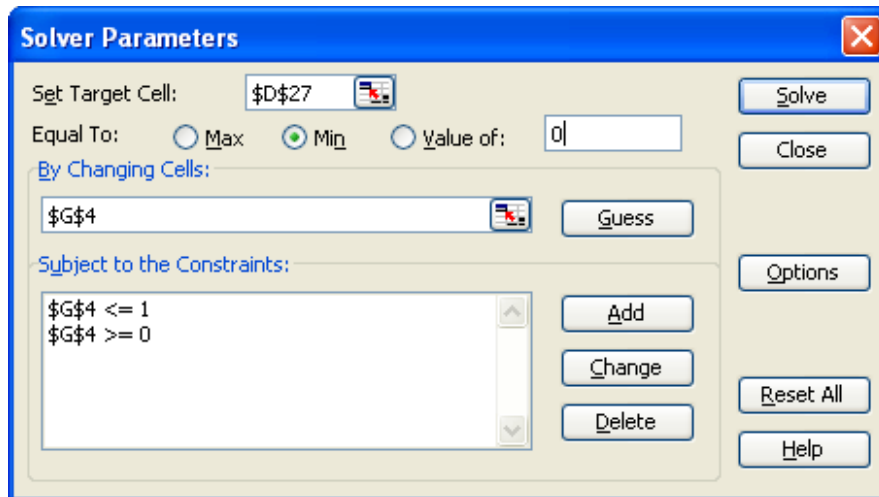


Figure 1.11

We will enter all the parameters in this dialogue box. It will look like Fig 1.11 above after all the parameters are entered.

**Set Target Cell:** MAD (D27);

**Equal to:** Min

**By Changing Cells:** the weights at G4

**Constraints** are that the  $0 \leq G4 \leq 1$ .

Click the Solve button. Solver will start to optimize.

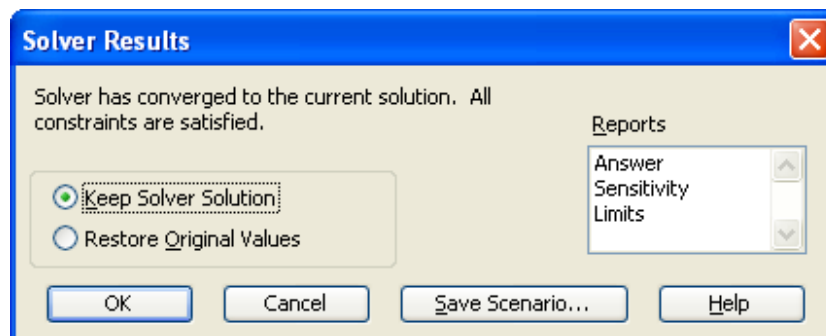


Figure 1.12

And Keep the Solver solution

The MAD in cell D27 is equal 2.8 and the  $\alpha$  in cell G4 is equal 0.31109. We have made a little improvement.

If at day 20 we have 49 gallons, how do you forecast the production at day 21? To forecast simply fill down the formula in cell C25 to cell C26 or enter the formula  $=\$G\$4*B25+(1-\$G\$4)*C25$ . Here the result is 51.8 gallons. (see Figure 1.13 below)

	A	B	C	D	E	F	G
1	<b>Simple Exponential Smoothing Example for Milk Production</b>						
2	<b>Use alpha = 0.3</b>						
3							<b>alpha</b>
4		<b>yi</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>		0.31109
5	<b>Day</b>	<b>Gallons</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>		
6	1	56	56.0				
7	2	58	56.0	2.0	4.0		
8	3	45	56.6	11.6	135.1		
9	4	50	53.0	3.0	9.0		
10	5	52	52.1	0.1	0.0		
11	6	55	52.0	3.0	8.7		
12	7	57	53.0	4.0	16.3		
13	8	53	54.2	1.2	1.5		
14	9	54	53.8	0.2	0.0		
15	10	48	53.9	5.9	34.7		
16	11	51	52.1	1.1	1.1		
17	12	54	51.7	2.3	5.2		
18	13	50	52.4	2.4	5.9		
19	14	48	51.7	3.7	13.5		
20	15	53	50.5	2.5	6.1		
21	16	54	51.3	2.7	7.3		
22	17	52	52.1	0.1	0.0		
23	18	55	52.1	2.9	8.4		
24	19	53	53.0	0.0	0.0		
25	20	49	53.0	4.0	16.0		
26			51.8				
27			<b>MAD</b>	2.8	14.4	<b>MSE</b>	
28							
29			<b>se</b>	3.5			

Figure 1.13

As you can see with this example, the simple exponential smoothing method is a little more complicated to build and should give us better result.

Exponential smoothing is useful when there is no trend. However if the data is trending, we need to use the Double Exponential Smoothing method which will be discussed below.

#### d) Double Exponential Smoothing (Holt's Method)

Double exponential smoothing is defined as Exponential smoothing of Exponential smoothing. Exponential smoothing does not excel in following the data when there is a trend. This situation can be improved by the introduction of a second equation with a second constant,  $\beta$ , the trend component, which must be chosen in conjunction with  $\alpha$ , the mean component. Double exponential smoothing is defined in the following manner :

$$\hat{y}_{i+1} = E_i + T_i, i = 1, 2, 3, \dots$$

where

$$E_i = \alpha y_i + (1-\alpha)(E_{i-1} + T_{i-1})$$

$$T_i = \beta(E_i - E_{i-1}) + (1-\beta)T_{i-1}$$

$0 < \alpha \leq 1$  and  $\beta$  is another smoothing constant where  $0 \leq \beta \leq 1$ .

This method works best when the time series has a positive or negative trend (i.e. upward or downward). After observing the value of the time series at period  $i$  ( $y_i$ ), this method computes an estimate of the base, or expected level of the time series ( $E_i$ ) and the expected rate of increase or decrease per period ( $T_i$ ). It is customary to assume that  $E_1 = y_1$  and unless told otherwise, and assume  $T_1 = 0$ .

To use the method, first calculate the base level  $E_i$  for time  $i$ . Then compute the expected trend value  $T_i$  for time period  $i$ . Finally, compute the forecast  $\hat{y}_{i+1}$ . Once an observation  $y_i$  is made, calculate the error and continue the process for the next time period. If you want to forecast  $k$  periods ahead, use the following logic.

$$\hat{y}_{i+k} = E_i + kT_i \quad \text{where } k = 1, 2, 3, \dots$$

Open the *Worksheet (double exp solution)*.

The data is for the monthly sales in thousands for a clothing company. Initially we use the value of 0.2 (cell I4) for  $\alpha$  and 0.3 (cell J4) for  $\beta$ . For  $E_1 = y_1$ , we enter =B6 in cell C6.

Then we enter the formula =I\$4\*B7+(1-I\$4)\*(C6+D6) in cell C7 and fill down to C29. As you can see from the formula, we start at period 2 in cell C7. (Note: we use the \$ sign to lock the cell in I4). This is how the formula  $E_i = \alpha y_i + (1-\alpha)(E_{i-1} + T_{i-1})$  is entered as Excel formula.

After that, we will calculate  $T_i = \beta(E_i - E_{i-1}) + (1-\beta)T_{i-1}$ . It is common practice to enter 0 at period 1 i.e cell D6. Starting at period 2, in cell D7, we enter the formula =J\$4\*(C7-C6)+(1-J\$4)\*D6 and fill down to D29. This is how the formula  $T_i = \beta(E_i - E_{i-1}) + (1-\beta)T_{i-1}$  is entered as Excel formula.

Now we combine the 2 formula together and enter the formula =C6+D6 in E7 and fill down to E29 i.e the formula **yhat<sub>i+k</sub> = E<sub>i</sub> +kT<sub>i</sub>**. The absolute error and squared error are entered in column F and G respectively. Cell F31 show the MAD and cell G31 is the MSE. We will use Excel Solver to minimize the MAD and to find a better value or optimize for  $\alpha$  and  $\beta$ . (see Figure 1.14 below)

	A	B	C	D	E	F	G	H	I	J
1	<b>Exponential Smoothing with Trend Example for Winter Wear Sales</b>									
2	<b>Use alpha = 0.2 and beta = 0.3</b>									
3									<b>alpha</b>	<b>beta</b>
4		<b>yi</b>	<b>Ei</b>	<b>Ti</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>		0.2	0.3
5	<b>Month</b>	<b>units</b>	<b>Base</b>	<b>Trend</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>			
6	1	684	684	0						
7	2	590	665.20	-5.64	684.00	94.00	8836.00			
8	3	750	677.65	-0.21	659.56	90.44	8179.39			
9	4	880	717.95	11.94	677.43	202.57	41032.82			
10	5	885	760.91	21.25	729.89	155.11	24059.78			
11	6	788	783.33	21.60	782.16	5.84	34.14			
12	7	1004	844.74	33.54	804.92	199.08	39631.45			
13	8	1111	924.82	47.51	878.28	232.72	54158.13			
14	9	1160	1009.86	58.77	972.33	187.67	35219.97			
15	10	1044	1063.70	57.29	1068.63	24.63	606.62			
16	11	1500	1196.79	80.03	1120.99	379.01	143647.44			
17	12	1610	1343.46	100.02	1276.82	333.18	111007.93			
18	13	1250	1404.78	88.41	1443.48	193.48	37433.03			
19	14	1730	1540.55	102.62	1493.19	236.81	56078.34			
20	15	1990	1712.54	123.43	1643.17	346.83	120289.66			
21	16	2030	1874.77	135.07	1835.97	194.03	37649.11			
22	17	2100	2027.87	140.48	2009.84	90.16	8128.17			
23	18	1760	2086.68	115.98	2168.35	408.35	166753.74			
24	19	2300	2222.13	121.82	2202.66	97.34	9474.56			
25	20	2620	2399.16	138.38	2343.95	276.05	76204.10			
26	21	2566	2543.23	140.09	2537.54	28.46	809.90			
27	22	2710	2688.66	141.69	2683.32	26.68	711.68			
28	23	2800	2824.28	139.87	2830.35	30.35	921.02			
29	24	2850	2941.32	133.02	2964.15	114.15	13029.74			
30	25				3074.34					
31					<b>MAD</b>	171.61	43212.90	<b>MSE</b>		
32					<b>se</b>	214.51				
33										
34										
35										
36										
37	32		Forecast for Month 32		4005.482					

Figure 1.14

Invoke Excel Solver to minimize the MAD. We enter all the parameters in this dialogue box. It will look like Fig 1.15 below after all the parameters are entered.

**Set Target Cell:** MAD (F31)

**Equal to:** Min

**By Changing Cells:** the weights at I4:J4

**Constraints** are that the  $0 \leq I4 \leq 1$ ,  $0 \leq J4 \leq 1$

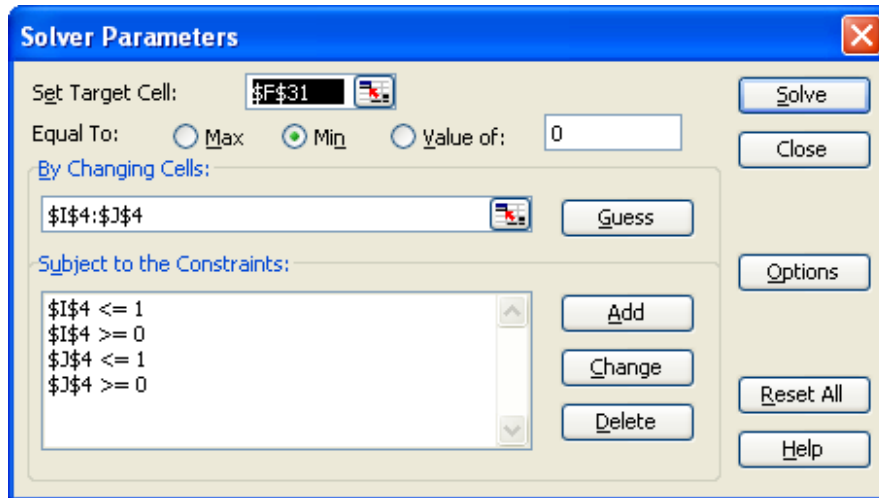


Figure 1.15

Click the Solve button. Solver will start to optimize.

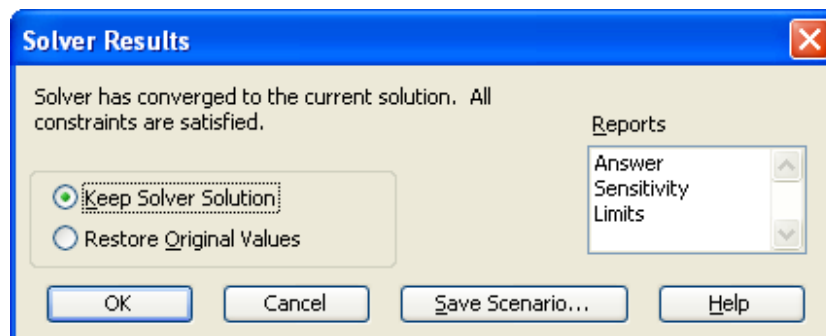


Figure 1.16

And Keep the Solver solution.

The MAD in cell F31 has been minimize and is equal to 140.91 and the  $\alpha$  in cell I4 is equal 0.126401. The  $\beta$  in cell J4 = 1. We have made a little improvement after optimizing with Excel Solver.

If at month 24 we have 2,850,000, how do you forecast the production at month 25? To forecast simply fill down the formula in cell E29 to cell E30 or enter the formula =C29+D29. Here the result is 3,037,810. (see Figure 1.17 below)

If you want to forecast k periods ahead, use the following logic.

$$\hat{y}_{i+k} = E_i + kT_i$$

In this example we want to forecast sales at month 32 i.e. 8 months ahead, so we enter =C29+8\*\$D\$29 as you can see in cell E37. The result is 3,868,383. (see Figure 1.17)

$$E_i = C29$$

$$T_i = D29$$

$$k = 8$$

	A	B	C	D	E	F	G	H	I	J
1	<b>Exponential Smoothing with Trend Example for Winter Wear Sales</b>									
2	<b>Use alpha = 0.2 and beta = 0.3</b>									
3									<b>alpha</b>	<b>beta</b>
4		<b>yi</b>	<b>Ei</b>	<b>Ti</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>		0.126401	1
5	<b>Month</b>	<b>units</b>	<b>Base</b>	<b>Trend</b>	<b>Forecast</b>	<b>absolute error</b>	<b>squared error</b>			
6	1	684	684	0						
7	2	590	672.12	-11.88	684.00	94.00	8836.00			
8	3	750	671.58	-0.54	660.24	88.76	8057.46			
9	4	880	697.46	25.88	671.05	208.95	43661.24			
10	5	885	743.77	46.31	723.34	161.66	26135.44			
11	6	788	789.82	46.05	790.08	2.08	4.33			
12	7	1004	857.12	67.30	835.87	168.13	28269.17			
13	8	1111	948.00	90.88	924.42	186.58	34812.80			
14	9	1160	1054.20	106.19	1038.89	121.11	14668.47			
15	10	1044	1145.68	91.48	1160.39	116.39	13546.29			
16	11	1500	1270.38	124.70	1237.16	262.84	69085.69			
17	12	1610	1422.25	151.87	1395.09	214.91	46187.69			
18	13	1250	1533.15	110.90	1574.12	324.12	105055.18			
19	14	1730	1654.92	121.76	1644.05	85.95	7386.77			
20	15	1990	1803.65	148.73	1776.68	213.32	45504.62			
21	16	2030	1962.19	158.54	1952.37	77.63	6025.86			
22	17	2100	2118.11	155.92	2120.73	20.73	429.56			
23	18	1760	2209.05	90.95	2274.03	514.03	264223.21			
24	19	2300	2300.00	90.95	2300.00	0.00	0.00			
25	20	2620	2419.90	119.90	2390.95	229.05	52465.25			
26	21	2566	2543.11	123.21	2539.80	26.20	686.49			
27	22	2710	2671.84	128.73	2666.32	43.68	1907.75			
28	23	2800	2800.50	128.66	2800.58	0.58	0.33			
29	24	2850	2919.16	118.65	2929.16	79.16	6266.64			
30	25				3037.81					
31					<b>MAD</b>	140.91	34052.88	<b>MSE</b>		
32										
33					<b>se</b>	176.13				
34										
35										
36										
37	32		Forecast for Month 32		3868.383					

Figure 1.17

As you can see with this example, the double exponential smoothing method is a little more complicated to build and should give us better result. What happens if the data show trend as well as seasonality? In this case double exponential smoothing will not work. We need to use the Triple Exponential Smoothing method which will be discussed below.

#### e) Triple Exponential Smoothing (Holt Winters Method)

This method is appropriate when trend and seasonality are present in the time series. It decomposes the times series down into three components: base, trend and seasonal components.

Let  $s_i$  = seasonal factor for period  $i$

If  $s_i = 1$ , then season is “typical”

If  $s_i < 1$ , then season is smaller than “typical”

If  $s_i > 1$ , then season is larger than “typical”

When an actual observation is divided by its corresponding seasonal factor, it is said to be “deseasonalized.” (i.e. the seasonal component has been removed.) This allows us to make meaningful comparisons across time periods.

Let  $c$  = the number of periods in a cycle (12 if months of year, 7 if days of week, ...)

The relevant formulas for this method follow.

$$E_i = \alpha (y_i / S_{i-c}) + (1-\alpha) (E_{i-1} + T_{i-1})$$

$$T_i = \beta (E_i - L_{i-1}) + (1-\beta) T_{i-1}$$

$$S_i = \gamma (y_i / L_i) + (1-\gamma) s_{i-c}$$

$$\hat{y}_{i+1} = (E_i + T_i) s_{i+1-c}$$

where  $\gamma$  is another smoothing constant between 0 and 1

This means that Holt-Winters' smoothing is similar to exponential smoothing if  $\beta$  and  $\gamma = 0$ . It will be similar to double exponential smoothing if  $\gamma = 0$ .

To start this method, we need  $L_1$ ,  $T_1$ , and a seasonal factor for each period in the cycle.

An easy way for developing initial estimates of the seasonal factors is to collect  $c$  observations and let:

$$s_i = y_i / [(1/c)(y_1 + y_2 + \dots y_c)]$$

Then assume that  $E_c = y_c/s_c$  and that  $T_c = 0$ .

So the steps of the process can be summarized as:

- i. forecast for period  $i$
- ii. collect observation for period  $i$
- iii. calculate smoothed average and the error for period  $i$
- iv. calculate the trend for period  $i$
- v. calculate the seasonal factor for period  $i$
- vi.  $i = i+1$ , go back to step 1

If you want to forecast k periods ahead, use the following logic

$$\hat{y}_{i+k} = (E_i + kT_i) s_{i+k-c}$$

Let's start building the model. Goto *Worksheet (triple exp solution)*

	A	B	C	D	E	F	G	H	I	J
1			Triple Exponential Smoothing Method for Fishing Rods Sales							
2			yi	Ei	Ti	si	yhati	ei	ei^2	
3		Month	Rods Sales	Smoothed Avg	Trend	Seasonal Factors	Forecast	Abs Error	Error Squared	
4	jan	1	7			0.21				
5	feb	2	5			0.15				
6	mar	3	15			0.45				
7	apr	4	25			0.74				
8	may	5	42			1.25				
9	jun	6	48			1.43				
10	jul	7	70			2.08				
11	aug	8	75			2.23				
12	sep	9	40			1.19				
13	oct	10	30			0.89				
14	nov	11	25			0.74				
15	dec	12	22	33.67	0.00	0.65				
16	jan	13	10	40.88	3.61	0.23	7.00	3.00	9.00	Alpha 0.50
17	feb	14	7	45.81	4.27	0.15	6.61	0.39	0.15	Beta 0.50
18	mar	15	20	47.48	2.97	0.43	22.31	2.31	5.35	Gamma 0.50
19	apr	16	32	46.77	1.13	0.71	37.47	5.47	29.88	
20	may	17	58	47.20	0.78	1.24	59.76	1.76	3.11	
21	jun	18	60	45.03	-0.70	1.38	68.40	8.40	70.58	
22	jul	19	90	43.81	-0.96	2.07	92.18	2.18	4.75	
23	aug	20	95	42.75	-1.01	2.23	95.46	0.46	0.21	
24	sep	21	60	46.12	1.18	1.24	49.59	10.41	108.36	
25	oct	22	40	46.09	0.58	0.88	42.15	2.15	4.62	
26	nov	23	37	48.25	1.37	0.75	34.66	2.34	5.49	
27	dec	24	30	47.76	0.44	0.64	32.42	2.42	5.87	
28	jan	25	16	59.46	6.07	0.25	10.91	5.09	25.94	
29	feb	26	20	99.14	22.87	0.18	9.87	10.13	102.58	
30	mar	27	50	118.69	21.21	0.43	52.88	2.88	8.27	
31	apr	28	80	126.03	14.27	0.67	99.80	19.80	392.21	
32	may	29	150	130.72	9.49	1.19	173.72	23.72	562.61	
33	jun	30	152	125.21	1.99	1.30	193.36	41.36	1710.70	
34	jul	31	235	120.45	-1.39	2.01	262.89	27.89	777.99	
35	aug	32	250	115.71	-3.06	2.19	264.92	14.92	222.66	
36	sep	33	175	126.63	3.93	1.31	140.20	34.80	1211.29	
37	oct	34	110	127.82	2.56	0.87	114.82	4.82	23.22	
38	nov	35	100	131.44	3.09	0.76	98.40	1.60	2.57	
39	dec	36	80	129.69	0.67	0.63	86.20	6.20	38.49	
40	jan	37					32.29			
41								9.77	221.91	
42								MAD	MSE	
43										
44								12.21		
45								se		
46										
47	aug	44					296.11			

Figure 1.18

Here we have the monthly sales of a company selling fishing rods. The sales peak during fishing season in spring and summer. Therefore we know that it is a yearly season where  $c = 12$ . The first step is to transform or deseasonalize the first year sales data in C4:C15. So we enter the formula  $=C4/((1/12)*SUM(\$C\$4:\$C\$15))$  in cell F4 and fill down to F15. This is actually how  $s_i = y_i / [(1/c)(y_1 + y_2 + \dots y_c)]$  is entered as Excel Function.

Secondly we enter the formula,  $E_i = \alpha (y_i / S_{i-c}) + (1-\alpha) (E_{i-1} + T_{i-1})$  in cell D16 which is  $=\$J\$16*C16/F4+(1-\$J\$16)*(D15+E15)$  and fill down till D39.



After that we enter this formula,  $T_i = \beta(E_i - L_{i-1}) + (1-\beta)T_{i-1}$  in cell E16 which is =J\$18\*(D16-D15)+(1-J\$18)\*E15 and fill down till E39.

The next step is entering the formula  $S_i = \gamma (y_i / L_i) + (1-\gamma) s_{i-c}$  in cell F16 which is =J\$20\*C16/D16+(1-\$J20)\*F4 and fill down till F39.

X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> more information on the book

[illegible]

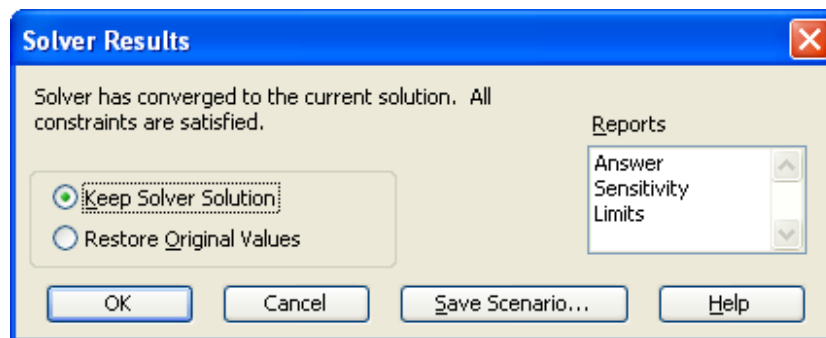


Figure 1.20

And Keep the Solver solution

MSE at cell I41 is 49.17 now compare to 221.91 initially.  $\alpha = 0.46$  in cell J16,  $\beta = 0.05$  in J18 and  $\gamma = 1$  in cell J20. (see Figure 1.21 below)

	A	B	C	D	E	F	G	H	I	J
1			<b>Triple Exponential Smoothing Method for Fishing Rods Sales</b>							
2			<b>yi</b>	<b>Ei</b>	<b>Ti</b>	<b>si</b>	<b>yhati</b>	<b> ei </b>	<b>ei^2</b>	
3		Month	<b>Rods Sales</b>	<b>Smoothed Avg</b>	<b>Trend</b>	<b>Seasonal Factors</b>	<b>Forecast</b>	<b>Abs Error</b>	<b>Error Squared</b>	
4	jan	1	7			0.21				
5	feb	2	5			0.15				
6	mar	3	15			0.45				
7	apr	4	25			0.74				
8	may	5	42			1.25				
9	jun	6	48			1.43				
10	jul	7	70			2.08				
11	aug	8	75			2.23				
12	sep	9	40			1.19				
13	oct	10	30			0.89				
14	nov	11	25			0.74				
15	dec	12	22	33.67	0.00	0.65				
16	jan	13	10	40.27	0.31	0.25	7.00	3.00	9.00	<b>Alpha</b>
17	feb	14	7	43.58	0.45	0.16	6.03	0.97	0.95	<b>Beta</b>
18	mar	15	20	44.42	0.46	0.45	19.61	0.39	0.15	<b>Gamma</b>
19	apr	16	32	44.06	0.43	0.73	33.33	1.33	1.77	<b>1.00</b>
20	may	17	58	45.41	0.47	1.28	55.50	2.50	6.23	
21	jun	18	60	44.14	0.39	1.36	65.41	5.41	29.24	
22	jul	19	90	43.96	0.36	2.05	92.59	2.59	6.68	
23	aug	20	95	43.55	0.33	2.18	98.74	3.74	13.97	
24	sep	21	60	46.91	0.47	1.28	52.14	7.86	61.85	
25	oct	22	40	46.24	0.41	0.87	42.22	2.22	4.91	
26	nov	23	37	48.10	0.48	0.77	34.64	2.36	5.56	
27	dec	24	30	47.36	0.42	0.63	31.75	1.75	3.06	
28	jan	25	16	55.40	0.78	0.29	11.67	4.13	17.08	
29	feb	26	20	87.44	2.23	0.23	9.02	10.98	120.46	
30	mar	27	50	99.45	2.69	0.50	40.38	9.62	92.61	
31	apr	28	80	105.81	2.86	0.76	74.18	5.82	33.93	
32	may	29	150	112.68	3.04	1.33	138.81	11.19	125.33	
33	jun	30	152	113.94	2.96	1.33	157.30	5.30	28.07	
34	jul	31	235	115.93	2.92	2.03	239.32	4.32	18.70	
35	aug	32	250	116.91	2.83	2.14	259.22	9.22	85.06	
36	sep	33	175	127.55	3.19	1.37	153.15	21.85	477.47	
37	oct	34	110	129.10	3.11	0.85	113.10	3.10	9.62	
38	nov	35	100	131.21	3.07	0.76	101.69	1.69	2.86	
39	dec	36	80	130.62	2.90	0.61	85.05	5.05	25.52	
40	jan	37					38.56			
41								5.27	49.17	
42								MAD	MSE	
43										
44								6.58		
45								se		
46										
47	aug	44					328.86			

Figure 1.21

We have made a big improvement after optimizing with Excel Solver.

If at month 36 we sell 80 fishing rods, how do you forecast the production at month 37?  
To forecast simply fill down the formula in cell G39 to cell G40 or enter the formula  $= (D39 + E39) * F28$ . Here the result is 38.56 rods or round up to 39.

If you want to forecast  $k$  periods ahead, use the following logic.

$$\hat{y}_{i+k} = (E_i + kT_i) s_{i+k-c}$$

In this example we want to forecast sales at month 44 i.e. 8 months ahead, so we enter  $= (D39 + 8 * E39) * F35$  as you can see in cell G47. The result is 328.86 or round up to 329 fishing rods. (see Figure 1.21 above)

$$E_i = D39$$

$$T_i = E39$$

$$s_{i+k-c} = F35$$

$$k = 8$$

$$c = 12$$

As you can see with this example, the triple exponential smoothing method is a little more complicated to build and should give us very good result.

## Conclusion

A moving average is commonly used with time series data to smooth out short-term fluctuations and highlight longer-term trends or cycles. The threshold between short-term and long-term depends on the application, and the parameters of the moving average will be set accordingly. For example, it is often used in technical analysis of financial data, like stock prices, returns or trading volumes. It is also used in economics to examine gross domestic product, employment or other macroeconomic time series.

Exponential smoothing has proven through the years to be very useful in many forecasting situations. It was first suggested by C.C. Holt in 1957 and was meant to be used for non-seasonal time series showing no trend. He later offered a procedure (1958) that does handle trends. Winters(1965) generalized the method to include seasonality, hence the name "Holt-Winters Method" or Triple Exponential Smoothing.

All these forecasting methods are very basic but very useful. Time series forecasting methods can be more advanced than those considered in our examples above. These are

based on **AutoRegressive Integrated Moving Average (ARIMA)** models (also known as Box-Jenkins technique). Essentially these assume that the time series has been generated by a probability process with future values related to past values, as well as to past forecast errors. To apply ARIMA models the time series needs to be stationary. A stationary time series is one whose statistical properties such as mean, variance and autocorrelation are constant over time. We will learn how to model ARIMA as a forecasting method in Chapter 2.

## Chapter 2:

### AutoRegressive Integrated Moving Average / Box-Jenkins technique

#### Introduction

**Box-Jenkins Methodology Or Arima Forecasting Method:** Box-Jenkins forecasting models are based on statistical concepts and principles and are able to model a wide spectrum of time series behavior. The underlying goal of this self- projecting time series forecasting method is to find an appropriate formula so that the residuals/errors are as small as possible and exhibit no pattern. The model- building process involves four steps. Repeated as necessary, to end up with a specific formula that replicates the patterns in the series as closely as possible and also produces accurate forecasts. (The term Arima and Box-Jenkin are use interchangeably)

It has a large class of models to choose from and a systematic approach for identifying the correct model form. There are both statistical tests for verifying model validity and statistical measures of forecast uncertainty. In contrast, traditional forecasting models offer a limited number of models relative to the complex behavior of many time series with little in the way of guidelines and statistical tests for verifying the validity of the selected model. (you learn this in Chapter 1)

**Basic Model:** With a stationary series in place, a basic model can now be identified. Three basic models exist, AR (autoregressive), MA (moving average) and a combined ARMA in addition to the previously specified RD (regular differencing) combine to provide the available tools. When regular differencing is applied together with AR and MA, they are referred to as ARIMA, with the I indicating "integrated" and referencing the differencing procedure.

ARIMA models are widely used in predicting stock prices, company sales, sunspot numbers, housing starts and many other fields. ARIMA models are also univariate, that is, they are based on a single time series variable. (There are multivariate models which are beyond the scope of this book and will not be discussed)

ARIMA processes appear, at first sight, to involve only one variable and its own history. Our intuition tells us that any economic variable is dependent on many other variables. How then can we account for the relative success of the Box Jenkins methodology. The use of univariate forecasts may be important for several reasons:

- In some cases we have a choice of modeling, say, the output of a large number of processes or of aggregate output, leaving the univariate model as the only feasible approach because of the sheer magnitude of the problem.

- It may be difficult to find variables which are related to the variable being forecast, leaving the univariate model as the only means for forecasting.
- Where multivariate methods are available the univariate method provides a yardstick against which the more sophisticated methods can be evaluated.
- The presence of large residuals in a univariate model may correspond to abnormal events—strikes etc.
- The study of univariate models can give useful information about trends long-term cycles, seasonal effects etc in the data.
- Some form of univariate analysis may be a necessary prerequisite to multivariate analysis if spurious regressions and related problems are to be avoided.

While univariate models perform well in the short term they are likely to be outperformed by multivariate methods at longer lead terms if variables related to the variable being forecast fluctuate in ways which are different to their past behavior.

Box and Jenkins have developed procedures for this multivariate modeling. However, in practice, even their univariate approach, sometimes, is not as well understood as the classic regression method. The objective of this book is to describe the basics of univariate Box- Jenkins models in simple and layman terms.

## The Mathematical Model

ARMA models can be described by a series of equations. The equations are somewhat simpler if the time series is first reduced to zero-mean by subtracting the sample mean. Therefore, we will work with the *mean-adjusted* series

$$\mathbf{y}(t) = \mathbf{y}(t) - \bar{Y} \quad (2.1)$$

Where  $y(t)$  is the original time series,  $\bar{Y}$  is its sample mean, and  $y(t)$  is the mean-adjusted series. One subset of ARMA models are the so-called *autoregressive*, or AR models. An AR model expresses a time series as a linear function of its past values. The *order* of the AR model tells how many lagged past values are included. The simplest AR model is the first-order autoregressive, or  $AR(1)$ , model

$$\mathbf{y}(t) = \mathbf{a}(1)*\mathbf{y}(t-1) + \mathbf{e}(t) \quad (2.2)$$

where  $y(t)$  is the mean-adjusted series in period  $t$ ,  $y(t-1)$  is the series in the previous period value,  $a(t)$  is the lag-1 autoregressive coefficient, and  $e(t)$  is the *noise*. The noise also goes by various other names: the *error*, the *random-shock*, and the *residual*. The residuals  $e(t)$  are assumed to be random in time (not autocorrelated), and normally distributed. We can see that the  $AR(1)$  model has the form of a regression model in which

$y(t)$  is regressed on its previous value. In this form,  $a(t)$  is analogous to the regression coefficient, and  $e(t)$  to the regression residuals. The name *autoregressive* refers to the regression on self (auto).

Higher-order autoregressive models include more lagged  $y(t)$  terms as predictors. For example, the second-order autoregressive model, AR(2), is given by

$$y(t) = a(1)*y(t-1) + a(2)*y(t-2) \quad (2.3)$$

where  $a(1)$ ,  $a(2)$  are the autoregressive coefficients on lags 1 and 2. The  $p^{th}$  order autoregressive model, AR( $p$ ) includes lagged terms on period  $t-1$  to  $t-p$ .

The *moving average* (MA) model is a form of ARMA model in which the time series is regarded as a moving average (unevenly weighted) of a random shock series  $e(t)$ . The *first-order moving average*, or MA(1), model is given by

$$y(t) = e(t) + c(1)*e(t-1) \quad (2.4)$$

where  $e(t)$ ,  $e(t-1)$  the residuals at period  $t$  and  $t-1$ , and  $c(1)$  is the first-order moving average coefficient. As with the AR models, higher-order MA models include higher lagged terms. For example, the second order moving average model, MA(2), is

$$y(t) = e(t) + c(1)*e(t-1) + c(2)*e(t-2) \quad (2.5)$$

The letter  $q$  is used for the order of the moving average model. The second-order moving average model is MA( $q$ ) with  $q = 2$ .

We have seen that the autoregressive model includes lagged terms on the time series itself, and that the moving average model includes lagged terms on the noise or residuals. By including both types of lagged terms, we arrive at what are called *autoregressive-moving-average*, or ARMA, models.

The order of the ARMA model is included in parentheses as ARMA( $p,q$ ), where  $p$  is the autoregressive order and  $q$  the moving-average order. The simplest, and most frequently used ARMA model is ARMA(1,1) model

$$y(t) = d + a(1)*y(t-1) + e(t) - c(1)*e(t-1) \quad (2.6)$$

The general *autoregressive moving average process* with AR order  $p$  and MA order  $q$  can be written as

$$y(t) = d + a(1)*y(t-1) + a(2)*y(t-2) + \dots + a(p)*y(t-p) - e(t) - c(1)*e(t-1) - c(2)*e(t-2) - \dots - c(p)*e(t-p) \quad (2.7)$$

The parameter  $d$  will be explained later in this Chapter.

## ARIMA MODELING

The purpose of ARIMA modeling is to establish a relationship between the present value of a time series and its past values so that forecasts can be made on the basis of the past values alone.

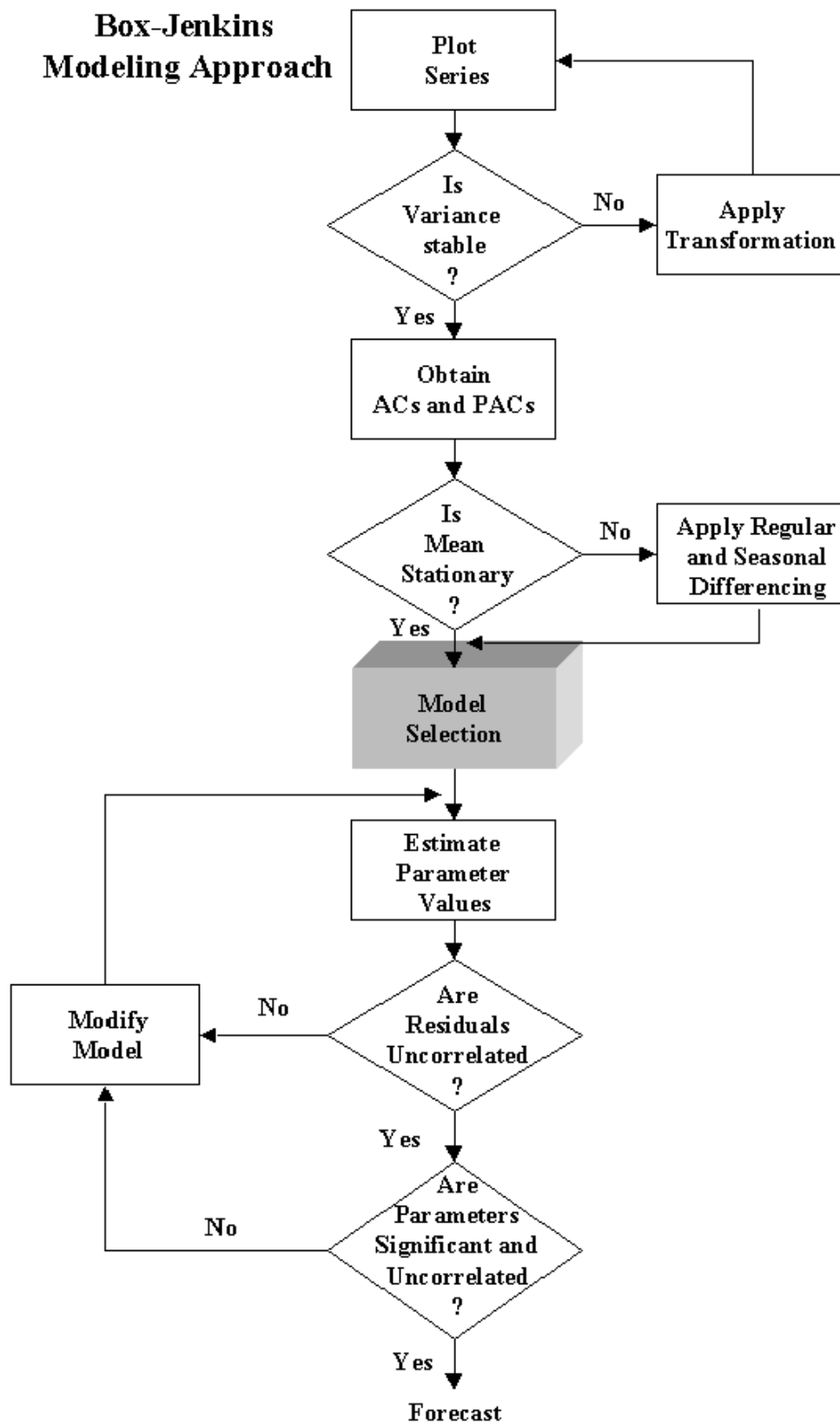
**Stationary Time Series:** The first requirement for ARIMA modeling is that the time series data to be modeled are either stationary or can be transformed into one. We can define that a stationary time series has a constant mean and has no trend overtime. A plot of the data is usually enough to see if the data are stationary. In practice, few time series can meet this condition, but as long as the data can be transformed into a stationary series, an Arima model can be developed. (I'll explain this concept in details later in this Chapter)

We emphasise again that, to forecast a time series using this approach to forecasting, we need to know whether the time series is stationary. If it is not, we need to make it stationary as otherwise the results will not make much sense. Not only this, in order to produce accurate and acceptable forecasts, we need to determine the class and the order of the model, i.e. whether it is an AR, MA or ARMA model and how many AR and MA coefficients (**p** and **q**) are appropriate. The analysis of the autocorrelation and partial autocorrelation functions provides clues to all of these questions. Both requirements above will be calculated and implemented in two Excel spreadsheet examples later.

The general steps for ARIMA modeling are shown in the chart below:



## Box-Jenkins Modeling Approach



## THE MODELING PROCESS

Box-Jenkins or Arima modeling of a stationary time series involves the following four major steps:

- A) Model identification
- B) Model estimation
- C) Diagnostic Checking
- D) Forecasting

The four steps are similar to those required for linear regression except that Step A is a little more involved. Box-Jenkins uses a statistical procedure to identify a model, which can be complicated. The other three steps are quite straightforward. Let's first discuss the mechanics of Step A, model identification, which we would do in great detail. Then we will use an example to illustrate the whole modeling process.

### A) MODEL IDENTIFICATION

ARIMA stands for Autoregressive- Integrated-Moving Average. The letter "I" (Integrated) indicates that the modeling time series has been transformed into a stationary time series. ARIMA represents three different types of models: It can be an AR (autoregressive) model, or a MA (moving average) model, or an ARMA which includes both AR and MA terms. Notice that we have dropped the "I" from ARIMA for simplicity.

Let's briefly define these three model forms again.

#### **AR Model:**

An AR model looks like a linear regression model except that in a regression model the dependent variable and its independent variables are different, whereas in an AR model the independent variables are simply the time-lagged values of the dependent variable, so it is autoregressive. An AR model can include different numbers of autoregressive terms. If an AR model includes only one autoregressive term, it is an AR ( 1 ) model; we can also have AR (2), AR (3), etc. An AR model can be linear or nonlinear. Below are a few examples:

#### **AR(1)**

$$y(t) = d + y(t-1) + e(t) \quad (2.8)$$

#### **AR(3)**

$$y(t) = d + a(1)*y(t-1) + a(2)*y(t-2) + a(3)*y(t-3) + e(t) \quad (2.9)$$

I will explain more on the **d** later.

#### **MA Model:**

A MA model is a weighted moving average of a fixed number of forecast errors produced in the past, so it is called moving average. Unlike the traditional moving average, the weights in a MA are not equal and do not sum up to 1. In a traditional moving average, the weight assigned to each of the  $n$  values to be averaged equals to  $1/n$ ; the  $n$  weights are equal and add up to 1. In a MA, the number of terms for the model and the weight for each term are statistically determined by the pattern of the data; the weights are not equal and do not add up to 1. Usually, in a MA the most recent value carries a larger weight than the more distant values,

For a stationary time series, one may use its mean or the immediate past value as a forecast for the next future period. Each forecast will produce a forecast error. If the errors so produced in the past exhibit any pattern, we can develop a MA model. Notice that these forecast errors are not observed values; they are generated values. All MA models, such as MA (1), MA (2), MA (3), are nonlinear. Below are a few examples:

**MA(1)**

$$y(t) = e(t) + c(1)*e(t-1) \quad (2.10)$$

**MA(2)**

$$y(t) = e(t) + c(1)*e(t-1) + c(2)*e(t-2) \quad (2.11)$$

**ARMA Model:**

An ARMA model requires both AR and MA terms. Given a stationary time series, we must first identify an appropriate model form. Is it an AR, or a MA or an ARMA? How many terms do we need in the identified model? To answer these questions we can use two methods

- 1) We can use a subjective way by calculating the autocorrelation function and the partial autocorrelation function of the series.
- 2) Or use objective methods of identifying the best ARMA model for the data at hand. (Automated Arima)

## Method 1

***i) What are Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF)?***

Understanding the ACF and PACF is very important in order to use method (1) to identify which model to use. Without going into the mathematics, ACF values fall between -1 and +1 calculated from the time series at different lags to measure the significance of correlations between the present observation and the past observations, and to determine how far back in time (i.e., of how many time-lags) are they correlated.

PACF values are the coefficients of a linear regression of the time series using its lagged values as independent variables. When the regression includes only one independent variable of one-period lag, the coefficient of the independent variable is called first order partial autocorrelation function; when a second term of two period lag is added to the regression, the coefficient of the second term is called the second order partial autocorrelation function, etc. The values of PACF will also fall between -1 and +1 if the time series is stationary.

Let's me show you how to calculate the ACF and PACF with an example. Open the workbook Arima in the Chapter 2 folder. Select *Worksheet (acf)*. This worksheet contains Dow Jones Industrial Composite Index (DJI) daily closing stock values between 20 July 2009 and 29 September 2009. All in all, this includes 51 daily values for the series.

### ACF

Below is the general formula for Autocorrelation (ACF):

$$\text{A sample autocorrelation is defined as } \hat{\rho}_k \equiv \frac{\hat{\gamma}_k}{\hat{\gamma}_0} = \frac{\text{cov}(R_{it}, R_{i,t-k})}{\text{var}(R_{it})}. \quad (2.12)$$

Don't be intimidated by this formula. It is easily implemented in a spreadsheet using Excel function. We can simplify this procedure by using some of the built in Excel formula. Formula above essentially tells us that the autocorrelation coefficient for some lag  $k$  is calculated as the covariance between the original series and the series removed  $k$  lags, divided by the variance of the original series.

Excel contains both the covariance and variance function, and they are: =VAR(range), and, =COVAR(range, range). Worksheet (acf) contains details how these two functions can be used to calculate autocorrelation coefficients.

	C	D	E
1	DJI	ACF	
2	9742.2	0.89359	=COVAR(\$C\$2:C51,C3:\$C\$52)/VAR(\$C\$2:\$C\$52)
3	9789.36	0.803544	=COVAR(\$C\$2:C50,C4:\$C\$52)/VAR(\$C\$2:\$C\$52)
4	9665.19	0.719049	=COVAR(\$C\$2:C49,C5:\$C\$52)/VAR(\$C\$2:\$C\$52)
5	9707.44	0.670069	=COVAR(\$C\$2:C48,C6:\$C\$52)/VAR(\$C\$2:\$C\$52)
6	9748.55	0.613015	=COVAR(\$C\$2:C47,C7:\$C\$52)/VAR(\$C\$2:\$C\$52)
7	9829.87	0.562698	=COVAR(\$C\$2:C46,C8:\$C\$52)/VAR(\$C\$2:\$C\$52)
8	9778.86	0.523415	=COVAR(\$C\$2:C45,C9:\$C\$52)/VAR(\$C\$2:\$C\$52)
9	9820.2	0.466678	
10	9783.92	0.433936	
11	9791.71	0.409159	
12	9683.41	0.421849	
13	9626.8	0.427594	
14	9605.41	0.426171	

Figure 2.0

From the formula (and we only show the first seven values and calculations) it is clear that the variance part is easy, i.e. just the range \$C\$2:\$C\$52 in our case. The covariance is just a bit more difficult to calculate. The ranges are:

\$C\$2:C51, C3:\$C\$52  
 \$C\$2:C50, C4:\$C\$52  
 \$C\$2:C49, C5:\$C\$52  
 \$C\$2:C48, C6:\$C\$52, etc.

This means that if we copy the cells downwards, C51 will become C52, then C53 etc. To avoid this problem, we can copy the formula down the column, but we need to manually change C51 onwards in a descending sequence. There you go. The ACF values are calculated in column D.

### PACF

The PACF plot is a plot of the *partial* correlation coefficients between the series and lags of itself. A partial *autocorrelation* is the amount of correlation between a variable and a lag of itself that is not explained by correlations at all *lower-order-lags*. The autocorrelation of a time series Y at lag 1 is the coefficient of correlation between Y(t) and Y(t-1), which is presumably also the correlation between Y(t-1) and Y(t-2). But if Y(t) is correlated with Y(t-1), and Y(t-1) is equally correlated with Y(t-2), then we should also expect to find correlation between Y(t) and Y(t-2). (In fact, the amount of correlation we should expect at lag 2 is precisely the *square* of the lag-1 correlation.) Thus, the correlation at lag 1 "propagates" to lag 2 and presumably to higher-order lags. The *partial* autocorrelation at lag 2 is therefore the difference between the actual correlation at lag 2 and the expected correlation due to the propagation of correlation at lag 1.

Select *Worksheet (pacf)*. This show how PACF is implemented and calculated in Excel. The PACF values are specify in Column C. The partial autocorrelation coefficients are defined as the last coefficient of a partial autoregression equation of order k. This is the general formula.

$$\pi_{\tau} = \frac{\rho_{\tau} - \sum_{j=1}^{\tau-1} \pi_{\tau-1,j} \cdot \rho_{\tau-j}}{1 - \sum_{j=1}^{\tau-1} \pi_{\tau-1,j} \cdot \rho_{\tau-j}}, \quad (2.13)$$

Where  $\tau > 1$ ,  $\rho$  is the autocorrelation,  $\pi$  is the PACF

The formula above is implemented on cell E4, F5, G6, H7, I8 and so on. (see Fig 2.1 below)

	A	B	C	D	E	F	G
1	<b>Lag</b>	<b>AC</b>	<b>PAC</b>				
2	k	$r_k$	$r_{k }$	$k,1$	$k,2$	$k,3$	$k,4$
3	1	0.8936	0.89	<b>0.89</b>			
4	2	0.8035	0.03	0.87	<b>0.03</b>		
5	3	0.719	-0.02	0.87	0.04	<b>-0.02</b>	
6	4	0.6701	0.13	0.87	0.03	-0.13	<b>0.13</b>

Figure 2.1

We can see that the PACF calculation is a bit difficult and complex. Fortunately for you I've written a macro to simplify the calculation. To use this macro, you need to load **nn\_Solve** into your Excel. I will show you the steps on how to do it with an example later. (see Appendix A on how to load **nn\_Solve**)

*ii) How do we use the pair of ACF and PACF functions to identify an appropriate model?*

A plot of the pair will provide us with a good indication of what type of model we want to entertain. The plot of a pair of ACF and PACF is called a correlogram. Figure 2.2 shows three pairs of theoretical ACF and PACF correlograms.

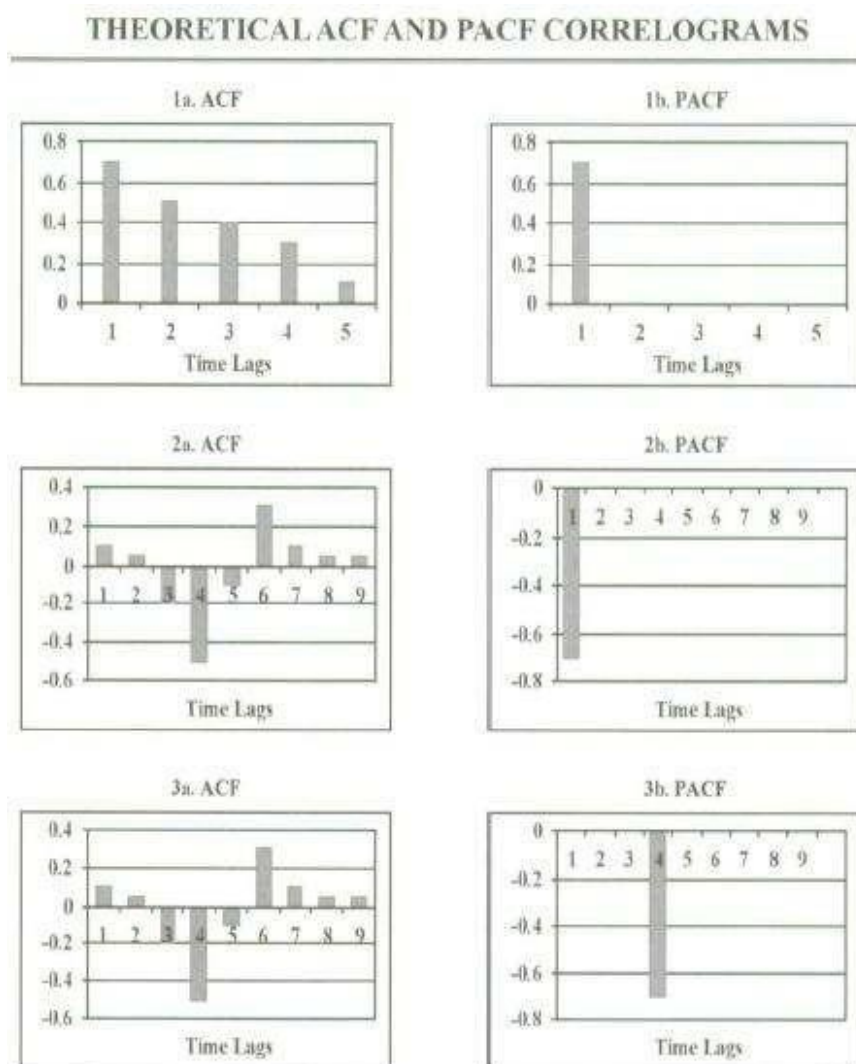


Figure 2.2

In modeling, if the actual correlogram looks like one of these three theoretical correlograms, in which the ACF diminishes quickly and the PACF has only one large spike, we will choose an AR (1) model for the data. The "1" in parenthesis indicates that

the AR model needs only one autoregressive term, and the model is an AR of order 1. Notice that the ACF patterns in 2a and 3a are the same, but the large PACF spike in 2b occurs at lag 1, whereas in 3b, it occurs at lag 4. Although both correlograms suggest an AR (1) model for the data, the 2a and 2b pattern indicates that the one autoregressive term in the model is of lag 1; but the 3a and 3b pattern indicates that the one autoregressive term in the model is of lag 4.

Suppose that in Figure 2.2, ACF and PACF exchange their patterns, that is, the patterns of PACF look like those of the ACF and the patterns of ACF look like the PACF having only one large spike, then we will choose a MA (I) model. Suppose again that the PACF in each pair looks the same as the ACF, and then we will try an ARMA(1, 1).

So far we have described the simplest AR, MA, and ARMA models. Models of higher order can be so identified, of course, with different patterns of correlograms.

Although the above catalogue is not exhaustive, it gives us a reasonable idea of what to expect when deciding about the most basic models. Unfortunately the above behavioural catalogue of autocorrelation and partial autocorrelation functions is only theoretical. In practise, the actual autocorrelations and partial autocorrelations only vaguely follow these patterns, which is what makes this subjective approach to forecasting very difficult. In addition to this, the real-life time series can be treated as just a sample of the underlying process. Therefore, the autocorrelations and partial autocorrelations that are calculated are just estimates of the actual values, subject to sampling error.

The autocorrelations and partial autocorrelations also play a prominent role in deciding whether a time series is stationary, to what class of models it belongs and how many coefficients it is characterised by. The question that is still open is how to calculate the coefficients **a** and **c** that constitute a particular model.

Before we proceed with how to estimate **a** and **c**, we shall return to the question of differencing and stationarity as promised earlier. In general we must be cautious where differencing is concerned, which will influence the class of the model. It would be wrong to assume that when unsure as to whether the series is nonstationary, it should simply be differenced. Over differencing can lead us to believe that the time series belongs to a completely different class, which is just one of the problems.

### **Rules For Differencing**

How do we, then, know if we have exaggerated and overdifferenced the series? One of the basic rules is: if the first autocorrelation of the differenced series is negative and more than  $-0.5$ , the series has probably been overdifferenced. Another basic rule is: if the variance for the higher level of differencing increases, we should return to the previous level of differencing. One of the rules of thumb is that the level of differencing corresponds to the degree of a polynomial trend that can be used to fit the actual time series.

The whole notion of differencing is closely related to the concept of so-called unit roots. Unit root means that an AR(1) or a MA(1) coefficient is equal to one (unity). For higher order models, this means that the sum of all coefficients is equal to one. If this happens we

have a problem. If an AR(1) model has a unit root, then this AR coefficient should be eliminated and the level of differencing should be increased. For higher AR(p) models, the number of AR coefficients has to be reduced and the level of differencing increased. For MA models showing unit roots, an MA coefficient should also be removed, but the level of differencing has to be decreased. Sometimes we do not “catch” unit roots early enough, and produce forecasts, which turn out to be very erratic. This is also a consequence of unit roots, which means that a reduction in AR or MA coefficients is necessary.

Another question we need to answer is: what is the meaning of **d**, how do we calculate it and when do we include it in a model?

Essentially, **d** in ARMA models plays the same role as the intercept in linear regression. Our model here is called an ARMA model with a level, where **d** represents this initial level of the model (an intercept). Sometimes it is also referred to as the trend parameter, or a constant.

If we want to calculate this trend parameter, we need to start with the formula for the expected value of an AR process, i.e. the mean value. The mean of any AR(p) process is calculated as:

$$Z = d / (1 - a(1) - \dots - a(p)) \quad (2.17)$$

Which, for AR(2) yields:

$$Z = d / (1 - a(1) - a(2)) \quad (2.18)$$

From this formula, the level d (or the trend component) for the AR(2) process is calculated as:

$$d = z * (1 - a(1) - a(2)) \quad (2.19)$$

In general, the level for any AR(p) process is calculated as:

$$d = z * \left(1 - \sum_{i=1}^p a(p)\right) \quad (2.20)$$

Now we know what it is and how to calculate it, the open part of the question is still: when do we include it in our model?

The set of rules can be summarised as follows:

- If a time series is non-stationary in its original form and we have had to difference it to make it stationary, then the constant is usually **not** needed
- Time series differenced more than twice do **not** need a constant
- If the original time series is stationary with zero mean, a constant is **not** needed.



- If the original series is stationary, but with a significantly large mean (which effectively means  $\bar{x} \pm \sigma_x > 1$ ), the constant is necessary.
  - If the model does not have an AR component (i.e. it is an MA or IMA model), then **the constant is equal to the mean value** of the series.
  - If the model has an AR component, the constant is calculated as in (2.20)

(I will show you another example where we will calculate the constant **d** later)

### Testing for zero mean to indicate stationarity

What happens if one level of differencing is not enough and the second level is too much? This sometimes happens in practise and a time series appears to be stationary, yet its mean value is not zero, despite the stationarity requirement that it should. If this happens, we have to ensure that the mean is at least close to zero. The easiest way to do this is to calculate the mean  $\bar{w}$  (the average) of the differenced series  $w_t$ , and subtract it from every observation: (Goto *Worksheet (Daily Sales)* ). The data from A2:A101 is the daily sales from a sport shop in thousand.

$$z_t = w_t - \bar{w} \quad \text{implemented in column B}$$

Once we have transformed the differenced time series in such a way, we can calculate this transformed series' mean value,  $\bar{z}$  in cell E3 and check whether it is zero. How do we check whether the mean is zero or close to zero? First we need to estimate this transformed series' standard error. You will remember that SE is a ratio between the standard deviation and the square root of the number of observations:

$$SE(z) = \frac{\sigma_z}{\sqrt{n}} \quad (2.14)$$

The transformed time series' mean,  $\bar{z}$ , is considered nonzero if:

$$|\bar{z}| < 1.96 SE(z) \quad (2.15) \quad \text{enter in cell E5}$$

Don't worry about the math symbols above. They can be easily implanted in an Excel spreadsheet. (see Figure 2.3 below). The result is non-zero (see cell E5) and from the chart we can see that the time series appears nonstationary i.e. it's trending up. So we need to pre-process the time series which is differencing. A one lag differencing i.e.  $y(t) = y(t) - y(t-1)$ . is applied. The values in C2:C100 are the 1 lag differencing value.

	D	E	F
1			
2	<b>Mean Original</b>	7.146824125	=AVERAGE(A2:A101)
3	<b>Mean Transformed</b>	-4.53859E-15	=AVERAGE(B2:B101)
4	<b>1.96*SE</b>	0.0088899266453850	=1.96*(SQRT(B2:B101)/COUNT(B2:B101))
5	<b>Zero Mean Test</b>	Non-zero	=IF(E3<E4,"Non-zero","Zero")

Figure 2.3

There is another common approach to transformations, which avoids differencing. In finance, for example, we are often more interested in returns, i.e. if we sell shares today ( $y_t$ ), how much we have earned when compared with when we bought them ( $y_{t-1}$ ). Mathematically this is simply:  $(y_t - y_{t-1})/y_{t-1}$ . Even if the shares values are jumping wildly, the series of such calculated returns will usually be stationary. The above mathematical expression is known to be approximately equal to  $\log(y_t) - \log(y_{t-1})$ , which is often used for calculating returns. This expression can also be used to transform a time series into a stationary form. Some stationary series are not strictly stationary and although they have a constant mean, their variance is not constant (remember the idea of homoscedasticity?). The log transformation suggested here is known to reduce heteroscedasticity.

After a stationary series is in place, a basic model can now be identified. Three basic models exist, AR (autoregressive), MA (moving average) and a combined ARMA in addition to the previously specified RD (regular differencing) combine to provide the available tools. When regular differencing is applied together with AR and MA, they are referred to as ARIMA, with the I indicating "integrated" and referencing the differencing procedure.

Bear in mind that we are using method (1) to identify the model. So far I have 3 components that are important for us to understand in order to identify the model.

- The ACF and PACF
- Data stationary
- Differencing

Let's use a spreadsheet example to show how to calculate the ACF and PACF first and then to demonstrate what we have just discussed i.e using ACF and PACF to determine the **p** and **q** parameters as in ARMA(p,q)

We copy the one lag differenced values in C2:C100 and paste them to range C2:C100 in *Worksheet (Daily Sales(1))*. From the chart below it now look stationary and random. And the test also indicates that the time series has zero mean in cell L4

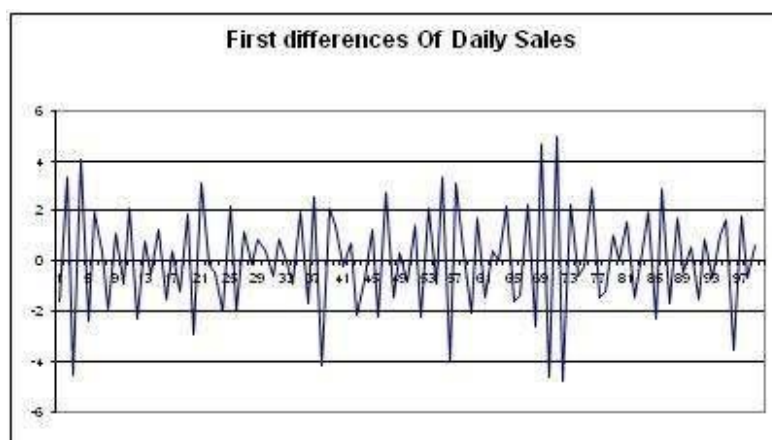


Figure 2.4

Now we need to calculate the ACF and PACF. Although, I've show you how to calculate them manually (see worksheet(acf) and worksheet (pacf), it is still very troublesome, especially when you calculate PACF. Fortunately, you can use the **nn\_Solve** addin written by me to calculate the ACF and PACF automatically. Load **nn\_Solve** to your Excel. (see Appendix on how to load **nn\_Solve**)

- 1) Select *Arima* on the **nn\_Solve** menu (see Figure 2.4a)

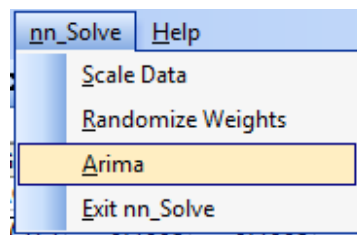


Figure 2.4a

Enter the reference that you want to calculate in the **Data Range**. In our case, we enter C2:C100. (see Figure 2.4b below). **The data range cannot start with row 1 like C1, A1, B1 and so on. Else nn\_Solve will give you an error. Always enter the data that you want to calculate starting from row 2 like C2, A2, B2 and so on in a spreadsheet.**

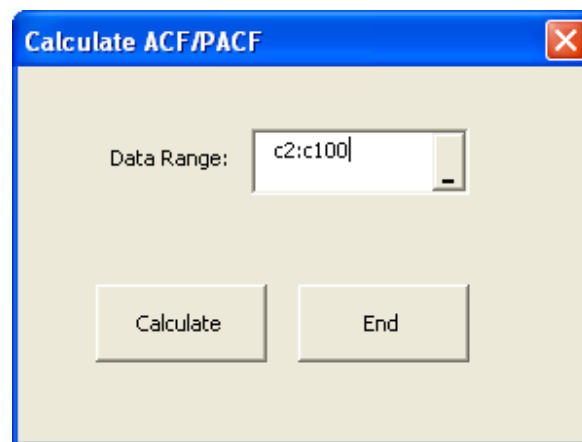


Figure 2.4b

- 2) Then click on the **Calculate** button. The ACF, PACF and the Standard Error will be calculated. (see Figure 2.4c)

	C	D	E	F	G	H	I
1		AC	SE+	SE-	PAC	SE+	SE-
2	-1.63509	-0.77171	0.148769	-0.14877	-0.77171	0.1005	-0.1005
3	3.386727	0.480725	0.163709	-0.16371	-0.28388	0.100504	-0.1005
4	-4.52892	-0.28475	0.168638	-0.16864	-0.07402	0.100504	-0.1005
5	4.051568	0.086718	0.169088	-0.16909	-0.23466	0.100504	-0.1005
6	-2.3803	0.029377	0.169139	-0.16914	-0.10969	0.100504	-0.1005
7	1.982004	-0.07349	0.169462	-0.16946	-0.06177	0.100504	-0.1005
8	0.334183	0.107416	0.170148	-0.17015	0.017955	0.100504	-0.1005
9	-1.89037	-0.03725	0.17023	-0.17023	0.205491	0.100504	-0.1005
10	1.106677	0.02557	0.170269	-0.17027	0.121509	0.100504	-0.1005

Figure 2.4c

Build the charts below using the data calculated. (see Fig. 2.5 and Fig 2.6).The autocorrelation and the partial autocorrelation functions for the differenced sales revenue data are given in Fig. 2.5 and Fig 2.6

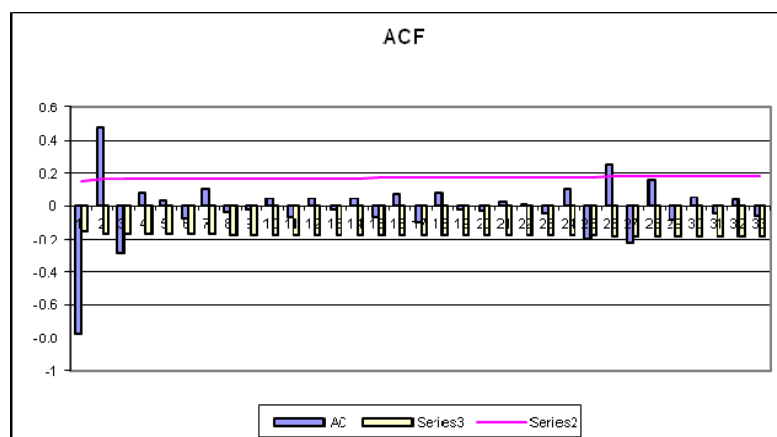


Figure 2.5

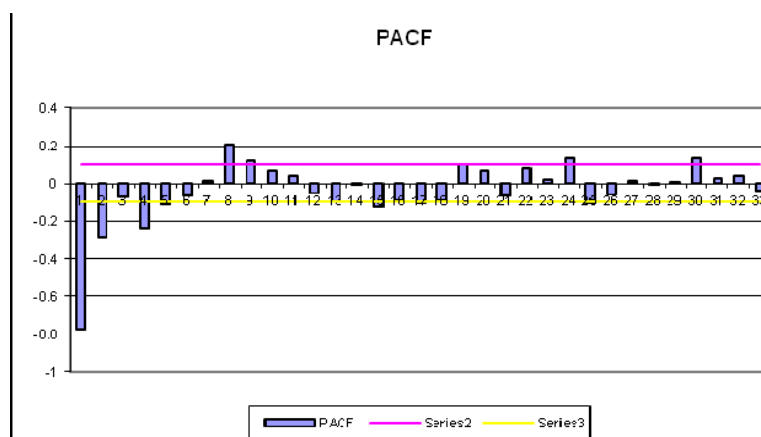


Figure 2.6

The partial autocorrelation function in Fig. 2.6 shows two coefficients as significantly non-zero, implying that this is an ARMA(p,q) model. The autocorrelation function confirms this assumption as it shows the pattern usually associated with an ARMA(p,q) model. Given that

we had to difference the original time series, the model we will use, therefore, is ARIMA(2,1,1) model or ARMA(2,1).

## B) MODEL ESTIMATION

The equation for this model is:

$$\mathbf{y}(t) = \mathbf{d} + \mathbf{a}(1)*\mathbf{y}(t-1) + \mathbf{a}(2)*\mathbf{y}(t-2) - \mathbf{e}(t) - \mathbf{c}(1)*\mathbf{e}(t-1) \quad (2.16)$$

Let's implement this formula in a spreadsheet to optimise the coefficients, fit the model and produce the forecasts.. Open worksheet (*Daily Sales(2)*). The 1 lag difference sales figures are enter in column A. In column B is the residual. Column C is the full formula. Press Ctrl + ~ to view the formula in your Excel sheet. (see Figure 2.7 below)

X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

[illegible]

	D	E
1		<b>Initial</b>
2	a(1)	0.1
3	a(2)	0.1
4	c(1)	0.1
5	Mean	0.05
6	St. Dev.	2.08
7	Measure	2.13
8	d	0
9	Implied $\mu$	0.0000
10	SSE	377.07
11		
12	$-1 < a(2) < 1$	0.1
13	$a(1) + a(2) < 1$	0.2
14	$a(2) - a(1) < 1$	0
15	$-1 < c(1) < 1$	0.1
16		
17	$\bar{e}$	0.02375
18	SE <sub>e</sub>	0.199149636
19	Value	0.390333286
20	Verdict:	<b>Zero mean</b>
21		
22		
23		<b>Durbin - Watson Test</b>
24		1292.5951
25		377.0698
26		3.42800

Figure 2.8

The initial values of  $a(1)$ ,  $a(2)$  and  $c(1)$  are set in cells E2, E3 and E4. Cells E5 and E6 contain the time series mean and standard deviation. Since we have applied differencing to the time series, the value  $d$  is not needed and enters as 0 in cell E8. I will show you another example where we will calculate the  $d$  later when we use method (2).

Our sales revenue data set was originally nonstationary and had to be differenced before modelling could be applied. This is the reason for omitting the constant  $d$  in the first place. So we set  $d$  as 0 in this example. (See Fig 2.8 above)

From the formula 2.16 we can easily extract  $e(t)$ , which is:

$$e(t) = y(t) - d + a(1)*y(t-1) + a(2)*y(t-2) - c(1)*e(t-1)$$

The above formula implies that to calculate  $e(1)$ . But we need to know  $e(0)$ , which we do not. The convention is to assign zeros to all the unknown values of  $e(0)$ . In Fig. 2.7 above, we can see zero in cell B2 and B3, which is the first cell needed to perform this calculation. Since the model is a ARMA(2,1), we also assign a 0 to B3.

Now we have all the errors  $e(t)$ , given just the initial values of  $a(1)$ ,  $a(2)$  and  $c(1)$ , we can calculate the so-called conditional sum of squares of residual (SSE), which is conditional on the values of  $a(1)$  and  $c(1)$ . The formula for SSE is:

$$SSE(a,c) = \sum_{t=1}^n e(t)$$

Cell E10 gives us the value of  $SSE = 377.07$  initially, which was obtained using Excel function  $=SUMSQ(B2:B100)$ . This cell is instrumental for estimating the optimum value of  $a(1)$ ,  $a(2)$  and  $c(1)$ , which will hopefully lead towards best possible forecast. To achieve this, we will use Excel Solver. Our objective is to minimise SSE (i.e. the value in cell E10), by changing values of E2:E4, i.e. the values of  $a(1)$ ,  $a(2)$  and  $c(1)$ . As before, we need to define the admissible region which will guarantee that our model is stationary and invertible. For ARIMA(2,1,1) processes, this is:  $-1 < a(1) < 1$  and  $-1 < c(1) < 1$ , or,  $|a(1)| < 1$  and  $|c(1)| < 1$ . Cells E12 to E15 define these conditions.

Before we show how to use Solver, we need to understand one more point about AR(p) coefficients  $a(1)$ ,  $a(2)$ , etc. A process that is generated using these coefficients has to be stationary. In other words, certain values of  $a(1)$ ,  $a(2)$ , etc., will not necessarily generate a stationary process. To satisfy this strict condition of stationarity, we need to define the admissible region for these coefficients. In case of AR(1), this admissible region is defined as:  $-1 < a(1) < 1$  (or,  $|a(1)| < 1$ ). In case of AR(2), this admissible region is defined by three conditions:

$$a(2) + a(1) < 1, \quad a(2) - a(1) < 1 \quad \text{and} \quad -1 < a(2) < 1 \quad (\text{or, } |a(2)| < 1) \quad \text{and} \quad -1 < c(1) < 1$$

We can see that our initial estimates of  $a(1)$ ,  $a(2)$ ,  $c(1)$  in Fig. 2.8 satisfy all these stationarity conditions. These parameters are entered in cell E12 to E15. One last thing before I show how to use Solver to calculate the coefficients.

Now we understand modelling (at least for this class of models), we must establish whether the estimated values of the model coefficients are truly the best ones available. Traditionally this question involves very complex and complicated calculations, which ensure that the maximum likelihood estimators are selected. Fortunately with help from Excel Solver, many of these operations are not necessary. Let's do it...

Our objective is to minimize the SSE value in cell E10.

Cell E10 gives us the value of  $SSE = 377.07$  initially, which was obtained using Excel function  $=SUMSQ(B2:B100)$ .

This cell, together with cells E12 to E15 is instrumental for estimating the optimum value of  $a(1)$ ,  $a(2)$  and  $c(1)$ , which will hopefully lead towards best possible forecast. To achieve this, we will use Excel Solver. Our objective is to minimise SSE (i.e. the value in cell E10), by changing values of E2:E4, i.e. the values of  $a(1)$ ,  $a(2)$  and  $c(1)$ . As before, we need to define the admissible region which will guarantee that our model is stationary and invertible. For ARIMA(2,1,1) processes, this is:  $-1 < a(1) < 1$  and  $-1 < c(1) < 1$ , or,  $|a(1)| < 1$  and  $|c(1)| < 1$ . Cells E12 to E15 define these conditions.

Upon invoking Solver from the Tools menu, a dialogue box appears as in Fig 2.9 below

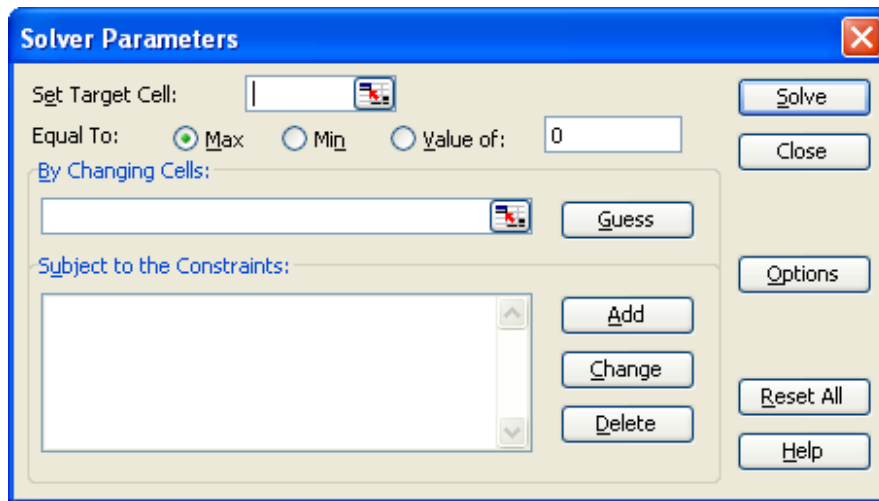


Figure 2.9

Next we will enter all the parameters in this dialogue box. It will look like this after all the parameters are entered. (see Fig 2.10)

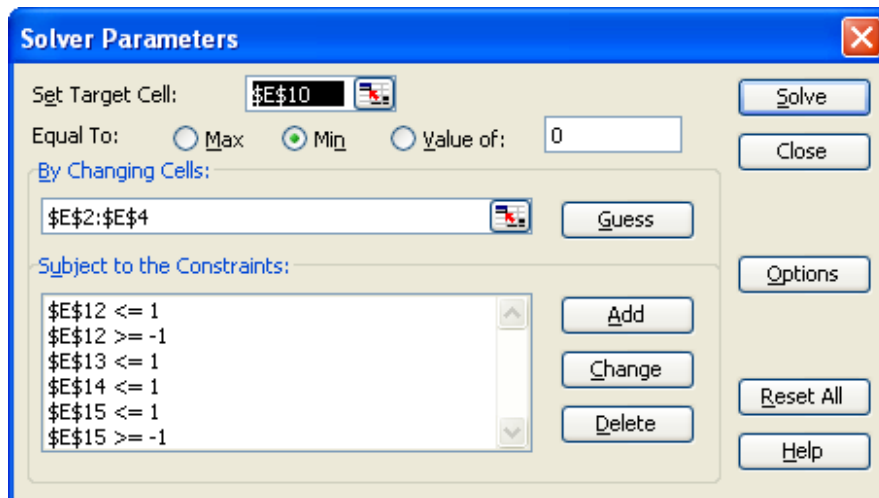


Figure 2.10

- a) Set Target Cell : E10 (the SSE)
- b) By Changing Cells : E2:E4 (a(1), a(2), c(1))
- c) The constraint as shown in cell E12:E15



	D	E
1		<b>Initial</b>
2	a(1)	0.1
3	a(2)	0.1
4	c(1)	0.1
5	Mean	0.05
6	St. Dev.	2.08
7	Measure	2.13
8	d	0
9	Implied $\mu$	0.0000
10	SSE	377.07

Figure 2.11

Before Optimization

- d) Click the Solve button. Solver will start to optimize.

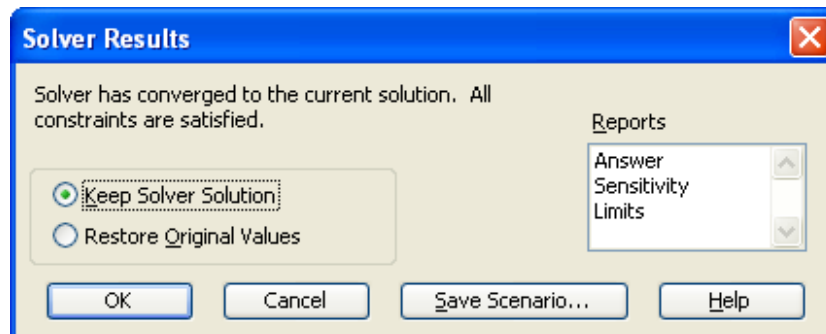


Figure 2.12

- e) And Keep the Solver solution

	D	E	F
1		<b>Final</b>	<b>Initial</b>
2	a(1)	-0.537871274	0.1
3	a(2)	0.058098633	0.1
4	c(1)	0.614100745	0.1
5	Mean	0.05	
6	St. Dev.	2.08	
7	Measure	2.13	
8	d	0	
9	Implied $\mu$	0.0000	
10	SSE	137.09	377.07
11			
12	$-1 < a(2) < 1$	0.058098633	
13	$a(1) + a(2) < 1$	-0.479772641	
14	$a(2) - a(1) < 1$	0.595969908	
15	$-1 < c(1) < 1$	0.614100745	
16			
17	$\bar{e}$	0.16058	
18	SE <sub>e</sub>	0.118952502	
19	Value	0.233146903	
20	Verdict:	<b>Zero mean</b>	
21			
22			
23		<b>Durbin - Watson Test</b>	
24		261.0780	=SUMXMY2(B3:B100,B2:B99)
25		137.0857	=SUMSQ(B2:B100)
26		1.90449	=E24/E25

Figure 2.13  
After optimization

The solution is instantly found and the values appear as you can see in Fig 2.13 above

As we can see a(1) is now -0.537871274, a(2) is 0.058098633 and c(1) becomes 0.614100745, which gives a much lower value of SSE = 137.09, compared with the previous value of 377.07

As we have calculated the e(t), implicitly in column B, if we really wanted it, we can explicitly calculate the values of the fitted time series, i.e. the ex-post forecasts y(t) in accordance with this model. We use the formula:

$$y(t) = d + (-0.537871274 * y(t-1)) + 0.058098633 * y(t-2) - 0.614100745 * e(t-1) \quad (2.18)$$

instead of

$$y(t) = d + a(1) * y(t-1) + a(2) * y(t-2) - e(t) - c(1) * e(t-1)$$

You can see that I have drop the e(t) from our formula as we have calculated them in column B to derive the values in Column C. Column C in Fig. 2.15 shows the values for y(t) and Fig. 2.14 shows the formula used to produce Fig. 2.15

	C	D	E
1	$y(t) = d + a(1)y(t-1) + a(2)y(t-2) - c(1)e(t-1)$		Final
2	=E8	a(1)	-0.537871274212718
3	=E8	a(2)	0.05809863343173
4	=\$E\$8+(\$E\$2*A3+\$E\$3*A2)-\$E\$4*B3	c(1)	0.614100745321989
5	=\$E\$8+(\$E\$2*A4+\$E\$3*A3)-\$E\$4*B4	Mean	=AVERAGE(A2:A100)
6	=\$E\$8+(\$E\$2*A5+\$E\$3*A4)-\$E\$4*B5	St. Dev.	=STDEV(A2:A100)
7	=\$E\$8+(\$E\$2*A6+\$E\$3*A5)-\$E\$4*B6	Measure	=E5+E6
8	=\$E\$8+(\$E\$2*A7+\$E\$3*A6)-\$E\$4*B7	d	0
9	=\$E\$8+(\$E\$2*A8+\$E\$3*A7)-\$E\$4*B8	Implied $\mu$	=E8/(1-E2)
10	=\$E\$8+(\$E\$2*A9+\$E\$3*A8)-\$E\$4*B9	SSE	=SUMSQ(B2:B100)
11	=\$E\$8+(\$E\$2*A10+\$E\$3*A9)-\$E\$4*B10		
12	=\$E\$8+(\$E\$2*A11+\$E\$3*A10)-\$E\$4*B11	-1<a(2)<1	=E3
13	=\$E\$8+(\$E\$2*A12+\$E\$3*A11)-\$E\$4*B12	a(1)+a(2)<1	=E2+E3
14	=\$E\$8+(\$E\$2*A13+\$E\$3*A12)-\$E\$4*B13	a(2)-a(1)<1	=E3-E2
15	=\$E\$8+(\$E\$2*A14+\$E\$3*A13)-\$E\$4*B14	-1<c(1)<1	=E4
16	=\$E\$8+(\$E\$2*A15+\$E\$3*A14)-\$E\$4*B15		
17	=\$E\$8+(\$E\$2*A16+\$E\$3*A15)-\$E\$4*B16	$\bar{e}$	=AVERAGE(B2:B100)
18	=\$E\$8+(\$E\$2*A17+\$E\$3*A16)-\$E\$4*B17	$SE_e$	=STDEV(B3:B100)/SQRT(COUNT(B3:B100))
19	=\$E\$8+(\$E\$2*A18+\$E\$3*A17)-\$E\$4*B18	Value	=1.96*E18
20	=\$E\$8+(\$E\$2*A19+\$E\$3*A18)-\$E\$4*B19	Verdict:	=IF(E17>E19,"Nonzero mean","Zero mean")
21	=\$E\$8+(\$E\$2*A20+\$E\$3*A19)-\$E\$4*B20		
22	=\$E\$8+(\$E\$2*A21+\$E\$3*A20)-\$E\$4*B21		
23	=\$E\$8+(\$E\$2*A22+\$E\$3*A21)-\$E\$4*B22		
24	=\$E\$8+(\$E\$2*A23+\$E\$3*A22)-\$E\$4*B23		Durbin - Watson Test
25	=\$E\$8+(\$E\$2*A24+\$E\$3*A23)-\$E\$4*B24		=SUMXMY2(B3:B100,B2:B99)
26	=\$E\$8+(\$E\$2*A25+\$E\$3*A24)-\$E\$4*B25		=SUMSQ(B2:B100)
27	=\$E\$8+(\$E\$2*A26+\$E\$3*A25)-\$E\$4*B26		=E24/E25

Figure 2.14

	A	B	C
1	$y(t)$	$e(t)$	$y(t) = d + a(1)y(t-1) + a(2)y(t-2) - c(1)e(t-1)$
2	-1.635	0.00	0.00
3	3.3867	0.00	0.00
4	-4.529	-2.61	-1.92
5	4.0516	-0.19	4.24
6	-2.38	-0.05	-2.33
7	1.982	0.43	1.55
8	0.3342	1.81	-1.47
9	-1.89	-0.72	-1.17
10	1.1066	-0.37	1.48

Figure 2.15

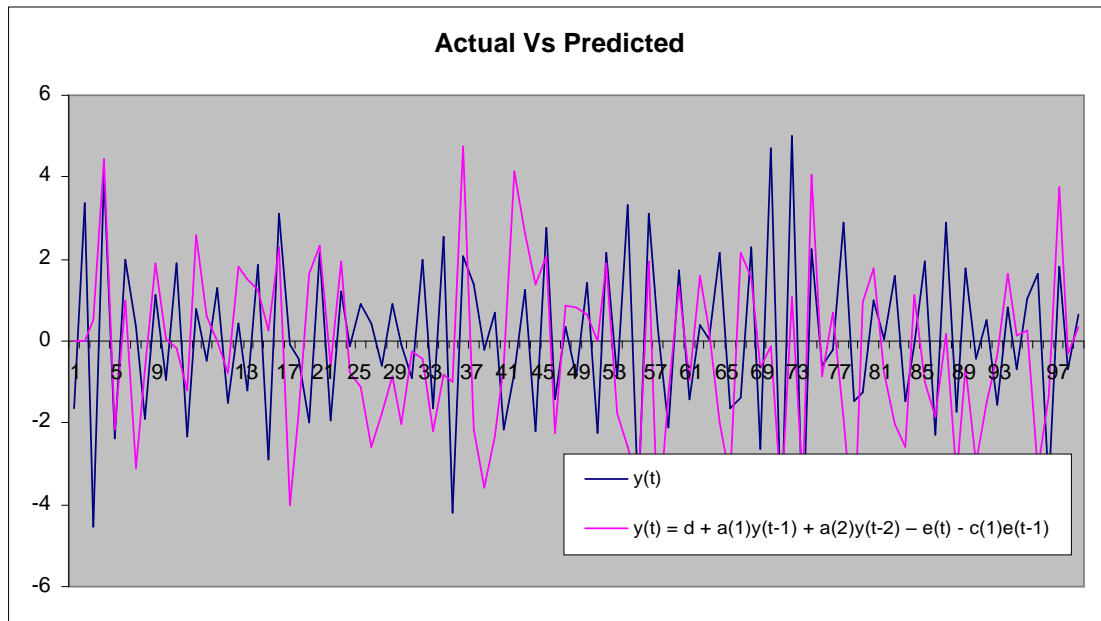


Figure 2.16

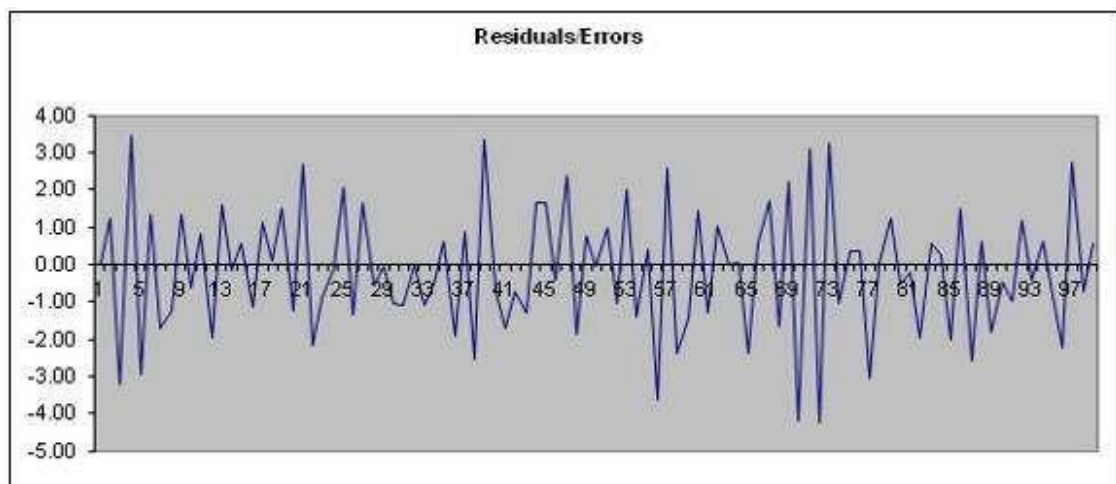


Figure 2.17

How closely the fitted values are matching the original time series can be seen in Fig.2.16 above. Forecasting errors from the column B are displayed in Fig. 2.17 above and they seem to be randomly distributed, as expected. Before we are ready to forecast, we need to do a diagnostic checking first.

### C) DIAGNOSTIC CHECKING:

How do we know that we have produced a reasonable model and that our model indeed reflects the actual time series? This is a part of the process that Box and Jenkins refer to as diagnostic checking. I will use two methods to conduct the diagnostic.

As we expect the forecasting errors to be completely random, the first step is to plot them, as we did in Fig 2.17 above for example. One of the requirements is that the residual mean should be zero, or close to zero. To establish that this is the case, we need to estimate the standard error of the mean error. This is calculated as:

$$\sigma_e = \sqrt{\frac{\sum_{t=1}^n (e_t - \bar{e})^2}{n}} \quad (2.19)$$

$$SE_e = \frac{\sigma_e}{\sqrt{n}} \quad (2.20) \text{ in cell E18}$$

Where  $\sigma_e$  is the residual standard deviation,  $\bar{e}$  is the mean error,  $n$  is the number of errors and  $SE_{\bar{e}}$  is the standard error of the mean error. If the residual mean  $\bar{e}$  is greater than 1.96 standard errors, then we can say that it is significantly non-zero:

$$\bar{e} > 1.96SE_{\bar{e}} \quad (2.21) \quad \text{in cell E20}$$

We can take an example from Column B for which the errors  $e(t)$  are calculated and shown in . How to estimate the standard residual error  $SE_e$  (standard error) is shown below in Fig 2.18 and the formula are given in Fig. 2.19 below

[illegible]

This part of the book is not available for viewing

This part of the book is not available for viewing

Cell E20 contains a brief IF statement evaluating whether the calculated mean value from E17 is greater than the standard error times 1.96. In our model we have zero mean which pass the test.

Another test that is quite popular is the Durbin-Watson test, which is use in the context of checking the validity of ARIMA models.

The **Durbin-Watson statistic** is a test statistic used to detect the presence of autocorrelation in the residuals from a regression analysis. It is named after James Durbin and Geoffrey Watson. If  $e_t$  is the residual associated with the observation at time  $t$ , then the test statistic is

$$w = \frac{\sum_{t=2}^T (e_t - e_{t-1})^2}{\sum_{t=1}^T e_t^2}. \quad (2.22)$$

Since  $w$  in cell E26 is approximately equal to  $2(1-r)$ , where  $r$  is the sample autocorrelation of the residuals,  $w = 2$  indicates no autocorrelation. The value of  $w$  always lies between 0 and 4. If the Durbin-Watson statistic is substantially less than 2, there is evidence of positive serial correlation. As a rough rule of thumb, if Durbin-Watson is less than 1.0, there may be cause for alarm. Small values of  $w$  indicate successive error terms are, on average, close in value to one another, or positively correlated. If  $w > 2$  successive error terms are, on average, much different in value to one another, i.e., negatively correlated. In regressions, this can imply an underestimation of the level of statistical significance.

	E	F
22		
23	<b>Durbin - Watson Test</b>	
24	261.0782	=SUMXMY2(B3:B100,B2:B99)
25	137.0857	=SUMSQ(B2:B100)
26	1.90449	=E24/E25
27		

Figure 2.20

In our model we have 1.90449 in cell E26 which is very close to 2 which indicates no autocorrelation. See Fig 2.20 above. We can now proceed with the forecast.

#### D) FORECAST

Now we are ready to produce real forecasts, i.e. those that go into the future. The equation can be applied “one step ahead” to get estimate  $y(t)$  from observed  $y(t-1)$ . A “ $k$ -step-ahead” prediction can also be made by recursive application of equation. In recursive application, the observed  $y$  at time 1 is used to generate the estimated  $y$  at time 2. That estimate is then substituted as  $y(t-1)$  to get the estimated  $y$  at time 3, and so on. The  $k$ -step-ahead predictions eventually converge to zero as the prediction horizon,  $k$ , increases. Goto cell A101:A105.

We will forecast as per the formula below: Arima(2,1,1) or ARMA(2,1). The formula is

$$y(t) = -0.537871274*y(t-1) + 0.058098633*y(t-2) - 0.614100745*e(t-1)$$

	A	B	C
98	1.8	-0.97	3.75
99	-0.69	-0.10	-0.31
100	0.66	0.12	0.35
101	-0.58		-0.46
102	0.32		0.29
103	-0.22		-0.18
104	0.39		0.11
105	0.04		-0.07

Figure 2.21

	A	B	C
98	1.802826736	=A98-(\$E\$8+(\$E\$2*A97+\$E\$3*A96)-\$E\$4*B97)	=\$E\$8+(\$E\$2*A97+\$E\$3*A98)-B98-\$E\$4*B97
99	-0.686138281	=A99-(\$E\$8+(\$E\$2*A98+\$E\$3*A97)-\$E\$4*B98)	=\$E\$8+(\$E\$2*A98+\$E\$3*A99)-B99-\$E\$4*B98
100	0.656560483	=A100-(\$E\$8+(\$E\$2*A99+\$E\$3*A98)-\$E\$4*B99)	=\$E\$8+(\$E\$2*A99+\$E\$3*A100)-B100-\$E\$4*B99
101	-0.58		=\$E\$8+(\$E\$2*A100+\$E\$3*A101)-\$E\$4*B100
102	0.32		=\$E\$8+(\$E\$2*C101+\$E\$3*A100)
103	-0.22		=\$E\$8+(\$E\$2*C102+\$E\$3*C101)
104	0.388682141		=\$E\$8+(\$E\$2*C103+\$E\$3*C102)
105	0.037989518		=\$E\$8+(\$E\$2*C104+\$E\$3*C103)

Figure 2.22

Fig. 2.21 shows the spreadsheet containing basic numbers and Fig 2.22 shows all the calculations.

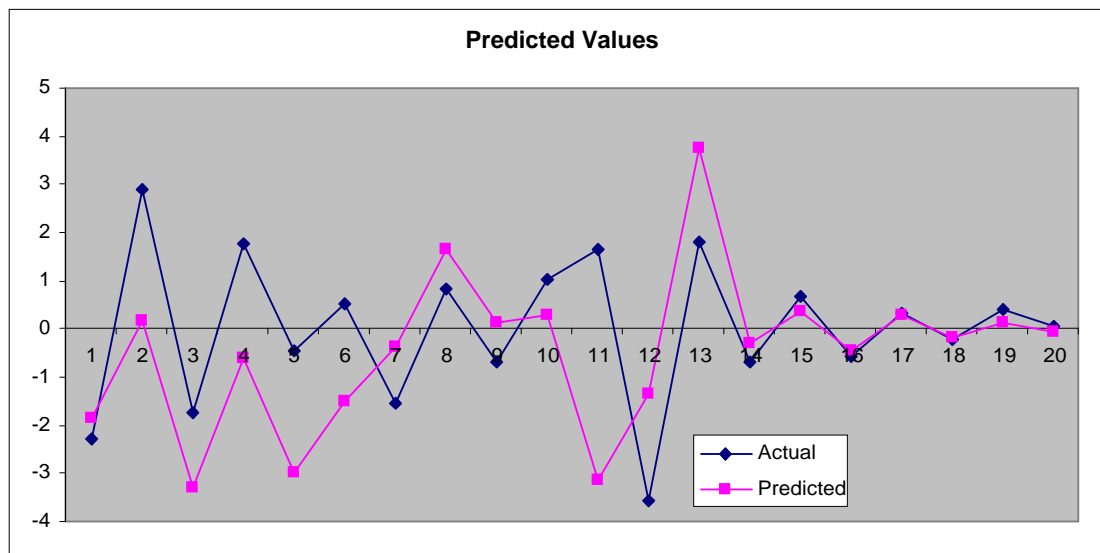


Figure 2.23

As we have already explained, once we have run out of actual values, the actual values of  $y(t)$  are replaced by their fitted values (starting from C102). This inevitably degrades forecasts, and we explained how different models behave. As we can see, our forecast for



cell C102 and C103 in Fig.2.21 is very good (as we know the actual values, we put them in cells A101:A105). Unfortunately our forecast for cell C104 begins to be significantly different from the known actual value in cell A104. This implies that for many time series the Box-Jenkins method is a very good fit, but only for short-term forecasts.

To summarise, in this section we not only showed the whole process of identifying models, fitting them and forecasting, but we also presented a much quicker way of doing it. We linked the values of ARMA coefficients directly with the residual sum of squares, which became a target value in the Solver, and which in one single step produced optimal values for these coefficients.

## **Automatic Box Jenkins/ARIMA: Method 2**

The subjective procedure outlined above requires considerable intervention from the statistician/economist completing the forecast. Various attempts have been made to automate the forecasts. The simplest of these fits a selection of models to the data, decides which is the “best” and then if the “best” is good enough uses that. Otherwise the forecast is referred back for “standard” analysis by the statistician/economist. The selection will be based on a criterion such as the AIC (Akaike’s Information Criterion) and the Bayesian (Schwarz) criterion (BIC or SIC)

Before I proceed to implement the automated model in a worksheet, there are 2 important Excel functions that I want to explain.

- i) **SumProduct ()**
- ii) **Offset ()**

### **i) SumProduct ()**

In Excel, the **SumProduct** function multiplies the corresponding items in the arrays and returns the sum of the results.

The syntax for the **SumProduct** function is:

**SumProduct( array1, array2, ... array\_n )**

*array1, array2, ... array\_n* are the ranges of cells or arrays that you wish to multiply. All arrays must have the same number of rows and columns. You must enter at least 2 arrays and you can have up to 30 arrays.

Note: If all arrays provided as parameters do not have the same number of rows and columns, the SumProduct function will return the #VALUE! error.

If there are non-numeric values in the arrays, these values are treated as 0's by the SumProduct function.

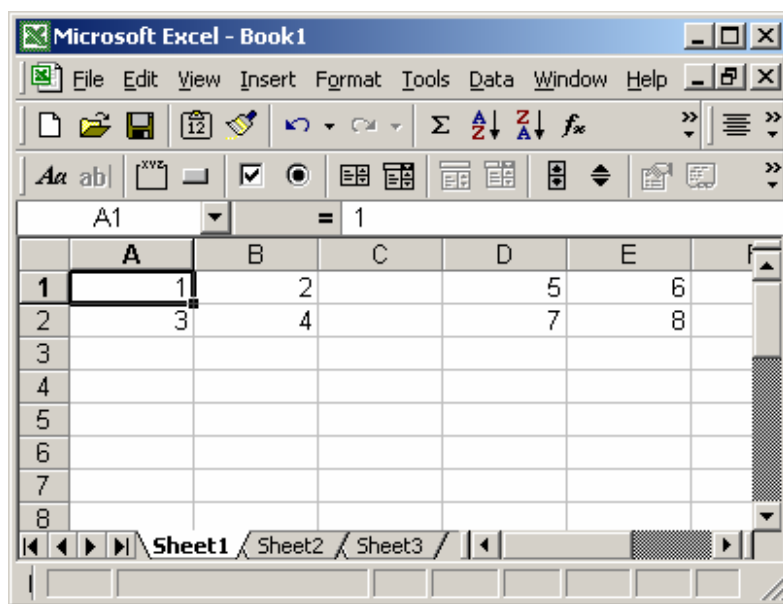
Let's take a look at an example:

`=SumProduct({1,2;3,4}, {5,6;7,8})`

The above example would return 70. The SumProduct calculates these arrays as follows:

`=(1*5) + (2*6) + (3*7) + (4*8)`

You could also reference ranges in Excel.



Based on the Excel spreadsheet above, you could enter the following formula:

`=SumProduct(A1:B2, D1:E2)`

This would also return the value 70. Another example:

	A	B	C
1	2		1
2	3		2
3	4		3
4	5		4

`=SumProduct(A1:A4, C1:C4).`

This will be  $(2*1) + (3*2) + (4*3) + (5*4) = 40$

## ii) Offset ()

The OFFSET() function returns a cell or range of cells that is a specified number of rows and/or columns from the reference cell. In this Tutorial we will explain the most common offset() applications and mistakes that are made using this function in Microsoft Excel.

The syntax for OFFSET () is

OFFSET (cell reference, rows, columns, [ height ], [ width ] )

Components in square brackets can be omitted from the formula.

### How does the Excel function OFFSET work?

The OFFSET() function returns a cell or range of cells that is a specified number of rows and/or columns from the reference cell. For specific descriptions of each component, please see the Help file in Excel.

If either of the “rows”, “columns”, “height” or “width” components is left blank, Excel will assume its value to be zero. For example, if the formula is written as OFFSET(C38, , 1, , ) Excel will interpret this as OFFSET(C38, 0, 1, 0, 0). This can also be written as OFFSET(C38, , 1) since “height” and “width” can be omitted.

Note that if “height” and “width” are included in the formula, they cannot be equal zero or a #REF! error will result. Examples below illustrate the function. Given the following set of numbers

#### OFFSET Example 1

OFFSET(D10, 1, 2) will give the value in F11 or 7, ie, Excel returns the value in the cell 1 row below and 2 columns to the right of D10

	A	B	C	D	E	F	G	H
9								
10				1	2	3	4	
11				5	6	7	8	
12				9	10	11	12	
13								

#### OFFSET Example 2

OFFSET(G12, -2, -2) will give the value in E10 or 2, ie, Excel returns the value in the cell 2 rows above and 2 columns to the left of G12

	A	B	C	D	E	F	G	H
9								
10				1	2	3	4	
11				5	6	7	8	
12				9	10	11	12	
13								

### OFFSET Example 3

OFFSET(F12, , , -2, -3) will return the 2 row by 3 column range D11:F12. Note that the reference cell F12 is included in this range

	A	B	C	D	E	F	G	H
9								
10				1	2	3	4	
11				5	6	7	8	
12				9	10	11	12	
13								

### OFFSET Example 4

OFFSET(D10, 1, 1, 2, 3) will return the range E11:G12, ie, Excel first calculates OFFSET(D10, 1, 1) which is E11 (1 row below and 1 column to the right of reference cell D10), then applies the formula OFFSET(E11, , , 2, 3)

	A	B	C	D	E	F	G	H
9								
10				1	2	3	4	
11				5	6	7	8	
12				9	10	11	12	
13								

### Common problems and mistakes with the OFFSET function

When tracing OFFSET( ) functions, only the reference cell is returned. For example, when tracing the precedent of OFFSET(D10, 1, 1, 2, 3) the returned cell is D10 and not E11:G12.

Excel excludes the reference cell when calculating the “rows” and “columns” components, but includes the reference cell when calculating the “height” and “width” components.

This can be confusing, and requires extreme care. OFFSET() is a complex concept to grasp which reduces user confidence in the model since it is not easily understood.

### Combining OFFSET() with Other Functions

Since OFFSET() returns a cell or a range of cells, it can be easily combined with other functions such as SUM(), SUMPRODUCT(), MIN(), MAX(), etc.

For example, SUM(OFFSET()) calculates the sum of the cell or range of cells returned by the OFFSET() function. Extending from Example 4 above, SUM(OFFSET(D10, 1, 1, 2, 3)) is equivalent to writing SUM(E11 : G12) (as OFFSET(D10, 1, 1, 2, 3) returns the range E11 : G12) which equals  $54 = 6 + 7 + 8 + 10 + 11 + 12$ . Similarly, AVERAGE(OFFSET(D10, 1, 1, 2, 3)) is equivalent to AVERAGE(E11 : G12).

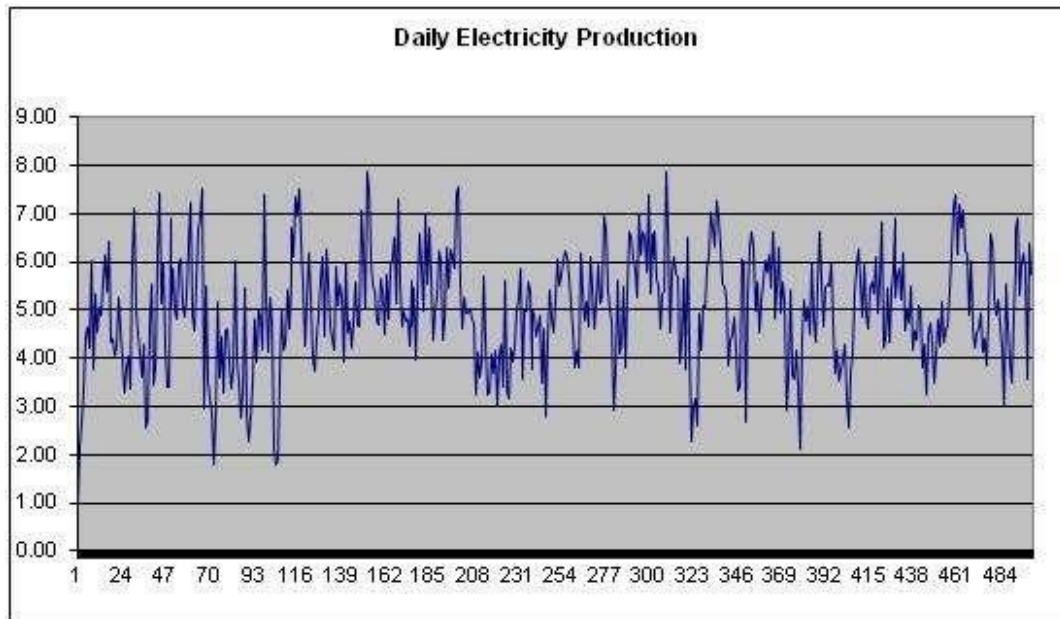
As explained in Method 1, Arima modeling involves 4 major steps.

### A) MODEL IDENTIFICATION

As mentioned earlier, there is a clear need for automatic, objective methods of identifying the best ARMA model for the data at hand. Objective methods become particularly crucial when trained experts in model building are not available. Furthermore, even for experts, objective methods provide a very useful additional tool, since the correlogram and partial correlogram do not always point clearly to single best model. The two most widely used criteria are the Akaike information criterion (AIC) and the Bayesian (Schwarz) criterion (BIC or SIC)

$$\begin{aligned} \text{AIC}(p, q) &= \ln(\hat{\sigma}^2) + \frac{2(p+q)}{T} \\ \text{BIC}(p, q) &= \ln(\hat{\sigma}^2) + \frac{\ln(T)(p+q)}{T} \\ \hat{\sigma}^2 &= \text{estimate of } \sigma^2 \text{ from ARMA}(p, q) \end{aligned}$$

Don't worry about the formula above. They are easily implemented in an Excel worksheet. Let's me show you how to build this automated model identification with a spreadsheet example. Open *Worksheet (Work(2))*. The data is from the daily electricity production in a developing country: million kilowatts per day are enter in cell A2:A501.



X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

	K	L	M	N
1	<b>p</b>	<b>3</b>	<b>2</b>	<b>q</b>
2	a(10)	0.1	0.1	c(10)
3	a(9)	0.1	0.2	c(9)
4	a(8)	0.1	0.3	c(8)
5	a(7)	0.1	0.1	c(7)
6	a(6)	0.1	0.1	c(6)
7	a(5)	0.1	0.1	c(5)
8	a(4)	0.1	0.1	c(4)
9	a(3)	0.1	0.1	c(3)
10	a(2)	0.1	0.1	c(2)
11	a(1)	0.1	0.1	c(1)
12		3	2	

Figure 2.24

The coefficients are as follow:

a(1) = L11, a(2) = L10, a(3) = L9  
c(1) = M11, c(2) = M10

Or if it is a ARMA(2,1) then the cell in L1 will show a 2 and M1 will be a 1. The corresponding coefficients are cells L10:L11 for the AR and cell M11 for the MA. (see the blue numbers in Fig 2.25 below)

	K	L	M	N
1	<b>p</b>	<b>2</b>	<b>1</b>	<b>q</b>
2	a(10)	0.1	0.1	c(10)
3	a(9)	0.1	0.2	c(9)
4	a(8)	0.1	0.3	c(8)
5	a(7)	0.1	0.1	c(7)
6	a(6)	0.1	0.1	c(6)
7	a(5)	0.1	0.1	c(5)
8	a(4)	0.1	0.1	c(4)
9	a(3)	0.1	0.1	c(3)
10	a(2)	0.1	0.1	c(2)
11	a(1)	0.1	0.1	c(1)
12		2	1	

Figure 2.25

The INT function in Excel is used to remove all decimal places leaving only the whole number. Removing decimal places, or the fractional part of a number is necessary to use Excel Solver for our modeling.

Cell L12 is related to L1 as L1 is a whole number or integer of L12. We need to enter the function Int() in cell L1 as Excel Solver will return an error if L1 is not an Integer. The same goes for M1 and M12. They are related for the same reason.

As promised earlier, I'll include the calculation of **d** in this example. The formula for **d** is enter in cell I5. (You can refer on how **d** is derived by looking at page 35 above.)

For easier understanding, the general equation above is broken into 3 parts.

- formula **d** is enter in cell I5
- a(1)\*y(t-1) + a(2)\*y(t-2) + ... + a(p)\*y(t-p)** enter in column B
- and **e(t) - c(1)\*e(t-1) - c(2)\*e(t-2) -...- c(p)\*e(t-p)** enter in column C

i) Read page 35 above to understand how **d** is calculated.

The formula for **d** in I5 is:

$$=I2*(1-(SUM(OFFSET(L12,-1,0):OFFSET(L12,-L1,0))))$$

For a ARMA(2,1), this mean  $I2 * (1 - (\text{SUM}(L10:L11)))$ .

For a ARMA(3,2) then it is  $I_2 * (1 - \text{SUM}(L9:L11))$ .

X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

[illegible]



This part of the book is not available for viewing

[illegible]

X

$$xc(1)*e(t-1) - c(2)*e(t-2) \Rightarrow M11*D3 - M10*D2$$

If  $q = 3$  then the formula is (calculation start from C5)

$$c(1)*e(t-1) - c(2)*e(t-2) - c(3)*e(t-3) \Rightarrow M11*D4 - M10*D3 - M9 - D2$$

etc...

As for the residuals or errors  $e(t)$ , these are enter in column D. (see Fig 2.29 below)

	D	E
1	$e(t)$	$y(t)$
2	0	0
3	=A3-(\$I\$5+(B3-C3))	=I\$5 +B3-C3
4	=A4-(\$I\$5+(B4-C4))	=I\$5 +B4-C4
5	=A5-(\$I\$5+(B5-C5))	=I\$5 +B5-C5
6	=A6-(\$I\$5+(B6-C6))	=I\$5 +B6-C6
7	=A7-(\$I\$5+(B7-C7))	=I\$5 +B7-C7
8	=A8-(\$I\$5+(B8-C8))	=I\$5 +B8-C8

Figure 2.29

And lastly the full formula are enter in column E

- $d = 15$
- Column B =  $a(1)*y(t-1) + a(2)*y(t-2) + \dots + a(p)*y(t-p)$
- Column C =  $e(t) - c(1)*e(t-1) - c(2)*e(t-2) - \dots - c(p)*e(t-p)$
- Column D =  $e(t)$

Thus the full formula in column E. (see Fig 2.29 above)

Before we invoke Excel Solver to solve the parameters  $p$ ,  $q$  and their coefficients, let me explain to you the AIC and BIC as we are using an objective method for model identification in the next section.

## B) MODEL ESTIMATION

Because of the highly subjective nature of the Box-Jenkins methodology, time series analysts have sought alternative objective methods for identifying ARMA models. Penalty function statistics, such as Akaike Information Criterion [AIC] or Final Prediction Error [FPE] Criterion (Akaike, 1974), Schwarz Criterion [SC] or Bayesian Information Criterion [BIC] (Schwarz, 1978) have been used to assist time series analysts in reconciling the need to minimize errors with the conflicting desire for model parsimony. These statistics all take the form minimizing the sum of the residual sum of squares plus a 'penalty' term which incorporates the number of estimated parameter coefficients to factor in model parsimony..

### Akaike Information Criterion (AIC)

$$AIC = \log\left(\frac{rss}{n}\right) + \left(2 * \frac{k}{n}\right)$$

### Bayesian Information Criterion (BIC)

$$BIC = \log\left(\frac{rss}{n}\right) + \left(\log(n) * \frac{k}{n}\right),$$

where,

**k** = number of coefficients estimated (1 + p + q + P + Q)

**rss** = residual sum of squares

**n** = number of observations.

Assuming there is a true ARMA model for the time series, the BIC and HQC have the best theoretical properties. The BIC is strongly consistent whereas AIC will usually result in an overparameterised model; that is a model with too many AR or MA terms (Mills 1993, p.29). Indeed, it is easy to verify that for n greater than seven the BIC imposes a greater penalty for additional parameters than does the AIC.

Thus, in practice, using the objective model selection criteria involves estimating a range of models and the one with the lowest information criterion is selected. These 2 formulas are entered in cells I11 for the AIC and I12 for the BIC. (See Fig 2.30 below)

	H	I
1		
2	<b>Mean</b>	=AVERAGE(A2:A501)
3	<b>St. Dev.</b>	=STDEV(A2:A501)
4	<b>Measure</b>	=I2+I3
5	<b>d</b>	=I2*(1-(SUM(OFFSET(L1,1,0,):OFFSET(L1,L1,0))))
6	$\bar{e}$	=AVERAGE(D3:D501)
7	<b>SE<sub>e</sub></b>	=STDEV(D3:D501)/SQRT(COUNT(D3:D501))
8	<b>Value</b>	=1.96*I7
9	<b>Verdict:</b>	=IF(I6>I8,"Nonzero mean","Zero mean")
10		
11	<b>AIC</b>	=LN(STDEV(D3:D501)/COUNT(D3:D501))+(2*I2/COUNT(D3:D501))
12	<b>BIC</b>	=LN(STDEV(D3:D501)/COUNT(D3:D501))+LN(COUNT(D3:D501)*2/COUNT(D3:D501))
13	<b>SSE</b>	=SUMSQ(D2:D501)

Figure 2.30

We also need to define the admissible region which will guarantee that our model is stationary and invertible. The coefficients of AR models must be within a permissible region in order to guarantee stationarity and there is also a permissible region for the coefficients of MA models which guarantees invertibility. Every MA model is stationary

by definition, but is invertible only if certain conditions are satisfied. By the way, AR models are invertible for all values of coefficients, but only stationary if coefficients are in a particular admissible region. In Fig 2.30 above the admissible region ensuring stationarity is given in cell I20 and the admissible region ensuring invertibility is given in cell I21. As we have a generalize model for automated Arima, the formula to ensure stationarity and invertibility are

$$-1 \leq \sum p \leq 1 \quad \text{and} \quad -1 \leq \sum q \leq 1.$$

*It's now time to use Excel Solver for our Automated Arima(p,q) model*

Open the *Worksheet(Work(3))*. *Worksheet(Work(3))* is just a copy of *Worksheet(Work(2))*. To use the Solver, click on the Tools heading on the menu bar and select the Solver . . . item. (see Figure 2.31)

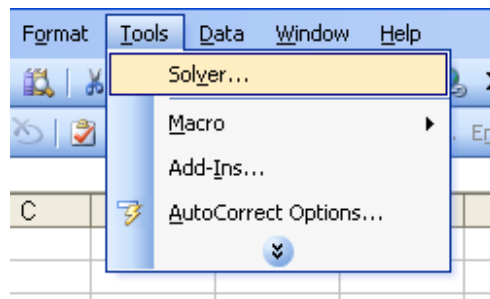


Figure 2.31

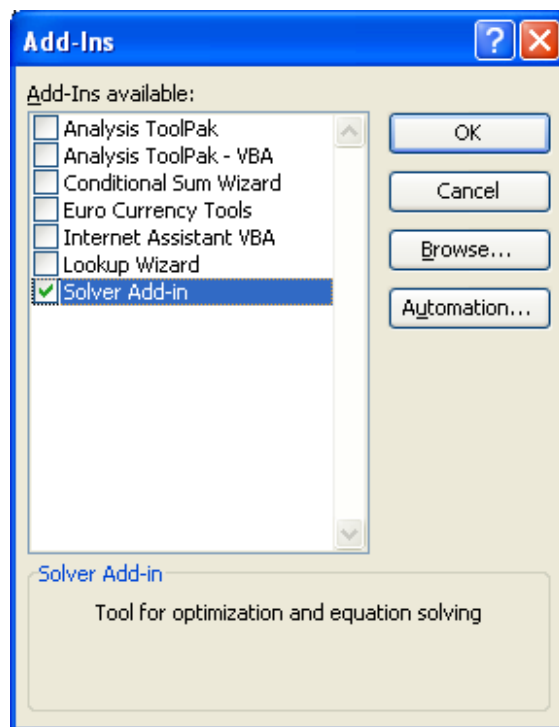


Figure 2.32

If Solver is not listed (see Figure 2.31) then you must manually include it in the algorithms that Excel has available. To do this, select Tools from the menu bar and choose the "Add-Ins . . ." item. In the Add-Ins dialog box, scroll down and click on the Solver Add-In so that the box is checked as shown Figure 2.32 above:

After selecting the Solver Add-In and clicking on the OK button, Excel takes a moment to call in the Solver file and adds it to the Tools menu.

If you cannot find the Solver Add-In, try using the Mac's Find File or Find in Windows to locate the file. Search for "solver." Note the location of the file, return to the Add-Ins dialog box (by executing Tools: Add-Ins...), click on Select or Browse, and open the Solver Add-In file.

What if you still cannot find it? Then it is likely your installation of Excel failed to include the Solver Add-In. Run your Excel or Office Setup again from the original CD-ROM and install the Solver Add-In. You should now be able to use the Solver by clicking on the Tools heading on the menu bar and selecting the Solver item.

Although Solver is proprietary, you can download a trial version from Frontline Systems, the makers of Solver, at [www.frontsys.com](http://www.frontsys.com).

After executing Tools: Solver . . . , you will be presented with the Solver Parameters dialog box below:

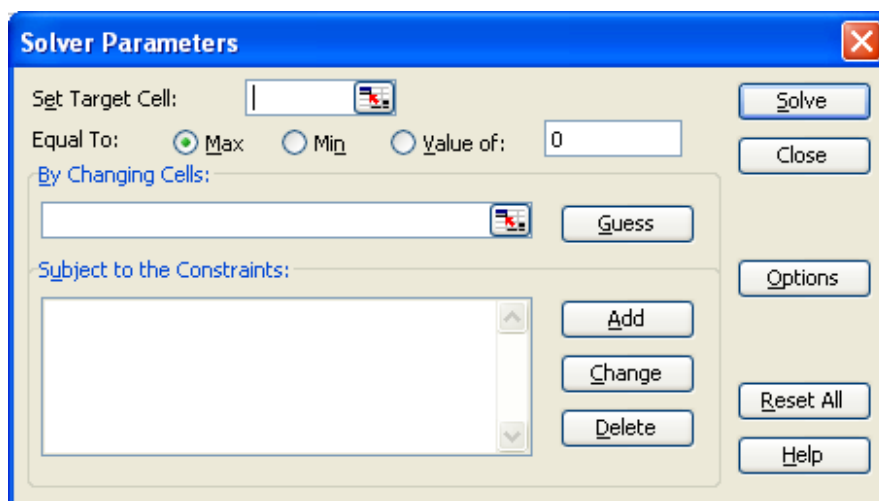


Figure 2.33

Let us review each part of this dialog box, one at a time.

**Set Target Cell** is where you indicate the objective function (or goal) to be optimized. This cell must contain a formula that depends on one or more other cells (including at

least one “changing cell”). You can either type in the cell address or click on the desired cell. Here we enter cell **A11**.

In our Arima model, the objective function is to minimize the AIC in cell I11. See Figure 2.34 below.

**Equal to:** gives you the option of treating the Target Cell in three alternative ways. **Max** (the default) tells Excel to maximize the Target Cell and **Min**, to minimize it, whereas **Value** is used if you want to reach a certain particular value of the Target Cell by choosing a particular value of the endogenous variable.

Here, we select **Min** as we want to minimize the AIC. (You may also try I12 the BIC)

For starting value, I use **p** and **q** = 5. The coefficients = 0.1 (see Fig 2.34 below).

	G	H	I	J	K	L	M	N
1					<b>p</b>	<b>5</b>	<b>5</b>	<b>q</b>
2		<b>Mean</b>	5.00			0.1	0.1	
3		<b>St. Dev.</b>	1.17			0.1	0.1	
4		<b>Measure</b>	6.17			0.1	0.1	
5		<b>d</b>	2.501482211			0.1	0.1	
6		$\bar{e}$	0.01582			0.1	0.1	
7		<b>SE<sub>e</sub></b>	0.044850793			0.1	0.1	
8		<b>Value</b>	0.087907553			0.1	0.1	
9		<b>Verdict:</b>	<b>Zero mean</b>			0.1	0.1	
10						0.1	0.1	
11		<b>AIC</b>	-6.202701035			0.1	0.1	
12		<b>BIC</b>	-5.517569886			5	5	
13		<b>SSE</b>	500.00998217					
14								
15		<b>Durbin - Watson Test</b>						
16		778.3995						
17		500.0100						
18		1.55677						
19								
20		<b>Permissible regions for p</b>	0.5					
21		<b>Permissible regions for q</b>	0.5					
22								
23								
24								
25								
26		<b>Mean</b>	5.00					
27		<b>1.96*SE</b>	0.0076314255549013					
28		<b>Zero Mean Test</b>	<b>Zero</b>					

Figure 2.34

**By Changing Cells** permits you to indicate which cells are the adjustable cells (i.e., endogenous variables). As in the Set Target Cell box, you may either type in a cell address or click on a cell in the spreadsheet. Excel handles multivariable optimization problems by allowing you to include additional cells in the By Changing Cells box. Each noncontiguous choice variable is separated by a comma. If you use the mouse technique (clicking on the cells), the comma separation is automatic.

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

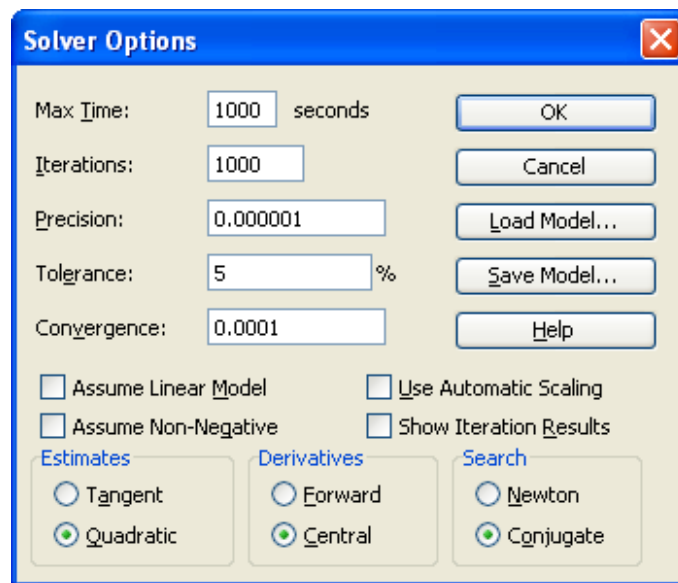


Figure 2.37

As you can see, a series of choices are included in the Solver Options dialog box that direct Solver's search for the optimum solution and for how long it will search. These options may be changed if Solver is having difficulty finding the optimal solution. Lowering the Precision, Tolerance, and Convergence values slows down the algorithm but may enable Solver to find a solution.

For a Arima model, you can set :

- i) Max Time: 1000 seconds
- ii) Iterations: 1000
- iii) Precision: 0.000001
- iv) Tolerance: 5%
- v) Convergence: 0.0001

Select **Conjugate** as the *Search* method. This prove to be very effective in minimizing the AIC.

The Load and Save Model buttons enable you to recall and keep a complicated set of constraints or choices so that you do not have to reenter them every time.

Clicking OK to return to the Solver Parameters dialog box.

**Solve** is obviously the button you click to get Excel's Solver to find a solution. This is the last thing you do in the Solver Parameters dialog box. So, click **Solve** to start training.



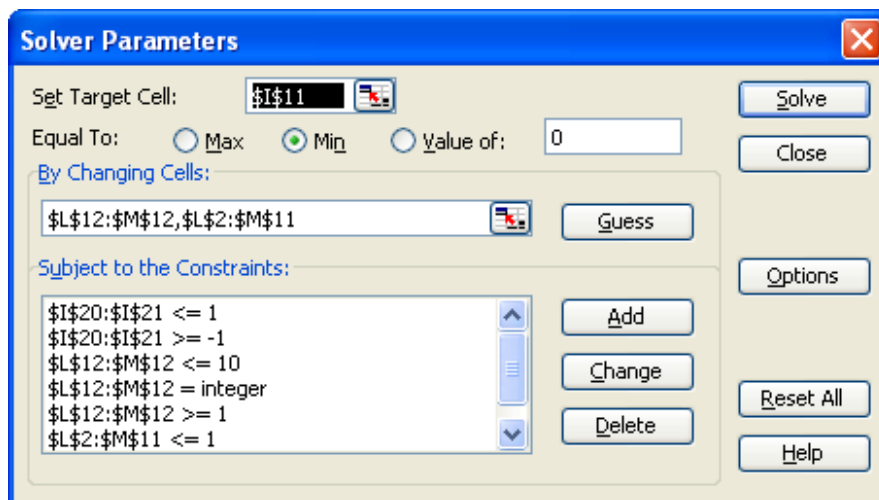


Figure 2.38

53	4.99	2.643130769	1.18581345	-0.70
54	4.83	3.68354068	-0.5922962	-0.10
55	6.00	2.658628949	0.95484608	0.50

Work (4) / Work (2) / Daily

Trial Solution: 7 Set Cell: -6.248060439

Figure 2.39

When Solver start optimizing, you will see the *Trial Solution* at the bottom left of your spreadsheet. See Figure 2.39 above.

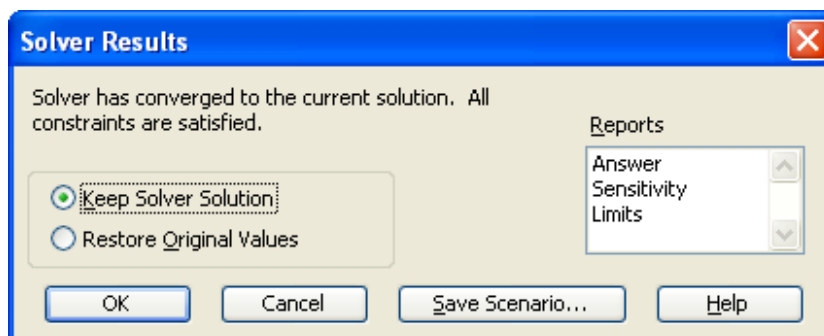


Figure 2.40

A message will appears after Solver has converged (see Figure 2.40)..In this case, Excel reports that “Solver has converged to the current solution. All constraints are satisfied.” This is good news!

Sometime, the solution is not satisfactory and Solver unable to find the solution at one go. For example it may failed the stationary test as indicated in cell I9 i.e not a zero mean. If this is the case then you, change the starting parameters for **p**, **q** and the coefficients and run Solver again. Follow the step discussed above. From my experience, usually you will need to run Solver a few times before Solver arrive at a satisfactory solution.

Bad news is a message like, “Solver could not find a solution.” If this happens, you must diagnose, debug, and otherwise think about what went wrong and how it could be fixed. The two quickest fixes are to try different initial **p**, **q** parameters and their coefficients.

From the Solver Results dialog box, you elect whether to have Excel write the solution it has found into the Changing Cells (i.e., Keep Solver Solution) or whether to leave the spreadsheet alone and NOT write the value of the solution into the Changing Cells (i.e., Restore Original Values). When Excel reports a successful run, you would usually want it to Keep the Solver Solution. On the right-hand side of the Solver Results dialog box, Excel presents a series of reports. The Answer, Sensitivity, and Limits reports are additional sheets inserted into the current workbook. They contain diagnostic and other information and should be selected if Solver is having trouble finding a solution.

	H	I	J	K	L	M	N
1				<b>p</b>	<b>1</b>	<b>1</b>	<b>q</b>
2	Mean	5.00	a(10)	0.09965	0.04717	c(10)	
3	St. Dev.	1.17	a(9)	0.09965	0.1459	c(9)	
4	Measure	6.17	a(8)	0.09965	0.24489	c(8)	
5	d	1.303359276	a(7)	0.09965	0.04688	c(7)	
6	$\bar{e}$	0.01317	a(6)	0.09965	0.04497	c(6)	
7	SE <sub>e</sub>	0.043377072	a(5)	0.09965	0.04477	c(5)	
8	Value	0.085019062	a(4)	0.09965	0.04537	c(4)	
9	Verdict:	Zero mean	a(3)	0.09965	0.04407	c(3)	
10			a(2)	0.09965	0.04462	c(2)	
11	AIC	-6.236111282	a(1)	0.73951	0.32419	c(1)	
12	BIC	-5.550980133			1	1	
13	SSE	467.66060910					
14							
15	Durbin - Watson Test						
16	920.4099						
17	467.6606						
18	1.96812						
19							
20	Permissible regions for p	0.099654584					
21	Permissible regions for q	0.047167779					
22							
23							
24							
25							
26	Mean	5.00					
27	1.96*SE	0.00763					
28	Zero Mean Test	Zero					

Figure 2.41

My first run of Excel Solver come to the above solution. AIC = -6.236111282 in cell I11. As indicated in the Figure 2.41 above, we have a ARMA(1,1) model. The coefficients are in cells L11 and M11. It has passed all the tests as you can see cells I9 and H18 in Figure 2.41. (Note: Depending on the data you have, sometimes you need to run Solver a few times before you come to a satisfactory solution).

### C) DIAGNOSTIC CHECKING:

How do we know that we have produced a reasonable model and that our model indeed reflects the actual time series? This is a part of the process that Box and Jenkins refer to as diagnostic checking. I will use two methods to conduct the diagnostic. As we expect the forecasting errors to be completely random, the first step is to plot them, as we did in Fig. 2.42 below for example. This residuals chart indicates randomness. But we want to make sure so we need to do the calculation.

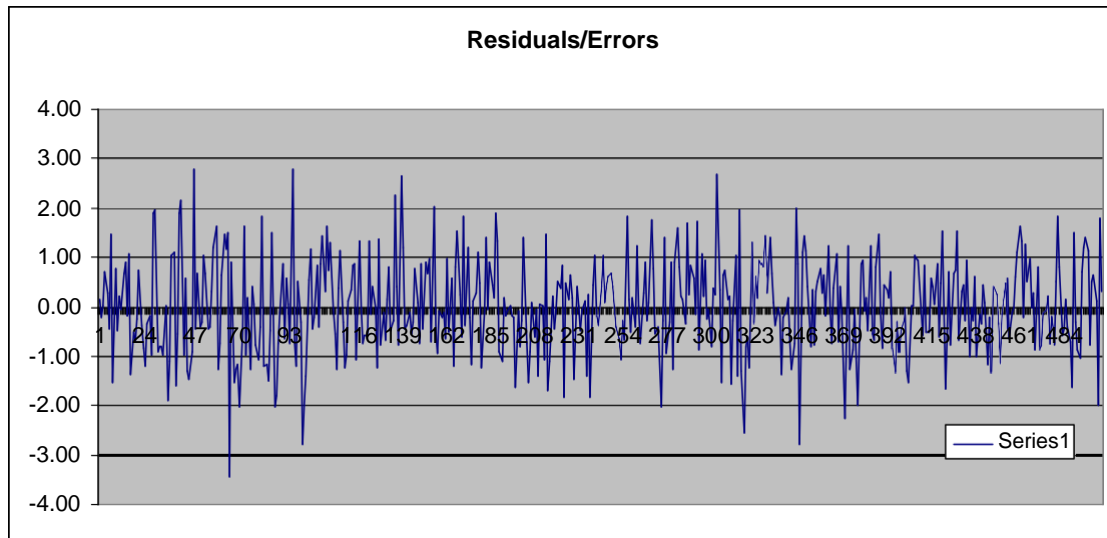


Figure 2.42

One of the requirements is that the residual mean should be zero, or close to zero. To establish that this is the case, we need to estimate the standard error of the mean error. This is calculated as:

$$\sigma_e = \sqrt{\frac{\sum_{t=1}^n (e_t - \bar{e})^2}{n}}$$

$$SE_{\bar{e}} = \frac{\sigma_e}{\sqrt{n}} \quad \text{in cell I7}$$

Where  $\sigma_e$  is the residual standard deviation,  $\bar{e}$  is the mean error,  $n$  is the number of errors and  $SE_{\bar{e}}$  is the standard error of the mean error. If the residual mean  $\bar{e}$  is greater than 1.96 standard errors, then we can say that it is significantly non-zero:

$$\bar{e} > 1.96 SE_{\bar{e}} \quad \text{in cell I9}$$

How to estimate the standard residual error  $SE_e$  (standard error) is shown below in Fig 2.43 and the formulas are given in Fig. 2.44 below

	H	I	J	K	L	M	N
1				<b>p</b>	<b>1</b>	<b>1</b>	<b>q</b>
2	Mean	5.00		a(10)	0.09965	0.04717	c(10)
3	St. Dev.	1.17		a(9)	0.09965	0.1459	c(9)
4	Measure	6.17		a(8)	0.09965	0.24489	c(8)
5	d	1.303359276		a(7)	0.09965	0.04688	c(7)
6	$\bar{e}$	0.01317		a(6)	0.09965	0.04497	c(6)
7	$SE_e$	0.043377072		a(5)	0.09965	0.04477	c(5)
8	Value	0.085019062		a(4)	0.09965	0.04537	c(4)
9	Verdict:	Zero mean		a(3)	0.09965	0.04407	c(3)
10				a(2)	0.09965	0.04462	c(2)
11	AIC	-6.236111282		a(1)	0.73951	0.32419	c(1)
12	BIC	-5.550980133			1	1	
13	SSE	467.66060910					
14							
15	Durbin - Watson Test						
16	920.4099						
17	467.6606						
18	1.96812						
19							
20	Permissible regions for p	0.099654584					
21	Permissible regions for q	0.047167779					
22							
23							
24							
25							
26	Mean	5.00					
27	1.96*SE	0.00763					
28	Zero Mean Test	Zero					

Figure 2.43

	H	I
1		
2	<b>Mean</b>	=AVERAGE(A2:A501)
3	<b>St. Dev.</b>	=STDEV(A2:A501)
4	<b>Measure</b>	=I2+I3
5	<b>d</b>	=I2*(1-(SUM(OFFSET(L1,1,0):OFFSET(L1,L1,0))))
6	$\bar{e}$	=AVERAGE(D3:D501)
7	<b>SE<sub>e</sub></b>	=STDEV(D3:D501)/SQRT(COUNT(D3:D501))
8	<b>Value</b>	=1.96*I7
9	<b>Verdict:</b>	=IF(I6>I8,"Nonzero mean","Zero mean")
10		
11	<b>AIC</b>	=LN(STDEV(D3:D501)/COUNT(D3:D501))+(2*2/COUNT(D3:D501))
12	<b>BIC</b>	=LN(STDEV(D3:D501)/COUNT(D3:D501))+(LN(COUNT(D3:D501)*2/COUNT(D3:D501)))
13	<b>SSE</b>	=SUMSQ(D2:D501)
14		
15	<b>Durbin - Watson Test</b>	
16	=SUMXMY2(D3:D501,D2:D500)	
17	=SUMSQ(D2:D501)	
18	=H16/H17	
19		
20	<b>Permissible regions for p</b>	=SUM(OFFSET(\$L\$1,1,0):OFFSET(\$L\$1,\$L\$1,0))
21	<b>Permissible regions for q</b>	=SUM(OFFSET(\$M\$1,1,0):OFFSET(\$M\$1,\$M\$1,0))
22		
23		
24		
25		
26	<b>Mean</b>	=AVERAGE(A2:A501)
27	<b>1.96*SE</b>	=1.96*(SQRT(A2:A501)/COUNT(A2:A501))
28	<b>Zero Mean Test</b>	=IF(I26<I27,"Non-zero","Zero")

Figure 2.44

Cell I9 contains a brief IF statement evaluating whether the calculated mean value from I6 is greater than the standard error times 1.96. In our model we have zero mean which pass the test.

Another test that is quite popular is the Durbin-Watson test, which is use in the context of checking the validity of ARIMA models. The **Durbin–Watson statistic** is a test statistic used to detect the presence of autocorrelation in the residuals from a regression analysis. It is named after James Durbin and Geoffrey Watson.

If  $e_t$  is the residual associated with the observation at time  $t$ , then the test statistic is

$$w = \frac{\sum_{t=2}^T (e_t - e_{t-1})^2}{\sum_{t=1}^T e_t^2}.$$

H16 contain the upper part of the above formula and H17 contain the lower part. Since  $w$  in cell H18 is approximately equal to  $2(1-r)$ , where  $r$  is the sample autocorrelation of the residuals,  $w = 2$  indicates no autocorrelation. The value of  $w$  always lies between 0 and 4. If the Durbin–Watson statistic is substantially less than 2, there is evidence of positive serial correlation. As a rough rule of thumb, if Durbin–Watson is less than 1.0, there may be cause for alarm. Small values of  $w$  indicate successive error terms are, on average,

close in value to one another, or positively correlated. If  $w > 2$  successive error terms are, on average, much different in value to one another, i.e., negatively correlated. In regressions, this can imply an underestimation of the level of statistical significance.

In our model we have 1.96812 in cell H18 which is very close to 2 which indicates no autocorrelation. See Fig 2.43 above. We can now proceed with the forecast.

#### D) FORECAST

Now we are ready to produce real forecasts, i.e. those that go into the future. The equation can be applied “one step ahead” to get estimate  $y(t)$  from observed  $y(t-1)$ . A “ $k$ -step-ahead” prediction can also be made by recursive application of equation. In recursive application, the observed  $y$  at time 1 is used to generate the estimated  $y$  at time 2. That estimate is then substituted as  $y(t-1)$  to get the estimated  $y$  at time 3, and so on. The  $k$ -step-ahead predictions eventually converge to zero as the prediction horizon,  $k$ , increases. On worksheet *Work(3)* goto cell A502:A507.

We will forecast as per the formula below: Arima(1,0,1) or ARMA(1,1)

We use the formula:

$$y(t) = 1.30335 + 0.73951*y(t-1) - 0.32419*e(t-1)$$

	A	B	C	D	E
497	6.20	4.425632928	0.16730218	0.64	5.56
498	5.80	4.584982642	0.206932279	0.12	5.68
499	3.55	4.292071756	0.03971441	-2.01	5.56
500	6.38	2.625272319	-0.650228773	1.80	4.58
501	5.75	4.717847394	0.583798854	0.31	5.44
502	5.38				5.4542
503	5.44				5.3368
504	5.28				5.25
505	5.00				5.1858
506	4.82				5.1383
507	4.72				5.1032

Figure 2.45

	A	B	C	D	E
500	6.379	=IF(\$L\$5=	=IF(\$M\$5=A500-	=(\$I\$5 +B500-C500	
501	5.75	=IF(\$L\$5=	=IF(\$M\$5=A501-	=(\$I\$5 +B501-C501	
502	5.376			=(\$I\$5+(\$L\$11*A501)-\$M\$11*D501	
503	5.435			=(\$I\$5+(\$L\$11*E502)	
504	5.278			=(\$I\$5+(\$L\$11*E503)	
505	5			=(\$I\$5+(\$L\$11*E504)	
506	4.819			=(\$I\$5+(\$L\$11*E505)	
507	4.721			=(\$I\$5+(\$L\$11*E506)	

Figure 2.46

Fig. 2.45 shows the spreadsheet containing the forecasted values numbers and Fig 2.46 shows all the calculations and formulas.

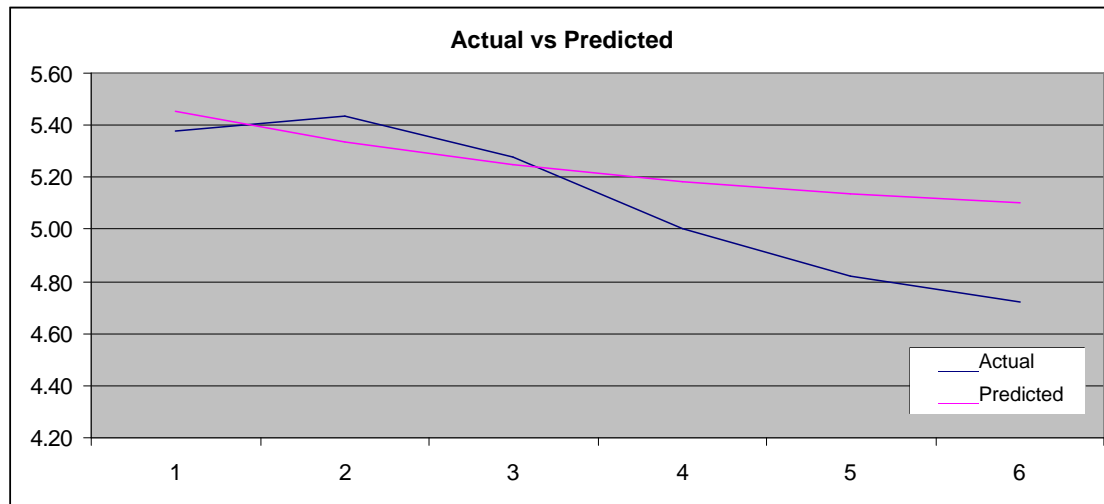


Figure 2.47

As we have already explained, once we have run out of actual values, the actual values of  $y(t)$  are replaced by their fitted values (starting from E503). This inevitably degrades forecasts, and we explained how different models behave. As we can see, our forecast for cell E502 and E505 in Fig 2.45 is very good (as we know the actual values, we put them in cells A502:A507 to compare). Unfortunately our forecast for cell E506 begins to be significantly different from the known actual value in cell A506. This implies that for many time series the Arima method is a very good fit, but only for short-term forecasts.


You may not have such an ideal model. It takes me about 10 run of Excel Solver on the model before I came to this result. Change the **p**, **q** and their coefficients starting values and then run Solver until you come to a satisfactory solution. It takes a little bit of testing and running.

Another way you can make use of the model is use Excel Solver to optimize the coefficients only. You enter the **p** and **q** manually and use Solver to optimize the coefficient. Let me give you an example. Open *Worksheet (Work(4))*. Enter 2 in both cells L12 and M12. (see Fig 2.48 below). So we have use an ARMA(2,2) model. Invoke Excel Solver and enter the parameters as shown in Fig 2.49 below.


	H	I	J	K	L	M	N
1				<b>p</b>	<b>2</b>	<b>2</b>	<b>q</b>
2	<b>Mean</b>	5.00		a(10)	0.1	0.1	c(10)
3	<b>St. Dev.</b>	1.17		a(9)	0.1	0.2	c(9)
4	<b>Measure</b>	6.17		a(8)	0.1	0.3	c(8)
5	<b>d</b>	4.002843671		a(7)	0.1	0.1	c(7)
6	$\bar{e}$	0.01151		a(6)	0.1	0.1	c(6)
7	<b>SE<sub>e</sub></b>	0.051401222		a(5)	0.1	0.1	c(5)
8	<b>Value</b>	0.100746395		a(4)	0.1	0.1	c(4)
9	<b>Verdict:</b>	<b>Zero mean</b>		a(3)	0.1	0.1	c(3)
10				a(2)	0.1	0.1	c(2)
11	<b>AIC</b>	-6.066380354		a(1)	0.1	0.1	c(1)
12	<b>BIC</b>	-5.381249205			2	2	
13	<b>SSE</b>	656.62962414					
14							
15	<b>Durbin - Watson Test</b>						
16	636.4766						
17	656.6296						
18	0.96931						
19							
20	missible regions for p	0.2					
21	missible regions for q	0.3					
22							
23							
24							
25							
26	<b>Mean</b>	5.00					
27	<b>1.96*SE</b>	0.00763					
28	<b>Zero Mean Test</b>	<b>Zero</b>					

Figure 2.48

**Solver Parameters**

Set Target Cell:  

Equal To: ☐ Max ☒ Min ☐ Value of:

By Changing Cells:  

Subject to the Constraints:

Buttons: **Solve**, **Close**, **Options**, **Reset All**, **Help**, **Add**, **Change**, **Delete**, **Guess**

Figure 2.49



Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

## Seasonal Arima Modeling (SARIMA):

We can use ARIMA models for seasonal time series forecasting. The underlying principles are identical to the ones for non-seasonal time series, described above. These seasonal time series show seasonal trends with periodicity  $s$

Seasonal series repeat themselves after a certain number of months, usually after twelve months, or every four months (quarterly seasonality). They can be either stationary or non-stationary. For the non-stationary seasonal time series, they need to be differenced. Unfortunately, ordinary differencing is not good enough for such cases. Seasonal differencing is what is needed.

For example,

- ☐ Monthly data has 12 observations per year
- ☐ Quarterly data has 4 observations per year
- ☐ Daily data has 5 or 7 (or some other number) of observations per week.

A SARIMA process has four components:

- ☐ *auto-regressive (AR)*,
- ☐ *moving-average (MA)*,
- ☐ *one-step differencing*,
- ☐ and *seasonal differencing*.

For a time series with a 12-month pattern, seasonal differencing is executed as follows:

The differencing formula requires that in a seasonal time series, we need to find differences between two comparable months, rather than between two successive months as it makes more sense. In this example, 12 is the number of months. If we assign letter  $s$  for seasonality, then the seasonal differencing is in general described as:

$$w(t) = y(t) - y(t-s)$$

Like ordinary differencing, sometimes a second level of differencing is needed. This is done as:

$$\nabla w(t) = w(t) - w(t-s)$$

If we substitute  $\nabla w(t) = w(t) - w(t-s)$   $w(t) = y(t) - y(t-s)$ , we get:

$$w(t) = [y(t) - y(t-s)] - [y(t-1) - y(t-s-1)] = y(t) - y(t-1) - y(t-s) + y(t-s-1)$$

Which for example for  $s=12$  gives:

$$\nabla w(t) = y(1) - y(t-1) - y(t-12) + y(t-13)$$

The above formula shows that:  $y(t) = y(t-1) + y(t-12) - y(t-13)$ , i.e. in this case the current observation is equal to the previous observation, plus the one twelve periods ago, less the one that preceded it! Sounds odd but if we rewrite it a little differently it will make a lot of sense:

$$y(t) - y(t-12) = y(t-1) - y(t-13)$$

Thus we are saying that this period's seasonal differences are the same as the seasonal differences observed in the previous period, which are more logical.

We can make an interesting digression here and ask ourselves what the next period's seasonal differences are going to be like. It is reasonable to assume that they will be something like:  $y_{t+1} - y_{t-11} = y_t - y_{t-12}$ , which is very interesting because we can see above, that  $y_t - y_{t-12} = y_{t-1} - y_{t-13}$ . Essentially we are saying that  $y_{t+1} - y_{t-11} = y_{t-1} - y_{t-13}$ . Does this mean that  $y_{t+1} - y_{t-11} = y_t - y_{t-12}$ ? Yes, this means that the forecasting origin will determine all the future seasonal differences.

Let us return to modelling seasonal time series. The above explanations implied that in order to fit a seasonal time series with an ARIMA model, it is not enough to just have a model of order  $(p,d,q)$ . We also need a seasonal order  $(P,D,Q)$ , which will be combined with these non-seasonal coefficients  $(p,d,q)$ . The general formula is

$$\text{ARIMA}(p,d,q)(P,D,Q)_s$$

How do we combine the two? We can use for example, an  $\text{SARIMA}(1,1,1)(1,1,1)_4$ , i.e. a model with  $s = 4$ . This model is described as:

$$(1-\phi_1 B)(1-\Phi_1 B^4)(1-B)(1-B^4)y_t = (1-\theta_1 B)(1-\Theta_1 B^4)e_t$$

Where  $\phi$  and  $\theta$  are the ordinary ARMA coefficients,  $\Phi$  and  $\Theta$  are the seasonal ARMA coefficients and  $B$  is the backshift operator. If we unravel the above equation, we get:

$$y_t = (1+\phi_1)y_{t-1} - \phi_1 y_{t-2} + (1+\phi_1)y_{t-4} - (1+\phi_1+\Phi_1+\phi_1\Phi_1)y_{t-5} + (\phi_1+\phi_1\Phi_1)y_{t-6} - \Phi_1 y_{t-8} + \\ + (\Phi_1+\phi_1\Phi_1)y_{t-9} - \phi_1\Phi_1 y_{t-10} + e_t - \theta_1 e_{t-1} - \Theta_1 e_{t-4} + \theta_1\Theta_1 e_{t-5}$$

As we can see, it is quite a long and messy equation. We should use abbreviated notation instead as it makes a lot of sense to use it. So, a seasonal  $\text{ARIMA}(p,d,q)(P,D,Q)_s$  model can be written in a short general form as:

$$(1-\phi_1 B)(1-\Phi_1 B^s)y_t = (1-\theta_1 B)(1-\Theta_1 B^s)e_t$$

which is much more elegant. An  $\text{ARIMA}(2,1,0)(0,1,0)_{12}$  model, for example, is therefore written as:

$$(1-\phi_1 B-\phi_2 B^2)(1-\Phi_1 B^{12})(1-B)y_t = e_t$$

Where,  $(1-\phi_1B-\phi_2B^2)$  represents a non-seasonal AR(2) part of the model,  $(1-\Phi_1B^{12})$  represents the seasonal AR(1) part and  $(1-B)$  are non-seasonal differences.

The seasonal parameters and coefficients are  $\Phi, \Theta, P, D, Q$  and  $\phi, \theta, p, d, q$  are for the non seasonal time series.  $s$  denotes the seasonality.

Following the three steps of the Box and Jenkins methodology (Box and Jenkins 1976), *identification, estimation and diagnostics checking* the SARIMA models are fitted to stationary or weakly stationary time-series data. The estimation of the AR ( $p, P$ ) and MA ( $q, Q$ ) parameters for fitting a SARIMA model is about the same when you model an Arima model.

*For example a SARIMA (1,0,0)(0,1,1)<sub>12</sub> model, in short form is*

$$(1-\phi_1B)(1-B^{12})y_t = (1-\Theta_1B^{12})e_t$$

which leads to

$$y_t = \phi_1 y_{t-1} + y_{t-12} - \phi_1 y_{t-12} + e_t - \Theta_1 e_{t-12}$$

*For example a SARIMA (0,1,1)(0,1,1)<sub>4</sub> model, in short form is*

$$(1-B)(1-B^4)y_t = (1-\theta_1B)(1-\Theta_1B^4)e_t$$

which leads to

$$y_t = y_{t-1} + y_{t-4} - y_{t-5} + \theta_1 e_{t-1} - \Theta_1 e_{t-4} + \theta_1 \Theta_1 e_{t-5}$$

What should we expect from the autocorrelation and partial autocorrelation functions for these models? In many ways they are identical, in terms of inference, as non-seasonal models. An ARIMA(0,0,0)(1,0,0)<sub>12</sub> model, for example, will have one significant partial autocorrelation at lag 12 and the autocorrelations will decay exponentially for all seasonal lags, i.e. 12, 24, 36, etc. An ARIMA(0,0,0)(0,0,1)<sub>12</sub> model, on the other hand, will have one significant autocorrelation at lag 12 and the partial autocorrelations will decay exponentially for all seasonal lags.

The principles of parameter estimation for seasonal models are the same as for the non-seasonal models, although the equations for SARIMA can be messier.

Please be aware that it is impractical and unnecessary to do seasonal differencing of the series twice. It is good practice not to difference the time series more than twice, regardless of what kind of differencing is used, i.e. use a maximum of one seasonal and one ordinary differencing or, at most, do ordinary differencing twice. One of the most popular and frequently used seasonal models in practice is an ARIMA(0,1,1)(0,1,1)<sub>s</sub>. Most time series can be fitted with an ARIMA(0,1,1)(0,1,1)<sub>s</sub>, so do not exaggerate.

## Conclusions:

ARIMA model offers a good technique for predicting the magnitude of any variable. Its strength lies in the fact that the method is suitable for any time series with any pattern of change and it does not require the forecaster to choose a priori the value of any parameter.

ARIMA models also provide useful tools for interested parties to be used as a point of reference to the performance of other forecasting models like neural network, kernel regression and so on. However, please bear in mind that the forecasting inaccuracy increased the farther away the forecast is from the used data, which is consistent with the expectation of ARIMA models. It takes lots of practice and experimenting. Hopefully with all the examples presented in this Chapter can speed up and shorten your learning curve.

## Chapter 3

### Monte Carlo Simulation With MS Excel

#### Introduction

Monte Carlo simulation is a widely used method for solving complex problems using computer algorithms to simulate the variables in the problem. Typically an algorithm is developed to "model" the problem, and then the algorithm is run many times (from a few hundred up to millions) in order to develop a statistical data set to study how the model behaves. This statistical data simulated can be represented as probability distributions (or histograms) or converted to *error bars*, *reliability predictions*, *tolerance zones*, and *confidence intervals*.

The Monte Carlo method is also one of many methods for analyzing **uncertainty propagation**, where the goal is to determine how *random variation*, *lack of knowledge*, or *error* affects the *sensitivity*, *performance*, or *reliability* of the system that is being modeled. Monte Carlo simulation is categorized as a **sampling method** because the inputs are randomly generated from *probability distributions* to simulate the process of sampling from an actual *population*. So, we try to choose a distribution for the inputs that most closely *matches data we already have*, or best represents our *current state of knowledge*.

For example, consider the basic coin toss where we have two possible outcomes (heads or tails), each with a 50% probability. In a million coin tosses, roughly half will be "heads" and half will be "tails". No complex math is required to know this. A simple Monte Carlo simulation would prove the same result. If you were to develop a spreadsheet with a random number generator resulting in 0 for "heads" and 1 for "tails" (which we will do later in this chapter), then make the spreadsheet to recalculate a million times, each time recording the results in a database, you could then run a report on the database which will show that very close to 50% of the recalculations resulted in 0 or "heads" and the other 50% in 1 or "tails".

In a more complicated coin tossing simulation, suppose you want to know the likelihood of getting "heads" 7 times out of 10 coin tosses. Again here, statistical mathematical equations provide an accurate answer without the need for Monte Carlo. A Monte Carlo spreadsheet can simulate a series of ten coin tosses as easy as a single coin toss, and can keep a record of how many times 7 "heads" were returned from 10 tosses after running a few thousand iterations of the model. The resulting data will have an answer very close to the mathematical statistical probability of getting 7 "heads".

Although we can represent and solve this problem using mathematical equations, a problem with this degree of complexity becomes more efficient to solve using a Monte Carlo simulation.

### **How the Monte Carlo Algorithm works?**

Monte Carlo Algorithm works based on the **Law of Large Numbers**. It says that if you generate *large* number of samples, eventually you will get the *approximate* desire distribution.

- To use Monte Carlo Simulation is quite simple and straightforward as long as the convergence can be guaranteed by the theory.
- Monte Carlo Simulation may provide statistical sampling for numerical experiment using computer
- For optimization problems, Monte Carlo Simulation often can reach global optimum and overcome local extreme
- Monte Carlo Simulation provides approximate solution to many mathematical problems
- The method can be used for **both** stochastic (involve probability) and deterministic (without probability).

That's essentially all there is to it. The more complex the problem, the better it will be to use Monte Carlo simulations over mathematical solving. Most of the times the problem may involve some guessing at how a particular variable behaves (is it a normal distribution curve, poison, or a linear?), but by using Excel spreadsheets to build your models, it's easy to change the assumptions behind each variable to study the sensitivity the results on each of your assumptions. I will use 3 examples to show you how to implement a Monte Carlo simulation on a spreadsheet.

Before we develop the Excel spreadsheet example, let me explain to you the RAND() function which is the most essential building block of Monte Carlo models in Excel. The function is Excel's native random number generator. This function returns an evenly distributed random real number greater than or equal to 0 and less than 1. A new random real number is returned every time the worksheet is calculated. To use the RAND() function, simply enter "=RAND()" in the spreadsheet cell. Each time the spreadsheet is updated or recalculated a new random number will be generated.

**Note: The results in this book and the example spreadsheets can be different due to Excel recalculation.**

### **Example 1: Coin tossing**

Developing Monte Carlo models require one to translate a real-world problem into Excel equations. This is a skill you can develop over time. The examples later in this chapter will give you a good start.

As a first example, look at the case of a single coin toss. In the real world we think in terms of "heads" and "tails", but for data analysis it will be more efficient to represent these outcomes to 0 and 1. Next you have to figure out how to convert the output generated by the RAND() function into the output of a coin toss -- i.e., take evenly distributed randomly generated numbers greater than or equal to 0 and less than 1, and translate them into two single outcomes - 0 or 1 - with a 50% probability of each.

As your model variables increase in complexity, however, it's important to master the art of RAND() manipulation. For the coin toss, a simple approach would be "=ROUND(RAND(),0)". Another approach that works equally well is "=INT(RAND()\*2)".

### Building a worksheet-based simple coin toss Monte Carlo simulation

As I have covered the basic of Monte Carlo concepts, it's time to build the first Monte Carlo simulation worksheet.

We will simulate a single coin toss. In order to determine the probability of the coin landing heads-up or tails-up, we will repeat the simple coin toss many times, then calculate the percentage of those tosses that yield heads. Open the workbook (Monte Carlo) in folder Chapter 3 and goto *Worksheet (Coin 1)*

First we enter the formula simulating a single toss, "=INT(RAND()\*2)", as illustrated in cell A4 below.

A4		=INT(RAND()*2)		
	A	B	C	
1	<b>Example 1: Coin Tossing</b>			
2				
3	Results			
4	0			
5				

Figure 3.1

Next, we copy cell A4 one thousand times, filling cells A4 through A1003. To do this, make sure cell A4 is highlighted, then just fill down until the selected region reaches cell A1003.

Now we'll calculate the results. In cell E2, enter the formula "=AVERAGE(A4:A1003)". Format the cell to show results as a percentage. The result should be somewhere near 50%. (see Figure 3.2)



E2					<b>f<sub>x</sub></b> =AVERAGE(A4:A1003)
	A	B	C	D	E
1	<b>Example 1: Coin Tossing</b>				
2				<b>Average:</b>	50.70%
3	Results				
4	0				
5	0				
6	0				
7	1				

Figure 3.2

So we have completed a Monte Carlo simulation in its simplest form. The "Average" function tells you that approximately 50% of the coin tosses will result in "heads".

### How many iterations are enough?

In this example, you probably noticed the answer didn't come out to exactly 50%, even though 50% is the statistically correct answer.

Every time you recalculate the spreadsheet by pressing the F9 key, you will see the average change with each recalculation. You may see instances where the average is as low as 46% or as high as 54%. But by increasing the number of iterations in your model, you will increase the accuracy of your results.

To prove this, we create another 5000 instances of coin tossing. Make a new column B of simulated coin tosses, by pasting & copying " =INT(RAND()\*2)" into 5000 cells. As shown below, the column with 5000 iterations returns an answer much closer to 50% than the column with only 1,000 iterations. (see Figure 3.3)

E3					<b>f<sub>x</sub></b> =AVERAGE(B4:B5003)
	A	B	C	D	E
1	<b>Example 1: Coin Tossing</b>				
2				<b>Average (n= 1000):</b>	47.50%
3	Results (n =1000)	Results (n =5000)		<b>Average (n= 5000):</b>	50.10%
4	1	0			
5	1	1			
6	1	1			
7	0	1			
8	0	1			

Figure 3.3

One method to determine whether you should increase the number of iterations is to recalculate the spreadsheet several times, and observe how much variance there is in the "Average". If you see more variance in the results than you are satisfied with, increase

the number of iterations until this spot-check method gives answers within an acceptably narrow range.

Thus, the number of iterations depends on the complexity of the variables and the required accuracy of the result.

### Building a worksheet-based multiple coin toss Monte Carlo simulation

We will take the simple coin toss simulation a step further.

Suppose we want to know:

1. What is the probability of getting "heads" on seven out of ten tosses, and
2. What is the probability of getting "heads" on at least seven out of ten tosses?

Like the case of the single coin toss above, we can actually solve the problem mathematically rather than through simulation. In this ten coin tosses there are exactly  $2^{10}$ , or 1024, possible outcomes of heads-tails combinations. Of these, exactly 120 have seven "heads", so the probability of getting exactly seven "heads" in ten coin tosses is  $120/1024$ , or 11.7%. Furthermore, 176 of the 1024 combinations have 7, 8, 9, or 10 heads, so the probability of getting heads on at least seven tosses is  $176/1024$ , or 17.2%.

For this example, we use the same formula to simulate a single toss that we used in the previous example. Goto *Worksheet (Coin 2)*. We enter the formula " $=\text{INT}(\text{RAND}()*2)$ " into cells A10 through J10, as shown in Figure 3.4 below.

L10		=SUM(A10:J10)										
	A	B	C	D	E	F	G	H	I	J	K	L
1	<b>Example 1: Multiple coin tossing</b>											
2												
3												
4												
5												
6												
7												
8												
9												
10	0	0	1	0	1	0	0	1	0	1		4
11												

Figure 3.4

Then we sum the results of columns A through J by entering " $=\text{SUM}(A10:J10)$ " in cell L10 as shown above. This represents the number of "heads" that came up in ten random tosses of a coin.

Now make 5,000 copies of row 10 in the rows below it. We fill down from A10:L10 to A5009:L5009. Thus we have a Monte Carlo worksheet-based simulation representing 5,000 iterations of tossing a coin ten times.

To study the results, we make use of the COUNT() and COUNTIF() functions. As indicated below in Fig 3.5, in cell D3 we enter the formula "`=COUNT(L10:L5009)`" to return the exact number of iterations in the model. We enter the formula "`=COUNTIF(L10:L5009,7)`" in cell D4 to count the number of iterations which returned exactly seven heads. Cell E4 returns the percent of all iterations that resulted in seven heads "`=D4/D3`", and in this particular instance the outcome was 11.88% - which is quite close to the expected statistical outcome of 11.7%.

The formula "`=COUNTIF(L15:L5014,">6")`" in cell D5 counts the number of iterations which returned AT LEAST seven heads. E5 shows that result as a percent of all iterations, and again our outcome of 17.88% is also quite close to the statistical probability of 17.2%. (see Figure 3.5 below)

D5 <code>=COUNTIF(L15:L5014,"&gt;6")</code>												
	A	B	C	D	E	F	G	H	I	J	K	L
1	<b>Example 1: Multiple coin tossing</b>											
2												
3	Simulation:			5000								
4	Number of 7:			594	11.88%							
5	Number of 7,8,9,10:			894	17.88%							
6												
7												
8												
9	Toss 1	Toss 2	Toss 3	Toss 4	Toss 5	Toss 6	Toss 7	Toss 8	Toss 9	Toss 10		Headcount
10	0	0	0	1	0	1	1	0	0	1		4
11	0	1	0	1	1	1	0	0	0	0		4
12	1	1	0	0	1	1	0	0	0	0		4
13	1	1	0	1	0	1	0	0	0	0		4
14	1	1	0	1	1	1	1	1	0	0		7
15	1	1	0	0	1	1	0	0	1	0		5

Figure 3.5

With this simple Monte Carlo example, we have now the basic knowledge to build a more complicated model that deals with sales forecasting.

## Example 2: Sales Forecasting

X

X

X

**X**

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

**X**

	A	B	C	D	E
1	<b>Sales Forecasting</b>				
2				Min	Max
3	Fixed Cost	100000		-	-
4	Sales Volume	65340.7		50000	80000
5	Unit Cost	6.04		5	8
6	Selling Price	9.523		8	12
7					

Figure 3.6

I	J	K	L	M
Sales Volume	Unit Cost	Selling Price		Net Profit
58741	6	11		193705
63252	6	11		216260
63301	7	11		153204
74243	7	11		196972
55082	6	11		175410
63364	6	10		153456
62025	5	8		86075
51049	5	8		53147
69487	6	11		247435
64614	6	8		29228
52305	6	9		56915

Figure 3.7

In Column I is the Sales Volume with maximum value 80000 and minimum value 50000. Column J is Unit Cost with maximum value 8 and minimum value 5. And Column K is the Selling Price with maximum value 12 and minimum value 8. (see Figure 3.7)

Column M shows the profit. There are 1000 rows which mean we are simulating 1000 sales scenarios, each row using a different combination of values for the sales volumes, sales price and unit costs. (You can simulate more than 1000 rows if you want)

We enter `=INT($D$4 + RAND()*(E$4-$D$4))` in cell I2 which is

the minimum value of Sales Volume in D4

plus a random value (derive with the Excel formula RAND())

times

(maximum value of Sales Volume in cell E4

minus the minimum value of Sales Volume in D4)

Since we are simulating 1000 scenarios, simply copy the formula down 1000 rows until cell I1001, making sure that you use relative references in the formula (the \$ signs).

We also enter the same basic formula for the Unit Cost and Selling Price. Activate cell J2 and K2 to view the formulas. The general formula to generate random number with an upper and lower bound is

$$r = \text{Int}(\text{minimum} + \text{Rand()} * (\text{maximum} - \text{minimum}))$$

We enter  $=\text{I2}*(\text{K2}-\text{J2})-\$B\$3$  in cell M2 and fill down to M1002 as the profit formula. Here there are 1000 possible profit values. (Note: you can simulate more than 1000 profit values). Because we have used the volatile RAND() formula, to re-run the simulation all we have to do is recalculate the worksheet (press **F9** is the shortcut).

After that, a histogram is build using the simulated profit values to further analyze the data.

### How to create a Histogram in Excel

First, the *Data Analysis ToolPak* must be installed. To do this, pull down the **Tools** menu from the Excel menu, and choose **Add-Ins**. You will see the *Analysis ToolPak* inside the **Add-Ins** dialog box.

To start, you need to have a column of numbers in the spreadsheet that you wish to create the histogram from, AND you need to have a column of intervals or "Bin" to be the upper boundary category labels on the X-axis of the histogram.

Cell O7:O57 is the bin values. Its contain 50 ranges/bins between the maximum profit 100000 and the minimum profit 55000. (see part of the formula below)

	O
1	Histogram Plot
2	Min
3	Max
4	N:
5	
6	<b>Bins</b>
7	=P2
8	=(P\$3-P\$2)/50+O7
9	=(P\$3-P\$2)/50+O8
10	=(P\$3-P\$2)/50+O9
11	=(P\$3-P\$2)/50+O10
12	=(P\$3-P\$2)/50+O11

Figure 3.7a

Pull Down the Tools Menu and Choose **Data Analysis**, and then choose **Histogram** and click **OK**. Enter the **Input Range** as **M2:M1001** and enter the **Bin Range** as **O7:O57**. Choose whether you want the output in a new worksheet ply, or in a defined output range on the same spreadsheet. (see Figure 3.8 below)

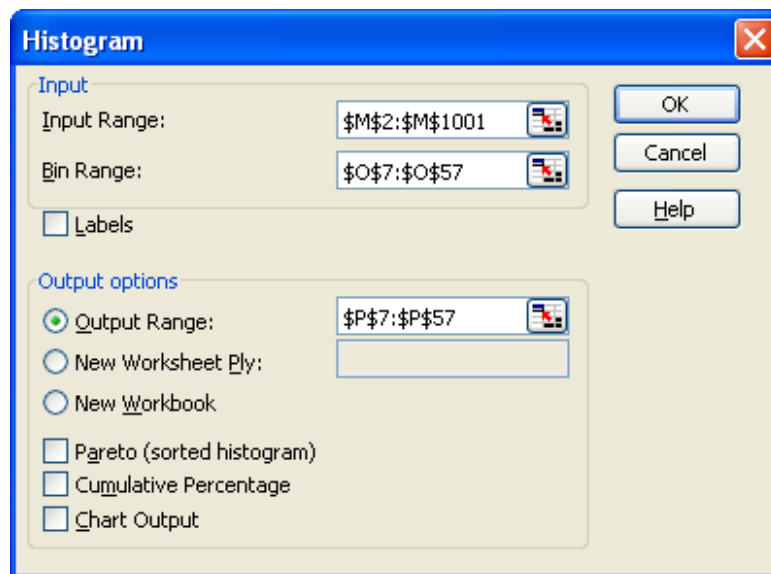


Figure 3.8

If you choose in the example above to have the output range **P7:P57**, and you clicked **OK**, the spreadsheet would look like this:

O	P	Q	R
Bins			
-100000	<i>Bin</i>	<i>Frequency</i>	
-87000	-100000	0	
-74000	-87000	0	
-61000	-74000	0	
-48000	-61000	0	
-35000	-48000	4	0.0040
-22000	-35000	28	0.0320
-9000	-22000	40	0.0720
4000	-9000	7	0.0790
17000	4000	14	0.0930
30000	17000	30	0.1230
43000	30000	34	0.1570
56000	43000	35	0.1920
69000	56000	37	0.2290
82000	69000	52	0.2810
95000	82000	49	0.3300
108000	95000	38	0.3680
121000	108000	54	0.4220
134000	121000	60	0.4820
147000	134000	60	0.5420
160000	147000	53	0.5950
173000	160000	41	0.6360
186000	173000	37	0.6730
199000	186000	35	0.7080
212000	199000	46	0.7540

Figure 3.9

The next step is to make a bar chart of the Frequency column. Select **Q8:Q58** and click on the graph Wizard, and choose Column Graph, click on **Finish**. Delete the Series legend, right click on the edge of the graph and choose **Source Data** , and enter the Bin frequencies (**P8:P58**) for the X-Axis Category labels. (see Figure 3.9a)

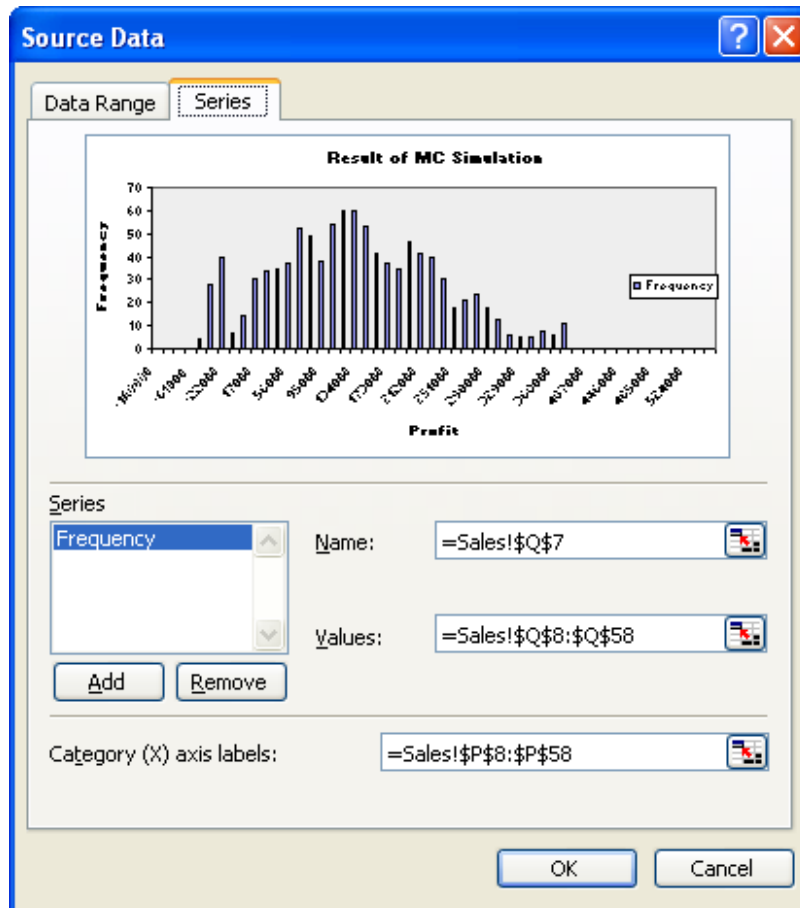
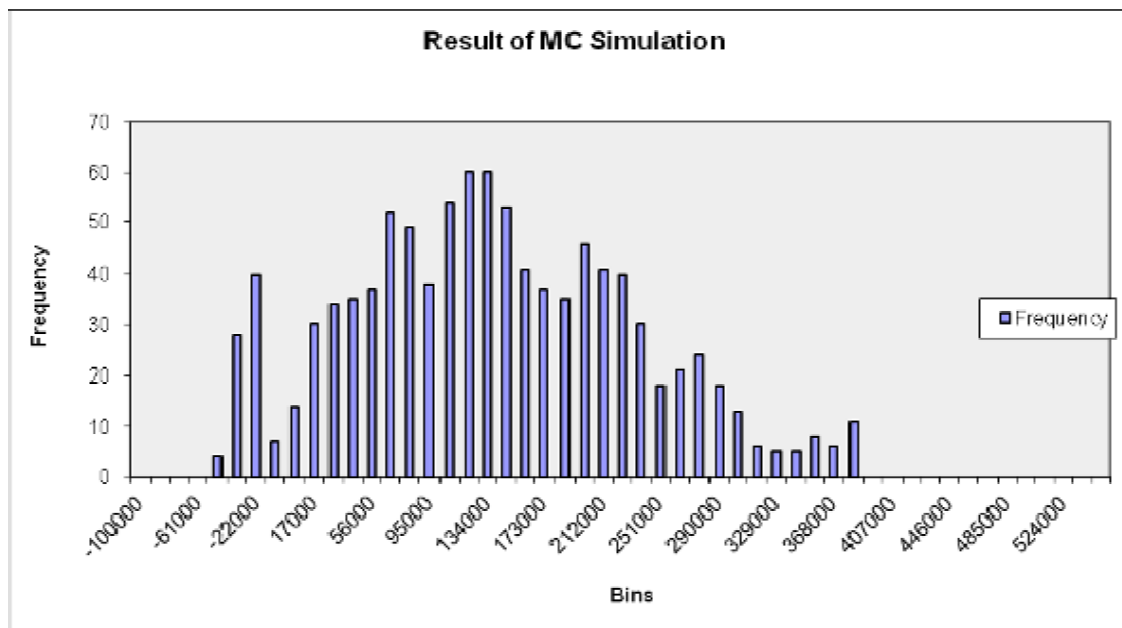


Figure 3.9a

Dress up the graph by right clicking on the edge of the graph and choosing **Chart Options**. Enter a complete descriptive title with data source, perhaps data labels, and axes labels. You may also right click and format the color of the bars and background. The completed Histogram should look something like this:





**Figure 3.10 :** A Histogram in Excel created using a Bar Chart.  
(From a Monte Carlo simulation using  $n = 1000$  points and 50 bins).

After creating the histogram, the next step is to analyze the results visually.

We can glean a lot of information from this histogram:

- It looks like profit will be positive, *most* of the time.
- The uncertainty is quite large, varying between -48000 to 381000.
- The distribution does not look like a perfect Normal distribution.
- There doesn't appear to be outliers, truncation, multiple modes, etc.

The histogram tells a good story, but in many cases, we want to estimate the **probability** of being below or above some value, or between a set of specification limits.

In our Monte Carlo Simulation example, we plotted the results as a histogram in order to **visualize** the uncertainty in profit. In order to provide a concise summary of the results, it is customary to report the **mean, median, standard deviation**, standard error, and a few other summary statistics to describe the resulting distribution. The screenshot below shows these statistics calculated using simple Excel formulas.

**NOTE:** *The results below and in the worksheet can differ from the Histogram in this book because when Excel recalculate by pressing F9, the Histogram that we created will not update automatically according with the recalculated data. We need to rebuild the Histogram again. Basically, I just want to show how to derive the statistics and how to visually interpret the Histogram.*

	A	B
10		
11	<b>Summary Statistics</b>	
12	Sample (n)	1000
13	SampleMean	124095.7
14	Standard Deviation	90649.31
15	Maximum	378776
16	Minimum	-49331
17	Skewness	0.36722
18	Kurtosis	-0.3383
19	1st Quartile	55465
20	3rd Quartile	188103
21	Median	117999

**Figure 3.11** : Summary statistics for the sales forecast example.

As you can see in Figure 3.11, we have:

**B12 = Sample Size (n):** =COUNT(M:M)

**B13 = Sample Mean:** =AVERAGE(M:M)

**B14 = Sample Standard Deviation ( $\sigma$ ):** =STDEV(M:M)

**B15 = Maximum:** =MAX(M:M)

**B16 = Minimum:** =MIN(M:M)

**B20 = Q(.75):** =QUARTILE(M:M,3)

**B19 = Q(.25):** =QUARTILE(M:M,1)

**B17 = Skewness:** =SKEW(M:M)

**B18 = Kurtosis:** =KURT(M:M)

**B21 = Median:** =MEDIAN(M:M)

### Sample Size ( $n$ )

The **sample size**,  $n$ , is the number of **observations** or data points from a single MC simulation. For this example, we obtained  $n = 1000$  simulated observations. Because the Monte Carlo method is **stochastic**, if we repeat the simulation, we will end up calculating a different set of summary statistics. The larger the sample size, the smaller the difference will be between the repeated simulations. (See **standard error** below).

### Central Tendency: Mean and Median

The **sample mean** and **median** statistics describe the **central tendency** or "location" of the distribution. The **arithmetic mean** is simply the **average** value of the observations.

The mean is also known as the "First Moment" of the distribution. In relation to physics, if the probability distribution represented mass, then the mean would be the balancing point, or the center of mass.

If you sort the results from lowest to highest, the **median** is the "middle" value or the 50th Percentile, meaning that **50%** of the results from the simulation are less than the median. If there is an even number of data points, then the median is the average of the middle two points.

### ***Spread: Standard Deviation, Range, Quartiles***

The **standard deviation** and **range** describe the **spread** of the data or observations. The standard deviation is calculated using the **STDEV** function in Excel.

The **range** is also a helpful statistic, and it is simply the maximum value minus the minimum value. Extreme values have a large effect on the range, so another measure of spread is something called the **Interquartile Range**.

The **Interquartile Range** represents the central 50% of the data. If you sorted the data from lowest to highest, and divided the data points into 4 sets, you would have 4 **Quartiles**:

**Q0** is the Minimum value: **=QUARTILE(M:M,0)** or just **=MIN(M:M)**,

**Q1** or Q(0.25) is the First quartile or 25th percentile: **=QUARTILE(M:M,1)**,

**Q2** or Q(0.5) is the Median value or 50th percentile: **=QUARTILE(M:M,2)** or **=MEDIAN(G:G)**,

**Q3** or Q(0.75) is the Third quartile or 75th percentile: **=QUARTILE(M:M,3)**,

**Q4** is the Maximum value: **=QUARTILE(M:M,4)** or just **MAX(G:G)**.

In Excel, the Interquartile Range is calculated as Q3-Q1 or:

**=QUARTILE(M:M,3)-QUARTILE(M:M,1)**

### ***Shape: Skewness and Kurtosis***

#### **Skewness**

**Skewness** describes the **asymmetry** of the distribution relative to the mean. A **positive** skewness indicates that the distribution has a longer right-hand tail (skewed towards more positive values). A **negative** skewness indicates that the distribution is skewed to the left.

#### **Kurtosis**

**Kurtosis** describes the **peakedness** or **flatness** of a distribution relative to the Normal distribution. **Positive** kurtosis indicates a more peaked distribution. **Negative** kurtosis indicates a **flatter** distribution.

### **Confidence Intervals for the True Population Mean**

The **sample mean** is just an estimate of the **true population mean**. How accurate is the estimate? You can see by repeating the simulation (pressing F9 in this Excel example) that the mean is not the same for each simulation.

## Standard Error

If you repeated the Monte Carlo simulation and recorded the sample mean each time, the distribution of the sample mean would end up following a Normal distribution (based upon the Central Limit Theorem). The **standard error** is a good estimate of the **standard deviation** of this distribution, assuming that the sample is sufficiently large ( $n \geq 30$ ).

The **standard error** is calculated using the following formula:

$$StErr = \frac{s}{\sqrt{n}}$$

In Excel: =STDEV(M:M)/SQRT(COUNT(M:M))

## 95% Confidence Interval

The standard error can be used to calculate **confidence intervals for the true population mean**. For a 95% **2-sided** confidence interval, the Upper Confidence Limit (UCL) and Lower Confidence Limit (LCL) are calculated as:

$$95\%UCL = Mean + 1.96 \times StErr = \bar{y} + 1.96 \frac{s}{\sqrt{n}}$$
$$95\%LCL = Mean - 1.96 \times StErr = \bar{y} - 1.96 \frac{s}{\sqrt{n}}$$

To get a 90% or 99% confidence interval, you would change the value 1.96 to 1.645 or 2.575, respectively. The value 1.96 represents the **97.5** percentile of the standard normal distribution. (You may often see this number rounded to 2). To calculate a different percentile of the standard normal distribution, you can use the NORMSINV() function in Excel.

**Example:**  $1.96 = \text{NORMSINV}(1-(1-.95)/2)$

### Note:

Keep in mind that confidence intervals make no sense (except to statisticians), but they tend to make people feel good. The correct interpretation: "**We can be 95% confident that the true mean of the population falls somewhere between the lower and upper limits.**" What population? The population we artificially created! Lest we forget, the results depend completely on the assumptions that we made in creating the model and choosing input distributions. "Garbage in ... Garbage out ..." So, I generally just stick to using the standard error as a measure of the uncertainty in the mean. Since I tend to use Monte Carlo simulation for **prediction purposes**, I often don't even worry about the mean. I am more concerned with the overall uncertainty (i.e. the spread).

As a final step in the sales forecast example, we are going to look at how to use the Excel **percentile function** and **percent rank function** to estimate important summary statistics from our Monte Carlo simulation results.

### Percentile and PercentRank Functions

Excel's PERCENTILE() and PercentRank function are useful in many Monte Carlo models, and is particularly useful in answering our questions in regards to forecasting profit . For example:

**Question 1:** *What percentage of the results was less than -\$22000?*

This question is answered using the percent rank function: **=PERCENTRANK(array,x,significant\_digits)**, where the **array** is the data range and **x** is -\$22000.

You can read more about the details of the RANK, PERCENTILE, and PERCENTRANK functions in the Excel help file (F1).

The Figure 3.14 below shows a screen shot of some examples where the percent rank function is used to **estimate** the cumulative probability based upon results of the Monte Carlo simulation. (Select the cells in the worksheet to see formula entered). So we have 6.87% of the result that are below -22000. And we know that 60.58% of the profit result is more than 100000.

B25		<b>f<sub>x</sub></b>	<b>=PERCENTRANK(M:M,-22000,4)</b>	
	A	B	C	D
22				
23	<b>Profit Probabilities</b>			
24	Profit > 0	92.50%		
25	Profit < -22000	6.87%		
26	Profit > 100000	60.58%		
27	-22000 < Profit < 10000	32.55%		

**Figure 3.14 :** Calculating probabilities using the Excel percent rank function.

The **accuracy** of the result will depend upon the **number of data points** and how far out on the **tails** of the distribution you are (and of course on how realistic the model is, how well the input distributions represent the true uncertainty or variation, and how good the random number generator is). Recalculating the spreadsheet a few times by pressing **F9** will give you an idea of how much the result may vary between each simulation.

**Question 2:** *What are the 95% central interval limits?*

Stated another way: **What are the 0.005 and 0.95 quantiles?**

This is probably one of the most important questions, since the answer provides an important summary statistic that describes the spread of the data. The **central interval** is

found by calculating the **0.05 and 0.95 quantiles**, or  $Q(\alpha/2)$  and  $Q(1-\alpha/2)$ , respectively.

Percentile is a measure that locates where a value stands in a data set. The kth percentile divides the data so that at least p percent are of this value or less and (100-p) percent are this value or more. If you have a set of data and need to find the value at a certain percentile, you use the PERCENTILE function in Excel.

The quantiles (or percentiles) are calculated by using the Excel percentile function: **=PERCENTILE(array,p)** where the **array** is the data range (column M) and **p** is the cumulative probability (0.05 or 0.95).

The figure below shows a screen shot of examples that use the percentile function in the Monte Carlo simulation example spreadsheet.

	A	B	C
28			
29	Percentiles		
30	Q(0.05)	-35006	
31	Q(0.95)	287202	

**Figure 3.15 :** Calculating quantiles using the Excel percentile function.

Note that we are **not** using the term "*confidence interval*" to describe this interval. We are estimating what *proportion* of the data we expect to be within the given limits based upon the results of the simulation. We call it a *central interval* because we are defining the interval based upon the central proportion of the data.

**NOTE:** *The results above and in the worksheet may differ from the Histogram in this book because when Excel recalculate by pressing F9, the result that we created will update accordingly with the recalculated data. Basically, I just want to show how to derive the statistics and how to visually interpret the Histogram.*

### ***A Few Things to Keep in Mind***

Beware that defining the uncertainty of an input value by a probability distribution that does not correspond to the real one and sampling from it will give incorrect results. In addition, the assumption that the input variables are independent might not be valid. Misleading results might come from inputs that are mutually exclusive or if significant correlation is found between two or more input distributions. Also note that the number of trials should not be too small, as it might not be sufficient to simulate the model, causing clustering of values to occur.

There you go. This example shows you how to use Monte Carlo simulation for sales forecast and planning. This example is not comprehensive, and there are many other factors that affect sales that have not been covered. However, I hope this model has given you a good introduction to the basics.

### Example 3: Modeling Stock Prices

Prior to the 1960's, most investors believed that future securities prices could be predicted (and that great riches were to be had) if only they could discover the secret. Many investors still believe this today, despite much evidence that suggests that they would be best served by simply owning the entire market (investing in index funds) rather than trying to pick individual stocks.

The efficient markets hypothesis (EMH) essentially states that techniques such as fundamental and technical analysis cannot be used to consistently earn excess profits in the long run. The EMH began with the observation that changes in securities prices appear to follow a random walk (technically, geometric Brownian motion with a positive drift). The random walk hypothesis was first proposed by mathematician Louis Bachelier in his doctoral thesis in 1900, and then promptly forgotten.

One of the best-known stories regarding the randomness of changes in stock prices is told by Burton Malkiel in *A Random Walk Down Wall Street* (a fascinating practitioner-oriented book now in its ninth edition). In that book he tells of having fooled a friend, who was a committed technical analyst, by showing him a "stock chart" that was generated by coin tosses, rather than actual stock prices. Apparently, the friend was quite interested in learning the name of the stock.

The purpose of this example is not to debate market efficiency, or to even state that the EMH is correct. Instead, I want to demonstrate how we can simulate stock prices of the kind that Malkiel discussed.

#### The Geometric Brownian Motion Process for Stock Prices

We assume that stock prices follow a (continuous time) geometric Brownian motion process:

$$dS = \mu S dt + \sigma S dz$$

where,

$S$  = the current stock price

$\mu$  = the expected stock return

$\sigma$  = the stock return volatility

$dz = \sum (dt)^{0.5}$ ,  $\sum$  is a standard random variable:  $\sum \in (0,1)$

The discrete time equivalent is:

$$\Delta S = \mu S \Delta t + \sigma S \varepsilon \sqrt{\Delta t}$$

$\Delta$  = change

$\Delta t$  = change in period

Don't worry about the formula and symbols here. They are easily implemented in Excel which is shown in the *Worksheet (Stock Price)*.

Goto *Worksheet (Stock Price)*. For our simulation, we will use the closing price of Google from 19 August 2004 to 6 June 2008. These data are entered in column A52:B1008. First of all, we calculate the percentage of daily return of the closing price. We enter the formula (today closing price – yesterday closing price)/ today closing price i.e C52 = (B52-B53) and fill down the formula until cell C1007. Column D shows the percentages of return. (see Figure 3.16)

	A	B	C	D
50				
51	Date	Adj Close	Daily Return	% Daily Return
52	6/6/2008	567	-19.3	-3.40%
53	5/6/2008	586.3	14.08	2.40%
54	4/6/2008	572.22	4.92	0.86%
55	3/6/2008	567.3	-7.7	-1.36%
56	2/6/2008	575	-10.8	-1.88%
57	30/5/2008	585.8	2.8	0.48%
58	29/5/2008	583	14.76	2.53%
59	28/5/2008	568.24	7.34	1.29%
60	27/5/2008	560.9	16.28	2.90%
61	23/5/2008	544.62	-4.84	-0.89%
62	22/5/2008	549.46	-0.53	-0.10%
63	21/5/2008	549.99	-28.61	-5.20%

Figure 3.16

Next we calculate the average and standard deviation of these percentages of daily returns. These are entered in cells B19 and B20 respectively.

	A	B	C
16			
17	Stock price to t=0	\$ 567.00	
18	Time (T)	1 year(s)	
19	Expected return ( $\mu$ )	0.16%	
20	Volatility ( $\sigma$ )	2.22%	
21	$\Delta T$	0.0033	

Figure 3.16a

The idea is using Monte Carlo simulation to resample or replicate the closing prices. We enter \$567 the closing price on 6/6/2008 in cell B17. We will use Monte Carlo to simulate 300 days into the future.



Before I go further, let me explain to you, the Excel function **=NORMINV()**

Excel does not provide a random number generator which directly produces samples from a general normal distribution. Instead it is necessary to first generate a uniform random number from the interval zero to one and then use this number to create a sample from the desired normal distribution. The good news is that Excel makes it very easy to perform each of these separate tasks.

The typical Excel formula is:

**=NORMINV(RAND(), average, standard deviation)**

Applied through many iterations, this formula will yield the normal distribution of data values described by the specified mean and standard deviation.

The RAND function produces the zero-one random number. The NORMINV function converts this into the sample from the normal distribution. The second argument to the NORMINV function is the mean of the distribution, in our case we enter the percentage of average daily return. The third argument is the standard deviation of the normal distribution, i.e. we enter the percentage of the standard deviation of the daily return.

Starting from range L6:Q6 we enter the closing price on 6/6/2008 at period 0. This is =\$B\$17. (see Figure 3.16b below)

L6			$\text{fx}$ =\$B\$17					
	J	K	L	M	N	O	P	Q
3								
4			S -1	S -2	S -3	S -4	S -5	S -6
5	Period	Time	S + ΔS	S + ΔS	S + ΔS	S + ΔS	S + ΔS	S + ΔS
6	0	0	\$ 567.00	\$ 567.00	\$ 567.00	\$ 567.00	\$ 567.00	\$ 567.00
7	1	0.003333	\$ 566.72	\$ 566.27	\$ 567.35	\$ 567.50	\$ 566.62	\$ 567.58
8	2	0.006667	\$ 567.07	\$ 566.35	\$ 567.07	\$ 567.14	\$ 566.89	\$ 568.41

Figure 3.16b

In this example, we are simulating 6 different stock price paths. You can simulate more stock price paths if you want. In cell L7 i.e period 1, we enter the formula (see Figure 3.17)

$=+\$B\$19*L6*(\$B\$21)+\$B\$20*L6*NORMINV(RAND(),0,1)*(\$B\$21)^{0.5}+L6$ .

This is actually how  $\Delta S = \mu S \Delta t + \sigma S \varepsilon \sqrt{\Delta t}$  is entered as Excel formula.

$\mu$  = cell B19 => mean/average

times

$S$  = cell L6 => stock price at previous period

times

X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> more  
information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

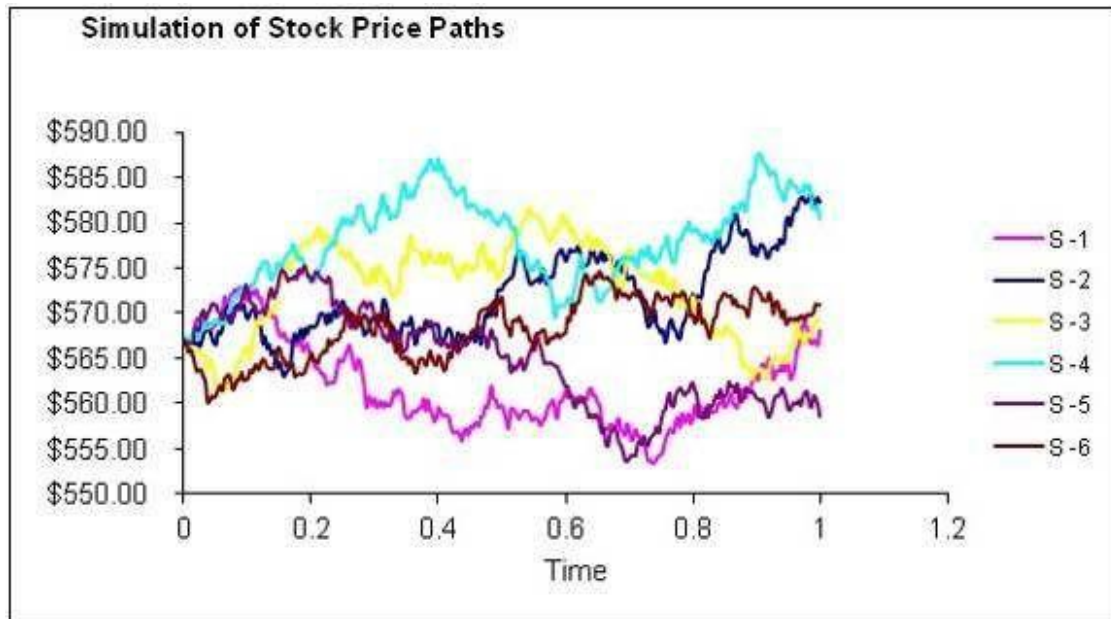


Figure 3.18

By repeating the path simulation, you can obtain a complete distribution of stock prices at the end of 300 days. You can see the 6 stock price paths graphically in Fig 3.18 above.

Press F9 to recalculate the worksheet. You can use the simulated stock price paths to test your stock trading strategy. Things like maximum drawdown, entry and exit strategy, value at risk calculation etc can be tested against these stock price paths. Thus you can optimize the best trading method for maximum return.

In practice, it is usually more convenient to buy an add-on for Excel than to do a Monte Carlo analysis from scratch every time. But not everyone has the money to spend, and hopefully the skills you have learned from these examples will aid in future data analysis and modeling.

## Conclusion

The Monte Carlo Simulation technique is straightforward and flexible. It cannot wipe out uncertainty and risk, but it can make them easier to understand by ascribing probabilistic characteristics to the inputs and outputs of a model. It can be very useful for determining different risks and factors that affect forecasted variables and, therefore, it can lead to more accurate predictions.

## Chapter 4:

### K Nearest Neighbors

#### Introduction

The K Nearest Neighbor or KNN prediction technique is among the oldest techniques used in data mining. Most people have an intuition that they understand what clustering is - namely that like records are grouped or clustered together. Nearest neighbor is a prediction technique that is quite similar to clustering - its essence is that in order to predict what a prediction value is in one record look for records with similar predictor values in the historical database and use the prediction value from the record that it “nearest” to the unclassified record. KNN is also part of supervised learning that has been used in many applications in the field of data mining, statistical pattern recognition, image processing and many others. Some successful applications include recognition of handwriting, satellite image and EKG pattern. Instead of using sophisticated software or any programming language, I will build 3 examples using only spreadsheet functions of Microsoft Excel. These examples include

- using KNN for classification
- using KNN for time series prediction
- Cross validation method

#### A simple explanation of nearest neighbor

A simple understanding of the nearest neighbor prediction algorithm is that if you look at the people in your neighborhood (in this case those people that are in fact geographically near to you). You may notice that, in general, you all have somewhat similar incomes. Thus if your neighbor has an income greater than \$150,000 chances are good that you too have a high income. Certainly the chances that you have a high income are greater when all of your neighbors have incomes over \$150,000 than if all of your neighbors have incomes of \$25,000. Within your neighborhood there may still be a wide variety of incomes possible among even your “closest” neighbors but if you had to predict someone’s income based on only knowing their neighbors you’re best chance of being right would be to predict the incomes of the neighbors who live closest to the unknown person.

The nearest neighbor prediction algorithm works in very much the same way except that “nearness” in a database may consist of a variety of factors not just where the person lives. It may, for instance, be far more important to know which school someone attended and what degree they attained when predicting income. The better definition of “near” might in fact be other people that you graduated from college with rather than the people that you live next to.

Nearest Neighbor techniques are among the easiest to use and understand because they work in a way similar to the way that people think - by detecting closely matching examples. They also perform quite well in terms of automation, as many of the algorithms are robust with respect to dirty data and missing data.

### **How to use Nearest Neighbor for Prediction**

One of the essential elements underlying the concept of nearest neighbor is that one particular object (whether they be cars, food or customers) can be closer to another object than can some other third object. It is interesting that most people have an innate sense of ordering placed on a variety of different objects. Most people would agree that an apple is closer to an orange than it is to a tomato and that a Toyota Vios is closer to a Honda Civic than to a Ferrari. This sense of ordering on many different objects helps us place them in time and space and to make sense of the world. It is what allows us to build clusters/neighbors - both in databases on computers as well as in our daily lives. This definition of nearness that seems to be ubiquitous also allows us to make predictions.

The nearest neighbor prediction algorithm simply stated is:

*Objects that are “near” to each other will have similar prediction values as well. Thus if you know the prediction value of one of the objects you can predict it for it's nearest neighbors.*

### **Where has the nearest neighbor technique been used in business?**

One of the classical places that nearest neighbor has been used for prediction has been in text retrieval. The problem to be solved in text retrieval is one where the end user defines a document (e.g. Wall Street Journal article, technical conference paper etc.) that is interesting to them and they solicit the system to “find more documents like this one”. Effectively defining a target of: “this is the interesting document” or “this is not interesting”. The prediction problem is that only a very few of the documents in the database actually have values for this prediction field (namely only the documents that the reader has had a chance to look at so far). The nearest neighbor technique is used to find other documents that share important characteristics with those documents that have been marked as interesting. Thus we use the k-nn method to classify data. This is only one example classification. K-nn method is applied in many other areas as well like recognition of handwriting, satellite image, EKG pattern, stock selection, DNA sequencing and etc. Let us study K-nearest neighbor algorithm for classification in details.

#### **a) Using KNN for classification (Example 1)**

K-nearest neighbor is a supervised learning algorithm where the result of new instance query is classified based on majority of K-nearest neighbor category. The purpose of this algorithm is to classify a new object based on attributes and training samples. The classifiers do not use any model to fit and only based on memory. Given a query point, we find K number of objects or (training points) closest to the query point. The

classification is using majority vote among the classification of the  $K$  objects. Any ties can be broken at random.  $K$  Nearest neighbor algorithm used neighborhood classification as the prediction value of the new query instance.

Here is step by step on how to compute  $K$ -nearest neighbors (KNN) algorithm:

- i. Determine parameter  $K$  = number of nearest neighbors
- ii. Calculate the distance between the query-instance and all the training samples
- iii. Determine nearest neighbors based on the  $K$ -th minimum distance
- iv. Gather the category  $Y$  of the nearest neighbors
- v. Use simple majority of the category of nearest neighbors as the prediction value of the query instance Or predict the mean, for numeric prediction

i) *What value to use for  $k$ ?* Determine parameter  $K$  = number of nearest neighbors

It depends on dataset size. Large data set need a higher  $k$ , whereas a high  $k$  for a small dataset might cross out of the class boundaries. Calculate accuracy on test set for increasing value of  $k$ , and use a hill climbing algorithm to find the best. Typically use an odd number to help avoid ties. Selecting the  $k$  value is quite intuitive. You can choose heuristically the optimal  $k$  nearest neighbor based on Mean Squared Error done by a cross validation technique on a test set. (I will show you this technique in example (3))

Below are 2 graphic examples to determine  $k$ :

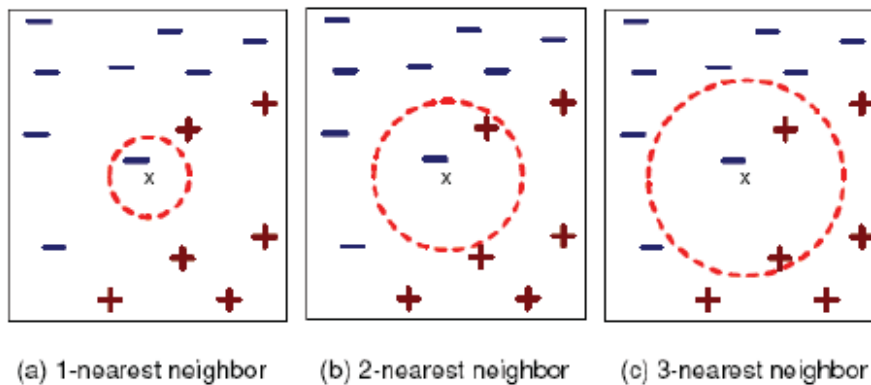
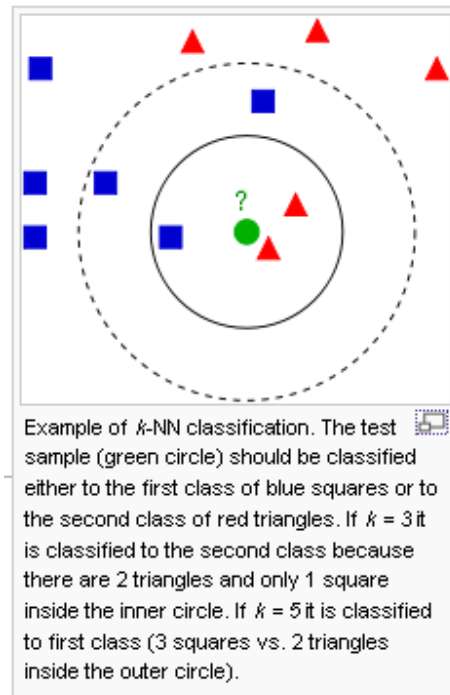


Figure 4.1

$K$ -nearest neighbors of a record  $\mathbf{x}$  are data points that have the  $k$  smallest distance to  $\mathbf{x}$ .



(source: Wikipedia)

Figure 4.2

ii) Calculate the distance between the query-instance and all the training samples

Open the workbook (KNN) in folder Chapter 4 and goto *Worksheet (Home)*. We have data from a bank record and objective testing with two attributes (income and lot's size) to classify the home ownership of a customer. Our objective is to determine whether a customer is entitled a lower interest rate for the second home loan. Here are the training samples

	A	B	C	D	E	F
1	Observation	Income (\$000's)	Lot Size (000's sq. ft.)	Owners = 1, Non-owners = 0	inc_std	lot_std
2	1	61.2	17.3	1	-0.373	-0.720
3	2	86	17	1	0.880	-0.853
4	3	65.5	21.3	1	-0.156	1.048
5	4	61	21	1	-0.383	0.915
6	5	88	22	1	0.982	1.358
7	6	109.8	20.2	1	2.083	0.562
8	7	107.9	17.7	1	1.987	-0.543
9	8	83	21.5	1	0.729	1.136
10	9	68.5	20.5	1	-0.004	0.694
11	10	92.7	21	1	1.219	0.915
12	11	50.5	22.3	1	-0.914	1.490
13	12	81.5	19.5	1	0.653	0.252
14	13	74.5	19.3	0	0.299	0.164
15	14	53	20.5	0	-0.788	0.694
16	15	65.1	17.5	0	-0.176	-0.632
17	16	43.5	21	0	-1.268	0.915
18	17	84.2	18	0	0.789	-0.411
19	18	49.8	17.5	0	-0.949	-0.632
20	19	60	16.2	0	-0.434	-1.206
21	20	66.3	18.5	0	-0.115	-0.190
22	21	48	16.2	0	-1.040	-1.206
23	22	32.5	19	0	-1.824	0.031
24	23	50.8	14.3	0	-0.899	-2.046
25	24	62.7	15	0	-0.297	-1.737
26						
27	Obs	60	20	?	-0.43388	0.473388
28						
29	MEAN	68.58	18.93			
30	STD	19.78	2.26			

Figure 4.3

First we need to scale and standardize the raw data so that they will have same dimensionality and range. We use the formula

$$(\text{Raw data} - \text{Mean of Raw Data}) / \text{Standard Deviation of Raw Data}$$

Range E2:F27 is the transform data. (see Figure 4.3 above). Select respective cells to view formula entered to do the transformation.

Now there is a customer with yearly income of USD60000 and live in a 20000 square feet apartment. How do we know whether the customer live in his/her own home without asking him/her. Fortunately, k nearest neighbor (KNN) algorithm can help you to predict this type of problem.

The data for this example consist of 2 multivariate attributes namely  $\mathbf{x}_i$  (income and lot's size) that will be used to classify  $\mathbf{y}$  (home ownership). The data of KNN can be any measurement scale from ordinal, nominal, to quantitative scale but for the moment let us deal with only quantitative  $\mathbf{x}_i$  and binary (nominal)  $\mathbf{y}$ . Later in this section, I will explain how to deal with other types of measurement scale.



Suppose we have the data in Figure 4.3 above. Row 27 is the query instance that we want to predict i.e whether the customer is an owner or non owner

Because we use only quantitative  $x_i$ , we can use Euclidean distance. The general formula for Euclidean distance is:

$$d_E(x, y) = \sum_{i=1}^N \sqrt{x_i^2 - y_i^2}$$

Let's enter the above formula as Excel function. Select cell H2. You will see the formula  $=((E2-E\$27)^2+(F2-F\$27)^2)^{0.5}$  entered. This is how the above formula looks like as Excel function. Fill down the formula until cell H25. Thus, we have calculated the Euclidean distance. (see Figure 4.4 below)

E	F	G	H	I
inc_std	lot_std		Distance	
-0.373	-0.720		1.1951	
0.880	-0.853		1.8671	
-0.156	1.048		0.6384	1
-0.383	0.915		0.4450	1
0.982	1.358		1.6688	
2.083	0.562		2.5189	
1.987	-0.543		2.6261	
0.729	1.136		1.3384	
-0.004	0.694		0.4832	1
1.219	0.915		1.7111	
-0.914	1.490		1.1245	
0.653	0.252		1.1091	
0.299	0.164		0.7956	
-0.788	0.694		0.4172	0
-0.176	-0.632		1.1349	
-1.268	0.915		0.9440	
0.789	-0.411		1.5094	
-0.949	-0.632		1.2195	
-0.434	-1.206		1.6799	
-0.115	-0.190		0.7356	
-1.040	-1.206		1.7860	
-1.824	0.031		1.4587	
-0.899	-2.046		2.5624	
-0.297	-1.737		2.2146	
-0.43388	0.473388		# of Owners (1)	3
			# of Non Owners(0)	1
		K=	4	
		Predicted	1	

Figure 4.4

iii. *Determine nearest neighbors based on the K-th minimum distance*

Now that we have established a measure in which to determine the distance between two scenarios, we can simply pass through the data set, one scenario at a time, and compare it to the query scenario. That is to find the K-nearest neighbors. We include a training sample as nearest neighbors if the distance of this training sample to the query instance is less than or equal to the K-th smallest distance. In other words, we rank the distance of all training samples to the query instance and determine the K-th minimum distance.

If the distance of the training sample is below the K-th minimum, then we gather the category **y** of this nearest neighbors' training samples. In MS excel, we can use MS Excel function =SMALL(array, K) to determine the K-th minimum value among the array.

Let me explain the SMALL(array,k) function in details first before we move on.

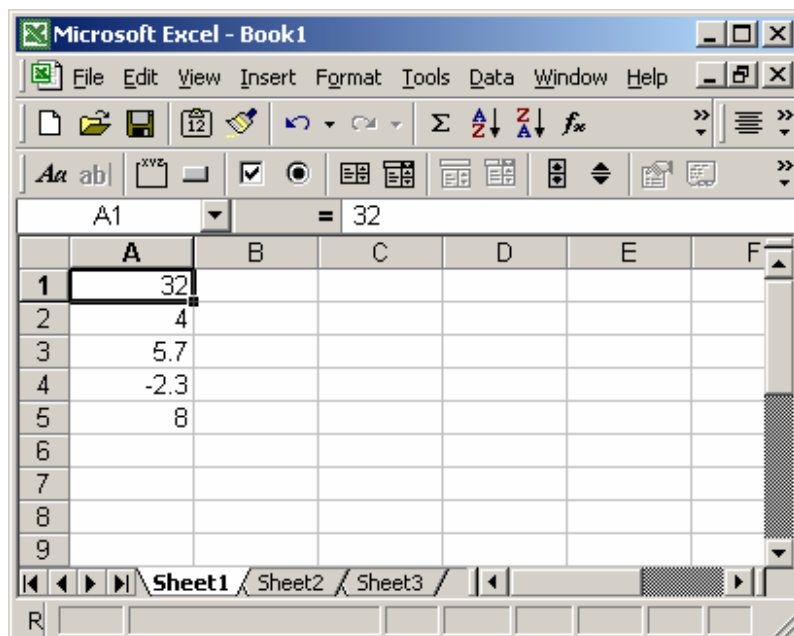
The syntax for the **Small** function is:

Small( array, nth\_position )

*array* is a range or array from which you want to return the nth smallest value.

*nth\_position* is the position from the smallest to return.

Let's take a look at an example:



The screenshot shows a Microsoft Excel window titled "Microsoft Excel - Book1". The spreadsheet has columns A through F and rows 1 through 9. The formula bar at the top shows "A1 = 32". The data in column A is as follows:

	A	B	C	D	E	F
1	32					
2	4					
3	5.7					
4	-2.3					
5	8					
6						
7						
8						
9						

Based on the Excel spreadsheet above:

X

X

X

X

X

X

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

v. *Use simple majority of the category of nearest neighbors as the prediction value of the query instance*

The KNN prediction of the query instance is based on simple majority of the category of nearest neighbors. In our example, the category is only binary, thus the majority can be taken as simple as counting the number of '1' and '0' signs. If the number of 1 is greater than 0, we predict the query instance as '1' and vice versa. If the number of 1 is equal to 0, we can choose arbitrary or determine as one of the 1 or 0.

To calculate the '1' and '0' in our example, we enter the formula =COUNTIF(I2:I25,1) for '1' and =COUNTIF(I2:I25,0) for '0' in cell I27 and I28 respectively. Here we have more '1' than '0'. To determine this we enter the formula =IF(I27>=I28,1,0) in cell H30. This simply say to return '1' if value in cell I27 is more than or equal to I28 else return '0'.

The predicted result is '1' i.e. the customer own a home if the income is \$60000 per annum and live in 20000 square feet lot.

If your training samples contain **y** as categorical data, take simple majority among this data like the example above. If the **y** is quantitative, take average or any central tendency or mean value such as median or geometric mean. We will build another example where **y** is quantitative below.

## **b) Using KNN for time series prediction (Example 2)**

*Using nearest neighbor for stock market data*

As with almost all prediction algorithms, nearest neighbor can be used in a variety of places. Its successful use is mostly dependent on the pre-formatting of the data so that nearness can be calculated and where individual records can be defined. In the text retrieval example this was not too difficult - the objects were documents. This is not always as easy as it is for text retrieval. Consider what it might be like in a time series problem - say for predicting the stock market. In this case the input data is just a long series of stock prices over time without any particular record that could be considered to be an object. The value to be predicted is just the next value of the stock price.

The way that this problem is solved for both nearest neighbor techniques and for some other types of prediction algorithms is to create training records by taking, for instance, 10 consecutive stock prices and using the first 9 as predictor values and the 10th as the prediction value. Doing things this way, if you had 100 data points in your time series you could create 10 different training records.

You could create even more training records than 10 by creating a new record starting at every data point. For instance, you could take the first 10 data points and create a record. Then you could take the 10 consecutive data points starting at the second data point, then

the 10 consecutive data point starting at the third data point. Even though some of the data points would overlap from one record to the next the prediction value would always be different. In our example, I'll only show 5 initial data points as predictor values and the 6<sup>th</sup> as the prediction value.

Let's look at the example now. Goto *Worksheet (Stock Price)*. Using the same principle discussed above, we can extend the K-Nearest Neighbor (KNN) algorithm for prediction (extrapolation) of quantitative data (e.g. time series). In classification, the dependent variable **y** is categorical data. In this example, the dependent variable has quantitative values.

Here is the step by step again on how to compute K-nearest neighbors KNN algorithm for quantitative data:

X

X

X

X

X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X

X

X

X

X

X

X

X

X

	A	B	C	D	E	F
1		<b>K-Nearest Neighbor for Time Series</b>				
2						
3		K	2			
4						
5						Nearest Neighbor
6	Date:	X	Y		distance	Value
7	39861	1	1.2		=+ABS(B7-B\$12)	=IF(E7<=SMALL(E\$7:E\$11,\$C\$3),C7,"")
8	39862	1.2	1.1		=+ABS(B8-B\$12)	=IF(E8<=SMALL(E\$7:E\$11,\$C\$3),C8,"")
9	39863	1.1	1.5		=+ABS(B9-B\$12)	=IF(E9<=SMALL(E\$7:E\$11,\$C\$3),C9,"")
10	39864	1.5	1.8		=+ABS(B10-B\$12)	=IF(E10<=SMALL(E\$7:E\$11,\$C\$3),C10,"")
11	39865	1.8	1.5		=+ABS(B11-B\$12)	=IF(E11<=SMALL(E\$7:E\$11,\$C\$3),C11,"")
12	39866	1.5	?			
13						
14						
15						
16					<b>result</b>	
17					KNN prediction	=+AVERAGE(F7:F11)

Figure 4.6

*1. Determine parameter K = number of nearest neighbors*

Suppose we use K = 2 in cell C3.

*2. Calculate the distance between the query-instance and all the training samples*

Coordinate of query instance is 1.5. As we are dealing with one-dimensional distance, we simply take absolute value from the query instance to value of x.

For instance for x =1, the distance is  $|1 - 1.5| = 0.5$ , for x = 1.2 the distance is  $|1.2 - 1.5| = 0.3$  and so on. (see Figure 4.5 above)

*3. Rank the distance and determine nearest neighbors based on the K-th minimum distance*

Since we have use K = 2, the 2<sup>nd</sup> rank value is 0.3 in this example. The nearest neighbors are 1.1, 1.8 and 1.5 (see Figure 4.5 above)

*4. Gather the values of y of the nearest neighbors*

The values are 1.1, 1.8 and 1.5

*5. Use average of nearest neighbors as the prediction value of the query instance*

X

X

X

gives the lowest MSE from the test set is the optimal k. Let's look at an example on how to implement the cross validation.

### c) Cross Validation Method Using MSE (Example 3)

Open *Worksheet Stock Price (2)*. The time series in B7:B16 is the daily stock price of a shipping company.

I've build 4 test set for this example. Test Set 1 use the prices from 2/17/2009 to 2/21/2009 to predict the price at 2/22/2009, Test Set 2 use the prices from 2/18/2009 to 2/22/2009 to predict the price at 2/23/2009 and so on. The same steps and the formulas in Example 2 are use to build these test set. (see Figure 4.7 below and select respective cell to view the formula entered)


	H	I	J	K	L	M	N	O	P	Q	R	S
1		<b>K-Nearest Neighbor for Cross Validation</b>										
2												
3												
4		Test Set 1						Test Set 2				
5												
6		x	y		Distance	NN Value		x	y		Distance	NN Value
7		1	1.2		0.8			1.2	1.1		0.3	1.1
8		1.2	1.1		0.6	1.1		1.1	1.5		0.4	
9		1.1	1.5		0.7	1.5		1.5	1.8		0	1.8
10		1.5	1.8		0.3	1.8		1.8	1.5		0.3	1.5
11		1.8	?					1.5	?			
12						1.46667						1.46667
13												
14												
15												
16												
17												
18												
19		Test Set 3						Test Set 4				
20												
21		x	y		Distance	NN Value		x	y		Distance	NN Value
22		1.1	1.5		0.35	1.5		1.5	1.8		0.1	1.8
23		1.5	1.8		0.05	1.8		1.8	1.5		0.2	
24		1.8	1.5		0.35	1.5		1.5	1.45		0.1	1.45
25		1.5	1.45		0.05	1.45		1.45	1.6		0.15	1.6
26		1.45	?					1.6	?			
27						1.5625						1.61667
28												
29												
30	Error for Test Set 1	0.001111										
31	Error for Test Set 2	0.000278										
32	Error for Test Set 3	0.001406										
33	Error for Test Set 4	0.001111										
34	MSE	0.000977										

Figure 4.7

The errors for each test set are entered in range I30:I33. For example the formula  $= (M12 - B12)^2$  is enter in cell I30. This means that the result of prediction in test set 1 in cell M12 minus the actual result in cell B12. And then we squared the value. We follow the same reasoning for test set 2, 3 and 4.

After that, we obtain the mean squared error in cell I34 by entering the formula =AVERAGE(I30:I33). This is the MSE we use for cross validation when we use different k in cell C3.

For k = 1 in cell C3 we have MSE = 0.053438

	H	I
29		
30	Error for Test Set 1	0.09
31	Error for Test Set 2	0.1225
32	Error for Test Set 3	0.000625
33	Error for Test Set 4	0.000625
34	MSE	0.053438

For k = 2 in cell C3 we have MSE = 0.001007

	H	I
29		
30	Error for Test Set 1	0.0025
31	Error for Test Set 2	0.000278
32	Error for Test Set 3	0.000625
33	Error for Test Set 4	0.000625
34	MSE	0.001007

For k = 3 in cell C3 we have MSE = 0.000977

	H	I
29		
30	Error for Test Set 1	0.001111
31	Error for Test Set 2	0.000278
32	Error for Test Set 3	0.001406
33	Error for Test Set 4	0.001111
34	MSE	0.000977

For k = 4 in cell C3 we have MSE = 0.003984

	H	I
29		
30	Error for Test Set 1	0.01
31	Error for Test Set 2	0.000625
32	Error for Test Set 3	0.001406
33	Error for Test Set 4	0.003906
34	MSE	0.003984

As you can see from the result above, we obtain the lowest MSE at 0.000977 when k = 3. This is how you do the cross validation.

Thus, for our prediction, we enter k = 3 in cell C3, and the result is 1.616666667 as you can see in cell F17. (see Figure 4.8 below)



	A	B	C	D	E	F
1	<b>K-Nearest Neighbor for Cross Validation</b>					
2						
3		K=	3			
4						
5						
6	Date:	x	y	Distance	Nearest Neighbors Value	
7	2/17/2009	1	1.2	0.5		
8	2/18/2009	1.2	1.1	0.3		
9	2/19/2009	1.1	1.5	0.4		
10	2/20/2009	1.5	1.8	0		1.8
11	2/21/2009	1.8	1.5	0.3		
12	2/22/2009	1.5	1.5	0		1.45
13	2/23/2009	1.45	1.6	0.05		1.6
14	2/24/2009	1.6	1.7	0.1		
15	2/25/2009	1.65	1.5	0.15		
16	2/26/2009	1.5	?			
17			Knn prediction =		1.616666667	

Figure 4.8

Generally, this is how the cross validation method is implemented. You can use more test set and different partition of the stock prices if you want.

## Conclusion

**KNN** is a very robust and simple method for data classification and prediction. It is very effective if the training data is large. However, as we can see from the examples above, it is quite difficult to determine K beforehand. The computation cost is also quite high because we need to compute distance of each query instance to all training samples. Nevertheless, KNN is widely deployed in the area of data mining and can perform well in many situations.

## Chapter 5:

### Building Neural Network Model With MS Excel

#### Introduction to Neural Network

Everyone try to forecast the future. Bankers need to predict credit worthiness of customers. Marketing analyst want to predict future sales. Economists want to predict economic cycles. And everybody wants to know whether the stock market will be up or down tomorrow. Over the years, many software have been developed for this purpose and one such software is the neural network based forecasting application. No, neural network is NOT a medical term. It is actually a branch of artificial intelligence which gains much prominence since the start of the millennium.

NN or neural network is a computer software (and possibly hardware) that simulates a simple model of neural cells in humans. The purpose of this simulation is to acquire the *intelligent* features of these cells. In this book, when terms like *neuron*, *neural network*, *learning*, or *experience* are mentioned, it should be understood that we are using them only in the context of a NN as computer system. NN have the ability to learn by example, e.g. a NN can be trained to recognize the image of car by showing it many examples of a car or to predict future stock prices by feeding it historical stock prices.

We can teach a neural network to perform these particular tasks by using the following procedure:

- I. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
- II. We determine how closely the actual output of the network matches the desired output.
- III. We change the weight of each connection so that the network produces a better approximation of the desired output.

I will show you later, on how to integrate the three steps described above with 5 MS Excel spreadsheet models. With these examples, you can easily understand NN as a non-linear forecasting tool. NO MORE complex C++ programming and complicated mathematic formula(s). I have spent much time and effort to simplify how to use NN as a forecasting tool for you. You only need to know how to use MS Excel, in modeling NN as a powerful forecasting method. THAT'S IT!.

#### Technical Stuff of neural network that you don't really have to know.

Neural networks are very effective when lots of examples must be analyzed, or when a structure in these data must be analyzed but a single algorithmic solution is impossible to formulate. When these conditions are present, neural networks are use as computational tools for examining data and developing models that help to identify interesting patterns

or structures in the data. The data used to develop these models is known as training data. Once a neural network has been trained, and has learned the patterns that exist in that data, it can be applied to new data thereby achieving a variety of outcomes. Neural networks can be used to

- learn to **predict** future events based on the patterns that have been observed in the historical training data;
- learn to **classify** unseen data into pre-defined groups based on characteristics observed in the training data;
- learn to **cluster** the training data into natural groups based on the similarity of characteristics in the training data.

We have seen many different neural network models that have been developed over the last fifty years or so to achieve these tasks of prediction, classification, and clustering. In this book we will be developing a neural network model that has successfully found application across a broad range of business areas. We call this model a **multilayered feedforward neural network (MFNN)** and is an example of a neural network trained with supervised learning.

We feed the neural network with the training data that contains complete information about the characteristics of the data and the observable outcomes in a supervised learning method. Models can be developed that learn the relationship between these characteristics (inputs) and outcomes (outputs). For example, we can develop a MFNN to model the relationship between money spent during last week's advertising campaign and this week's sales figures is a prediction application. Another example of using a MFNN is to model and classify the relationship between a customer's demographic characteristics and their status as a high-value or low-value customer. For both of these example applications, the training data must contain numeric information on both the inputs and the outputs in order for the MFNN to generate a model. The MFNN is then repeatedly trained with this data until it learns to represent these relationships correctly.

For a given input pattern or data, the network produces an output (or set of outputs), and this response is compared to the known desired response of each neuron. For classification problems, the desired response of each neuron will be either zero or one, while for prediction problems it tends to be continuous valued. Correction and changes are made to the weights of the network to reduce the errors before the next pattern is presented. The weights are continually updated in this manner until the total error across all training patterns is reduced below some pre-defined tolerance level. We call this learning algorithm as the backpropagation.

### ***Process of a backpropagation***

- I. Forward pass, where the outputs are calculated and the error at the output units calculated.
- II. Backward pass, the output unit error is used to alter weights on the output units.

Then the error at the hidden nodes is calculated (by back-propagating the error at the output units through the weights), and the weights on the hidden nodes altered using these values.

*The main steps of the back propagation learning algorithm are summarized below:*

Step 1: Input training data.

Step 2: Hidden nodes calculate their outputs.

Step 3: Output nodes calculate their outputs on the basis of Step 2.

Step 4: Calculate the differences between the results of Step 3 and targets.

Step 5: Apply the first part of the training rule using the results of Step 4.

Step 6: For each hidden node,  $n$ , calculate  $d(n)$ . (derivative)

Step 7: Apply the second part of the training rule using the results of Step 6.

Steps 1 through 3 are often called the *forward pass*, and steps 4 through 7 are often called the *backward pass*. Hence, the name: back-propagation. For each data pair to be learned a forward pass and backwards pass is performed. This is repeated over and over again until the error is at a low enough level (or we give up).

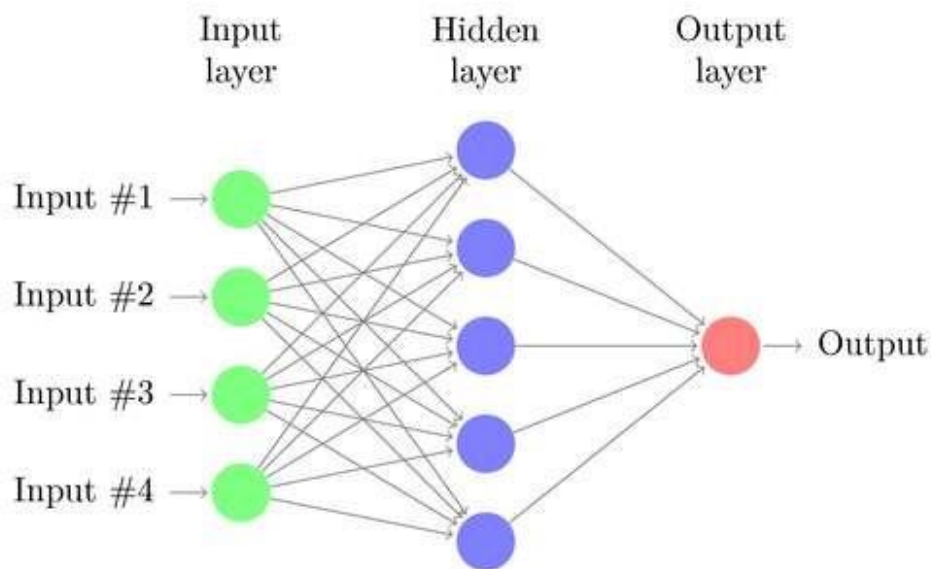


Figure 5.1

### ***Calculations and Transfer Function***

The behaviour of a NN (Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

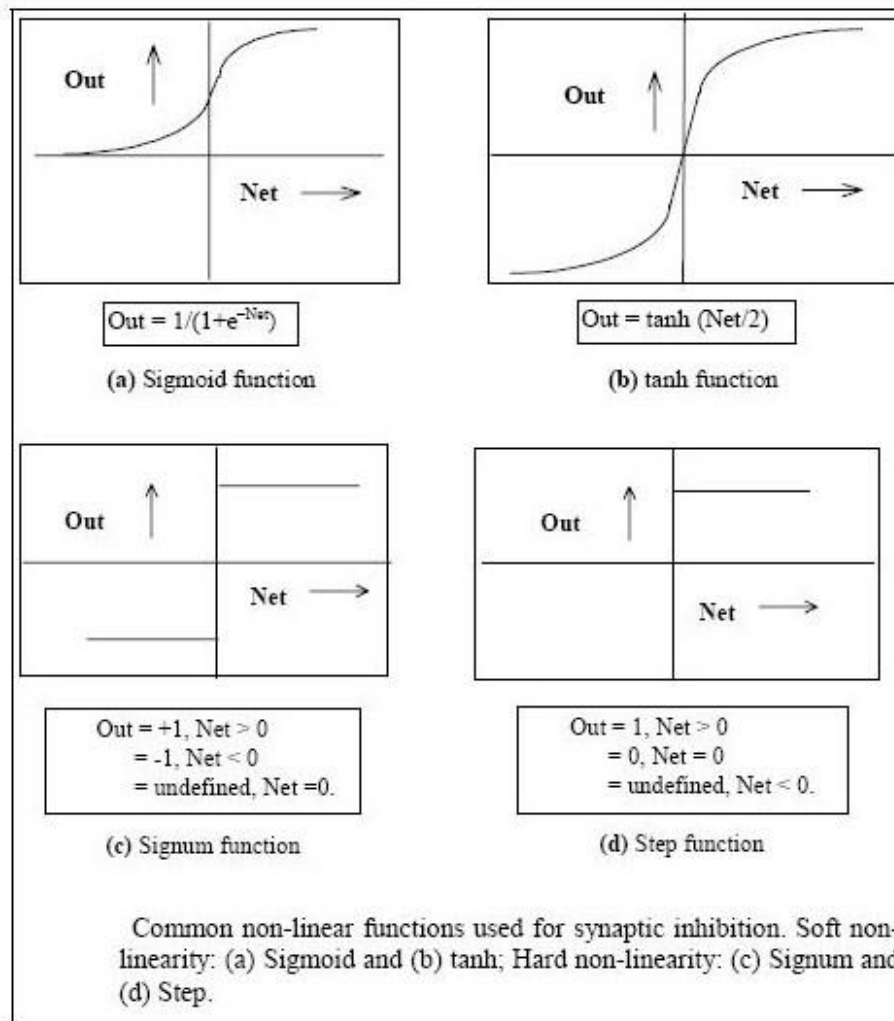
- ☐ linear
- ☐ threshold
- ☐ sigmoid

For linear units, the output activity is proportional to the total weighted output.

For threshold units, the output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

It should be noted that the sigmoid curve is widely used as a transfer function because it has the effect of "squashing" the inputs into the range  $[0,1]$ . Other functions with similar features can be used, most commonly tanh which has an output range of  $[-1,1]$ . The sigmoid function has the additional benefit of having an extremely simple derivative function for backpropagating errors through a feed-forward neural network. This is how the transfer functions look like:



To make a neural network performs some specific task, we must choose how the units are connected to one another (see Figure 5.1), and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

Typically the weights in a neural network are initially set to small random values; this represents the network knowing nothing. As the training process proceeds, these weights will converge to values allowing them to perform a useful computation. Thus it can be said that the neural network commences knowing nothing and moves on to gain some real knowledge.

To summarize, we can teach a three-layer network to perform a particular task by using the following procedure:

- I. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
- II. We determine how closely the actual output of the network matches the desired output.
- III. We change the weight of each connection so that the network produces a better approximation of the desired output.

***The advantages of using Artificial Neural Networks software are:***

- I. They are extremely powerful computational devices
- II. Massive parallelism makes them very efficient.
- III. They can learn and generalize from training data – so there is no need for enormous feats of programming.
- IV. They are particularly fault tolerant – this is equivalent to the “graceful degradation” found in biological systems.
- V. They are very noise tolerant – so they can cope with situations where normal symbolic systems would have difficulty.
- VI. In principle, they can do anything a symbolic/logic system can do, and more.

**Real life applications**

The applications of artificial neural networks are found to fall within the following broad categories:

*Manufacturing and industry:*

- ☐ Beer flavor prediction
- ☐ Wine grading prediction
- ☐ For highway maintenance programs

*Government:*

- ☐ Missile targeting
- ☐ Criminal behavior prediction

*Banking and finance:*

- ☐ Loan underwriting
- ☐ Credit scoring
- ☐ Stock market prediction
- ☐ Credit card fraud detection
- ☐ Real-estate appraisal

*Science and medicine:*

- ☐ Protein sequencing
- ☐ Tumor and tissue diagnosis
- ☐ Heart attack diagnosis
- ☐ New drug effectiveness
- ☐ Prediction of air and sea currents

In this book we will examine some detailed case studies with Excel spreadsheets demonstrating how the MFNN has been successfully applied to problems as diverse as

- ☐ Credit Approval,
- ☐ Sales Forecasting,
- ☐ Predicting DJIA weekly prices,
- ☐ Predicting Real Estate value
- ☐ Classify Type of Flowers

This book contains 5 neural network models develop using Excel worksheets described above. Instructions on how to build neural network model with Excel will be explained step by step by looking at the 5 main sections shown below...

- a) Selecting and transforming data
- b) the neural network architecture,
- c) simple mathematic operations inside the neural network model
- d) training the model and
- e) using the trained model for forecasting

Let's start building:

## 1) The Credit Approval Model

Credit scoring is a technique to predict the creditworthiness of a candidate applying for a loan, credit card, or mortgage. The ability to accurately predict the creditworthiness of an applicant is a significant determinant of success in the financial lending industry. Refusing credit to creditworthy applicants results in lost opportunity, while heavy financial losses occur if credit is given indiscriminately to applicants who later default on their obligations.

In this example, we will use neural network to forecast the risk level of granting a loan to the applicant. It can be used to guide decisions for granting or denying new loan applications.

### a) Selecting and transforming data

Open the *Workbook(Credit\_Approval)* in folder Chapter 5 and bring up *worksheet (Raw*



*Data*). Here we have 400 inputs patterns and desire outputs. There are 10 input factors and 1 desire output (end result). We can see that, the data are still in alphabet form. Neural network (NN) can only be fed with numeric data for training. So we need to transform these raw data into numeric form.

This worksheet is self explanatory. For example, in column B (Input 2), we have the marital status. NN cannot take or understand “married or single”. So we transform them to 1 for “married” and 0 for “single”. We have to do this one by one manually. If you select the *worksheet(Transform Data)*, it contains exactly what has been transform from *worksheet(Raw Data)*.

	A	B	C	D	E	F
1	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6
2	Age	Marital Status	Occupation	Sex	Address Time	Job Time
3		Married = 1	Unemployed = 0	Male = 1		
4		Single = 0	Semi-professional = 0.15	Female = 0		
5			Professional = 0.3			
6			Blue Collar = 0.45			
7			Manager = 0.6			
8			Office = 0.75			
9			Principal = 0.9			
10			Retired = 1			
11						
12	18.5	1	0.45	1	0	1.25
13	59.5	1	0.3	0	4.46	3.04
14	24.5	1	0.3	0	0.5	1.5
15	28.5	1	0.15	0	1.25	0
16	25.92	1	0.45	1	0.875	0.375
17	23.08	1	0.45	1	0	1
18	39.85	1	0.15	1	5	0

Figure 5.2

Now, we can see that Column A to column L in the *worksheet (Transform Data)* are all numerical. (see Figure 5.2 above) Apart from this transformation, we also need to “massage” the numeric data a little bit. This is because NN will learn better if there is uniformity in the data.

We need to transform all the 400 rows of data into range between the values 0 to 1. The first 398 rows will be used as training data. The last 2 rows will be used for testing our prediction later. Thus we need to scale all the data into the value between 0 to 1. How do we do that?

- 1- Copy all data from Column A to L to Column N To Column Y
- 2- Then, select *Scale Data* on the **nn\_Solve** menu (see Figure 5.3) (see Appendix A on how to load **nn\_Solve**).

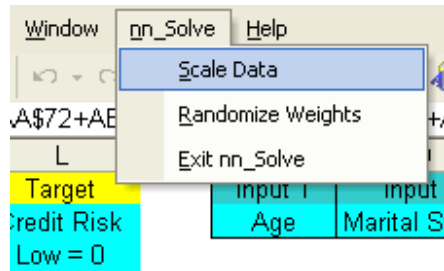


Figure 5.3

Enter the reference that you want to scale in the **Data Range**. We scale Input 1 (Age) first. Enter N12:N411 in the **Data Range**. Press the Tab key on your keyboard to exit. When you press Tab, **nn\_Solve** will automatically load the maximum (70) and the minimum (15.83) in the *Raw Data* frame *Min* and *Max* textbox. (see Figure 5.4 below)

3- Then specify the maximum (1) and minimum (0) scale range. Click on the **Scale Now** button. The raw data will be scaled.

**nn\_Solve** will also automatically store the minimum (in cell N414) and the maximum (cell N413) value of the raw data in the last row and first column of the raw data. (see Figure 5.5 below)

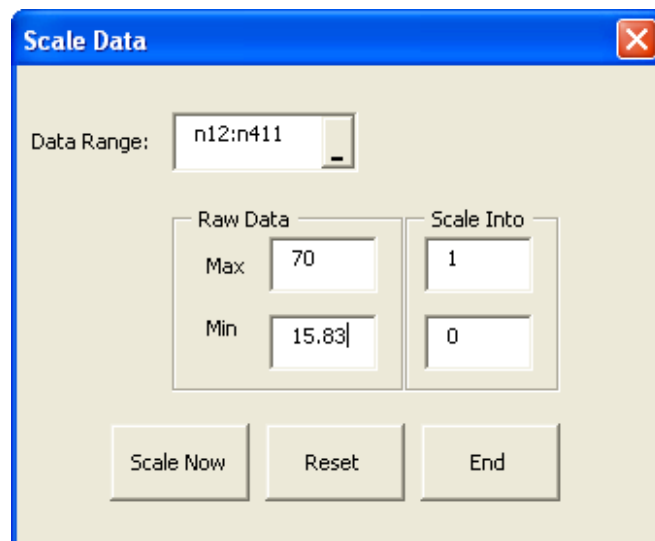


Figure 5.4

	M	N	O	P
1		Input 1	Input 2	Input 3
2		Age	Marital Status	Occupation
3				
4				
5				
6				
7				
8				
9				
10				
408		0.343179	1	0.6
409		0.353886	1	0.6
410		0.504707	1	0.6
411		0.126269	1	0.45
412				
413		70		
414		15.83		
415				

Figure 5.5

The raw input data that need to be scale are Input 5 (Address Time), Input 6 (Job Time) and Input 9 (Payment History). We do not need to scale Input 2,3,4,7,6,10 as these values are within 0 to 1. We do not need to scale the desire output (Credit Risk) as the value is already within 0 to 1.

	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Input 8	Input 9	Input 10		Desire
2	Age	Marital Status	Occupation	Sex	Address Time	Job Time	Checking	Savings	Payment History	Home Ownership		Credit Risk
3		Married = 1	Unemployed = 0	Male = 1			Yes = 1			Own = 1		Low = 0
4		Single = 0	Professional = 0	Female = 0			No = 0			Rent = 0		High = 1
5			Professional = 0.3									
6			Blue Collar = 0.45									
7			Manager = 0.6									
8			Office = 0.75									
9			Principal = 0.9									
10			Retired = 1									
405	0.0923	0	0.75	0	0.017857143	0.03509	0	0	0	0		1
406	0.23703	1	0.45	1	0.517857143	0.00439	0	0	0	0		1
407	0.08935	0	0.45	1	0.029821429	0.07018	0	0	0	1		1
408	0.34318	1	0.6	1	0.047678571	0.00439	0	0	0	1		1
409	0.35389	1	0.6	1	0.008928571	0.14035	0	0	0	1		1
410	0.50471	1	0.6	1	0.178571429	0.07895	0	0	0	1		1
411	0.12627	1	0.45	0	0.25	0.00579	0	0	0	0		1
412												
413	70				28	28.5			67			
414	15.83				0	0			0			
415												

Figure 5.6

Figure 5.6 above show the data after they have been scaled. We need the raw minimum and maximum values later when we reverse the scale values back to raw value.

### b) the neural network architecture

A neural network is a group of neurons connected together. Connecting neurons to form a NN can be done in various ways. This worksheet; column N to column AK actually contain the NN architecture shown below:

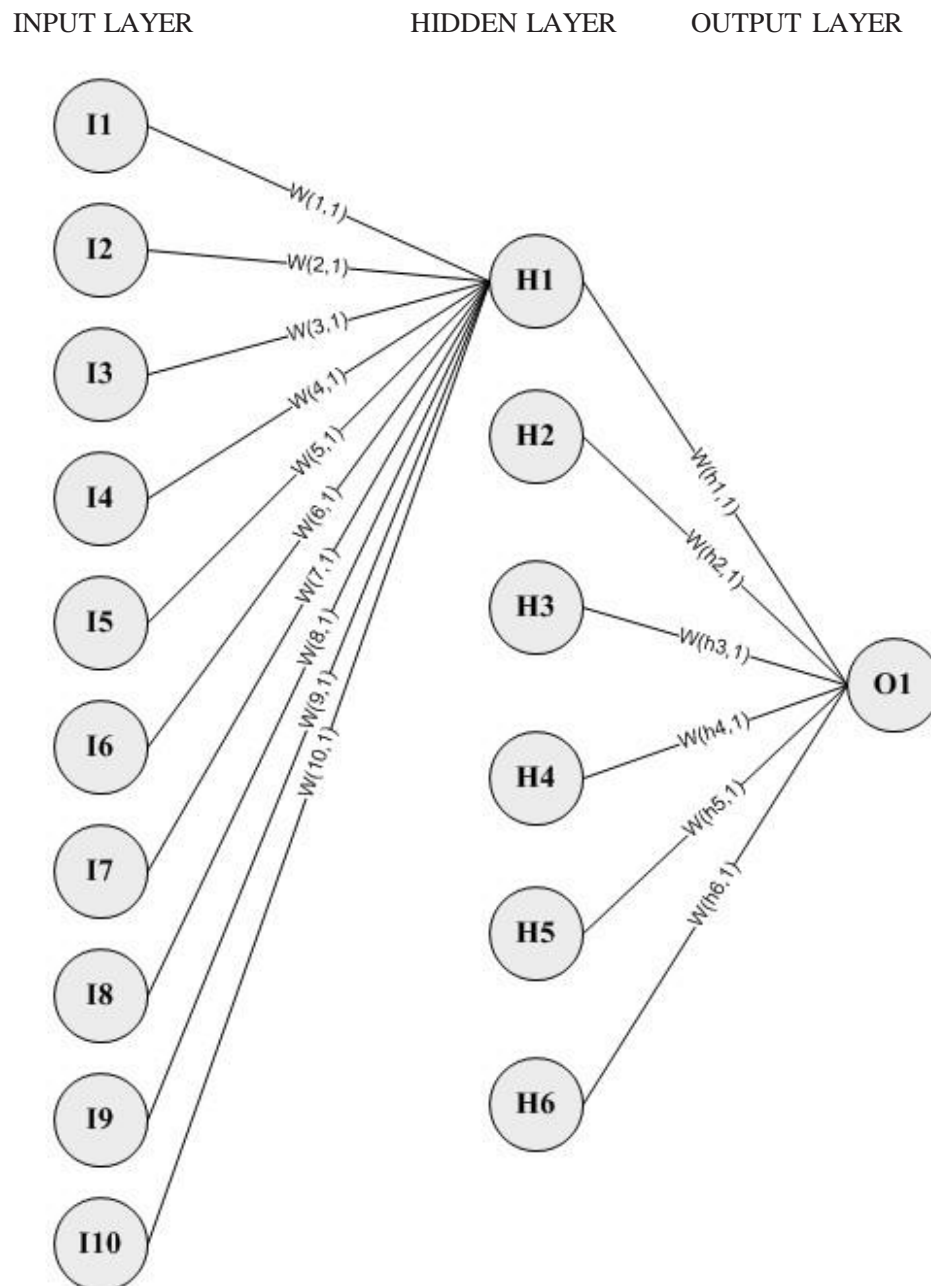


Figure 5.7

There are 10 nodes or neuron on the input layer. 6 neurons on the hidden layer and 1 neuron on the output layer.

Those lines that connect the nodes are call weights. I only connect part of them. In reality all the weights are connected layer by layer, like Figure 5.1 above

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. Here we have 400 input patterns map to 400 desired or target outputs. We reserve 2 input patterns for testing later.

Like what you see from the Figure 5.7 above, this NN model consists of three layers:

1. Input layer with 10 neurons.

Column N = Input 1 (I1); Column O = Input 2 (I2); Column P = Input 3 (I3);  
Column Q = Input 4 (I4) ; Column R = Input 5 (I5) ; Column S = Input 6 (I6);  
Column T = Input 7 (I7); Column U = Input 8 (I8); Column V = Input 9 (I9);  
Column W = Input 10 (I10)

2. Hidden layer with 6 neurons.

Column AD = Hidden Node 1 (H1); Column AE = Hidden Node 2 (H2);  
Column AF = Hidden Node 3 (H3); Column AG = Hidden Node 4 (H4);  
Column AH = Hidden Node 5 (H5); Column AI = Hidden Node 6 (H6)

3. Output layer with 1 neurons.

Column AK = Output Node 1

Now let's talk about the weights that connection all the neurons together

Note that:

- ✓ The output of a neuron in a layer goes to all neurons in the following layer.  
(in Fig 5.7, I only connect the weights between all the Input nodes to Hidden Node 1)
- ✓ We have 10 inputs node and 6 hidden nodes and 1 output nodes. Here the number of weights are  $(10 \times 6) + (6 \times 1) = 66$
- ✓ Each neuron has its own input weights.
- ✓ The output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input.

I have put the weights vector in one column AA. So the weights are contain in cells:

From Input Layer to Hidden Layer

$w(1,1) = \$AA\$12$  -> connecting I1 to H1

$w(2,1) = \$AA\$13$  -> connecting I2 to H1

$w(3,1) = \$AA\$14$  -> connecting I3 to H1

$w(4,1) = \$AA\$15$  -> connecting I4 to H1

$w(5,1) = \$AA\$16$  -> connecting I5 to H1

$w(6,1) = \$AA\$17$  -> connecting I6 to H1

$w(7,1) = \$AA\$18$  -> connecting I7 to H1  
 $w(8,1) = \$AA\$19$  -> connecting I8 to H1  
 $w(9,1) = \$AA\$20$  -> connecting I9 to H1  
 $w(10,1) = \$AA\$21$  -> connecting I10 to H1

$w(1,2) = \$AA\$22$  -> connecting I1 to H2  
 $w(2,2) = \$AA\$23$  -> connecting I2 to H2  
 $w(3,2) = \$AA\$24$  -> connecting I3 to H2  
 $w(4,2) = \$AA\$25$  -> connecting I4 to H2  
 $w(5,2) = \$AA\$26$  -> connecting I5 to H2  
 $w(6,2) = \$AA\$27$  -> connecting I6 to H2  
 $w(7,2) = \$AA\$28$  -> connecting I7 to H2  
 $w(8,2) = \$AA\$29$  -> connecting I8 to H2  
 $w(9,2) = \$AA\$30$  -> connecting I9 to H2  
 $w(10,2) = \$AA\$31$  -> connecting I10 to H2

“ and

“ so

“ on

$w(1,6) = \$AA\$62$  -> connecting I1 to H6  
 $w(2, 6) = \$AA\$63$  -> connecting I2 to H6  
 $w(3, 6) = \$AA\$64$  -> connecting I3 to H6  
 $w(4, 6) = \$AA\$65$  -> connecting I4 to H6  
 $w(5, 6) = \$AA\$66$  -> connecting I5 to H6  
 $w(6, 6) = \$AA\$67$  -> connecting I6 to H6  
 $w(7, 6) = \$AA\$68$  -> connecting I7 to H6  
 $w(8, 6) = \$AA\$69$  -> connecting I8 to H6  
 $w(9, 6) = \$AA\$70$  -> connecting I9 to H6  
 $w(10, 6) = \$AA\$71$  -> connecting I10 to H6

From Hidden Layer to Output Layer

$w(h1,1) = \$AA\$72$  -> connecting H1 to O1  
 $w(h2, 1) = \$AA\$73$  -> connecting H2 to O1

$w(h3, 1) = \$AA\$74$  -> connecting H3 to O1

$w(h4, 1) = \$AA\$75$  -> connecting H4 to O1

$w(h5, 1) = \$AA\$76$  -> connecting H5 to O1

$w(h6, 1) = \$AA\$77$  -> connecting H6 to O1

After mapping the NN architecture to the worksheet and entering the input and desired output data., it is time to see what is happening inside those nodes.

### c) simple mathematic operations inside the neural network model

The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, and could change per network configuration and type of data. In general the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance. You could start a network configuration using a single hidden layer, and add more hidden layers if you notice that the network is not learning as well as you like.

For this Credit Approval model, 1 hidden layer is sufficient. Select the cell AD12 (H1), you can see Figure 5.8

X

X

X

X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X

X

X

X

X

X

X

X

X

X

X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X



actual output is 0.498473 in cell AK12. Since 0.495473 is not what we want, we minus this with the desire output.

$$\text{Error} = \text{AK12} - \text{AL12}$$

After that we square the error to get a positive value

$$\text{AM12} = \text{AK12} - \text{AL12}^2 ; \text{ we get } 0.245494 \text{ (in AM12)}$$

The closer the actual output of the network matches the desired output, the better. This is only one pattern error. Since we have 398 rows of patterns, we fill down the formula again until row 409 in this spreadsheet. Sum up all the error and take the average.

$$\text{MSE} = \text{SUM}(\text{AM12}:\text{AM409})/398$$

Our objective is to minimize the MSE. We need to change the weight of each connection so that the network produces a better approximation of the desired output.

In NN technical term, we call this step of changing the weights as TRAINING THE NEURAL NETWORK.

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection. Phew, what craps is this back propagation!!!. Fortunately, you don't need to understand this, if you use MS Excel Solver to build and train a neural network model.

#### **d) Training NN as an Optimization Task Using Excel Solver**

Training a neural network is, in most cases, an exercise in numerical optimization of a usually nonlinear function. Methods of nonlinear optimization have been studied for hundreds of years, and there is a huge literature on the subject in fields such as numerical analysis, operations research, and statistical computing, e.g., Bertsekas 1995, Gill,

Murray, and Wright 1981. There is no single best method for nonlinear optimization. You need to choose a method based on the characteristics of the problem to be solved.

MS Excel's Solver is a numerical optimization add-in (an additional file that extends the capabilities of Excel). It can be fast, easy, and accurate.

For a medium size neural network model with moderate number of weights, various quasi-Newton algorithms are efficient. For a large number of weights, various conjugate-gradient algorithms are efficient. These two optimization method are available with Excel Solver

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence. Values between -1 to 1 will be the best starting weights.

Let' fill out the weight vector. The weights are contain in AA12:AA77. From the **nn\_Solve** menu, select *Randomize Weights*

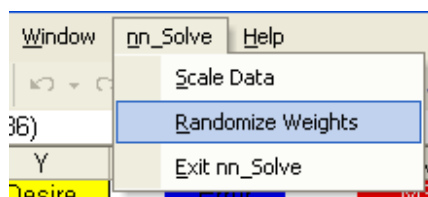


Figure 5.10

Enter AA12:AA77 and click on the **Randomize Weights** button. AA12:AA77 will be filled out with values between -1 to 1. (see Figure 5.11 below)

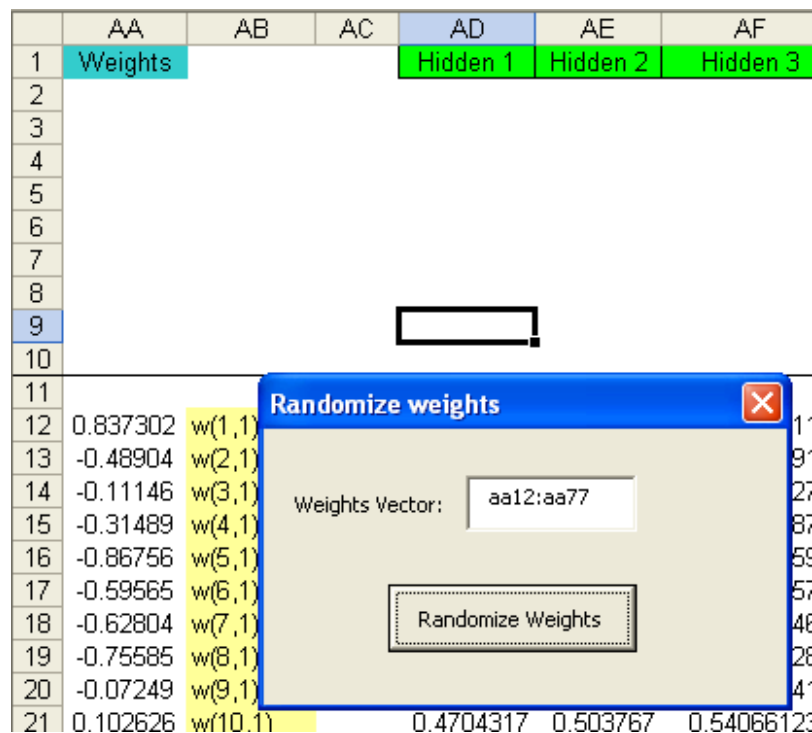


Figure 5.11

The learning algorithm improves the performance of the network by gradually changing each weight in the proper direction. This is called an **iterative** procedure. Each iteration makes the weights slightly more efficient at separating the target from the nontarget examples. The iteration loop is usually carried out until no further improvement is being made. In typical neural networks, this may be anywhere from ten to ten-thousand iterations. Fortunately, we have Excel Solver. This tool has simplified neural network training so much.

### *Accessing Excel's Solver (see Appendix C to access Excel's Solver for Excel 2007)*

To use the Solver, click on the Tools heading on the menu bar and select the Solver . . . item. (see Figure 5.12)

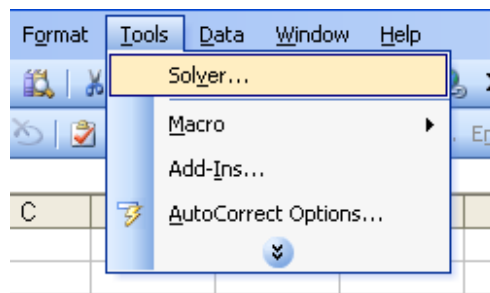


Figure 5.12

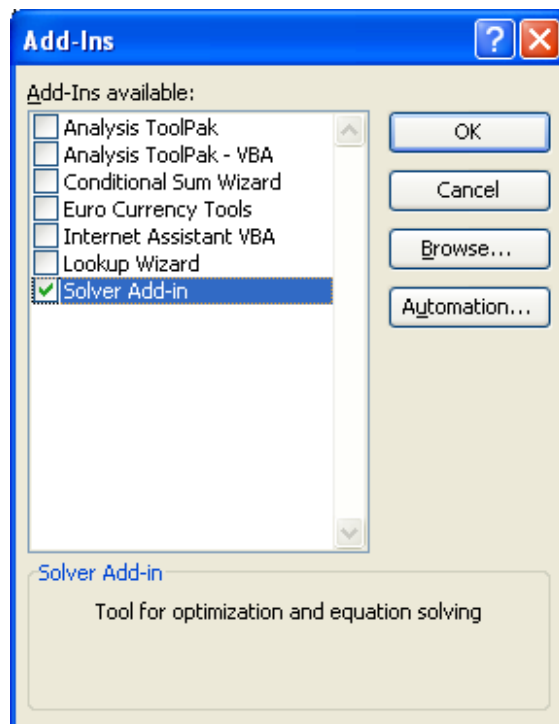


Figure 5.14

If Solver is not listed (see Figure 5.12) then you must manually include it in the algorithms that Excel has available. To do this, select Tools from the menu bar and choose the "Add-Ins . . ." item. In the Add-Ins dialog box, scroll down and click on the Solver Add-In so that the box is checked as shown Figure 5.14 above:

After selecting the Solver Add-In and clicking on the OK button, Excel takes a moment to call in the Solver file and adds it to the Tools menu.

If you cannot find the Solver Add-In, try using the Mac's Find File or Find in Windows to locate the file. Search for "solver." Note the location of the file, return to the Add-Ins dialog box (by executing Tools: Add-Ins...), click on Select or Browse, and open the Solver Add-In file.

What if you still cannot find it? Then it is likely your installation of Excel failed to include the Solver Add-In. Run your Excel or Office Setup again from the original CD-ROM and install the Solver Add-In. You should now be able to use the Solver by clicking on the Tools heading on the menu bar and selecting the Solver item.

Although Solver is proprietary, you can download a trial version from Frontline Systems, the makers of Solver, at [www.frontsys.com](http://www.frontsys.com).

After executing Tools: Solver . . . , you will be presented with the Solver Parameters dialog box below:

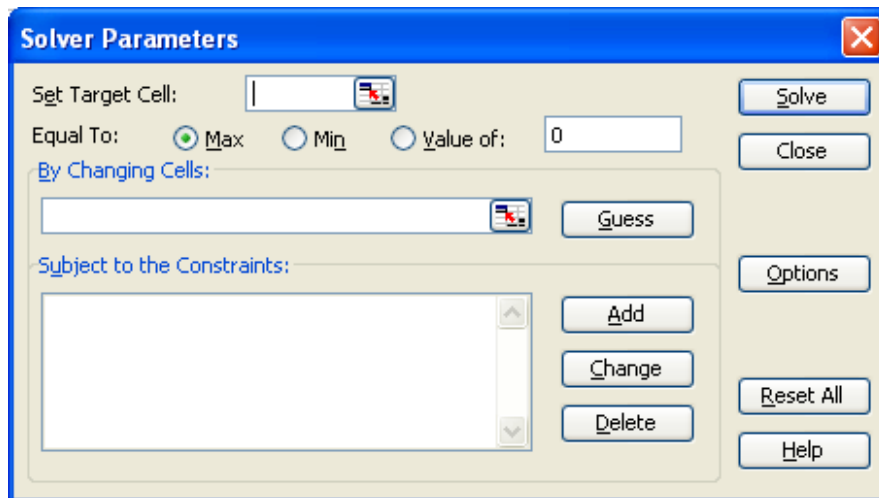


Figure 5.15

Let us review each part of this dialog box, one at a time.

**Set Target Cell** is where you indicate the objective function (or goal) to be optimized. This cell must contain a formula that depends on one or more other cells (including at least one “changing cell”). You can either type in the cell address or click on the desired cell. Here we enter cell AO1.

In our NN model, the objective function is to minimize the Mean Squared Error. See Figure 5.16 below

AL	AM	AN	AO	AP	AQ	AR	
Desire	Error	MSE	0.082670549				

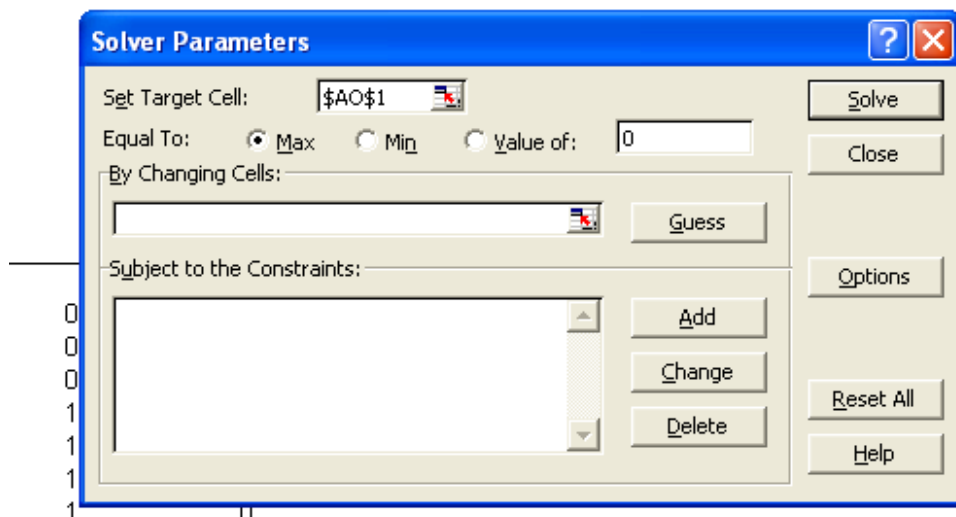


Figure 5.16

**Equal to:** gives you the option of treating the Target Cell in three alternative ways. **Max** (the default) tells Excel to maximize the Target Cell and **Min**, to minimize it, whereas **Value** is used if you want to reach a certain particular value of the Target Cell by choosing a particular value of the endogenous variable.

Here, we select **Min** as we want to minimize the MSE

**By Changing Cells** permits you to indicate which cells are the adjustable cells (i.e., endogenous variables). As in the Set Target Cell box, you may either type in a cell address or click on a cell in the spreadsheet. Excel handles multivariable optimization problems by allowing you to include additional cells in the By Changing Cells box. Each noncontiguous choice variable is separated by a comma. If you use the mouse technique (clicking on the cells), the comma separation is automatic.

Here, the cells that need to be changed is the weights vector. In our model, the weights are contain in Range AA12:AA77. So we enter, AA12:AA77. See Figure 5.17 below

AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ
Weights			Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5	Hidden 6	

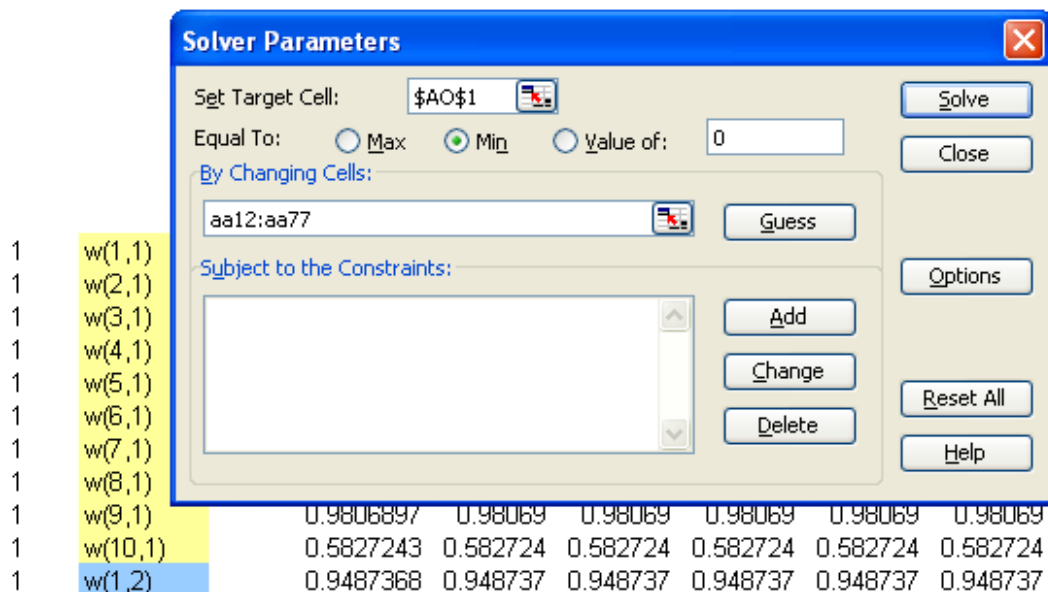


Figure 5.17

**Subject to the Constraints** is used to impose constraints on the endogenous variables. We will rely on this important part of Solver when we do Constrained Optimization problems. Although, training a neural network is not a Constrained Optimization problem, but it will be efficient if we limit the maximum and minimum value of the weights to a range of -100 to 100. Therefore, we put in

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

138

Please visit <http://www.xlptert.com/forecast.htm> for more information on the book

[illegible]



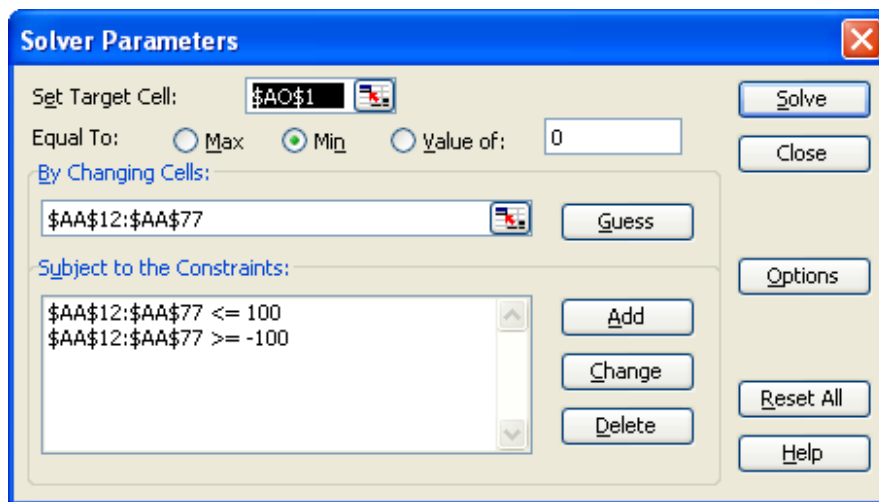


Figure 5.21

**Solve** is obviously the button you click to get Excel's Solver to find a solution. This is the last thing you do in the Solver Parameters dialog box. So, click **Solve** to start training.

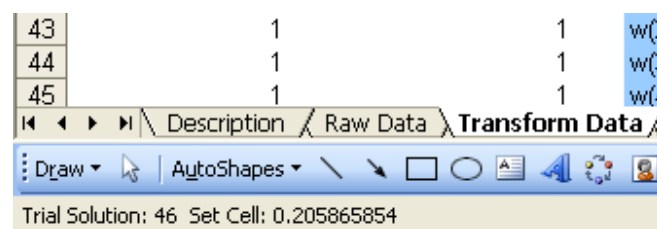


Figure 5.22

When Solver start optimizing, you will see the *Trial Solution* at the bottom left of your spreadsheet. See Figure 5.22 above.

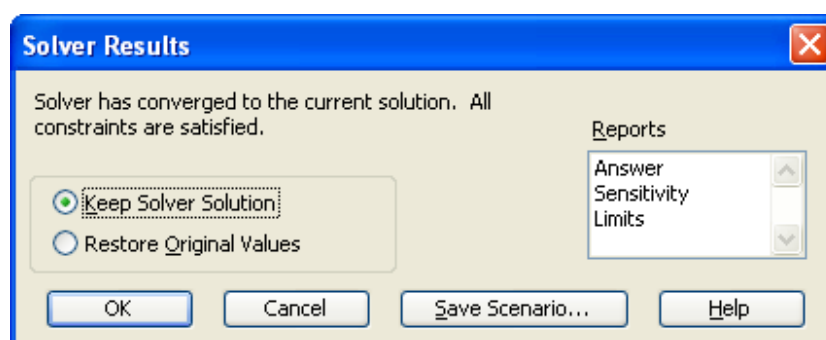


Figure 5.23

A message will appear after Solver has converged (see Figure 5.23)..In this case, Excel reports that “Solver has converged to the current solution. All constraints are satisfied.” This is good news!

Sometime, the Mean Square Error is not satisfactory and Solver unable to find the solution at one go. If this is the case then you, Keep the Solver solution and run Solver again. Follow the step discussed above. From experience, usually you will need to run Solver a few times before Solver arrive at a satisfactory Mean Square Error. (Note: value less than 0.01 will be satisfactory)

Bad news is a message like, “Solver could not find a solution.” If this happens, you must diagnose, debug, and otherwise think about what went wrong and how it could be fixed. The two quickest fixes are to try different initial weights values and to add bigger or smaller constraints to the weights.

Or you may change the network architecture by adding more hidden nodes.

From the Solver Results dialog box, you elect whether to have Excel write the solution it has found into the Changing Cells (i.e., Keep Solver Solution) or whether to leave the spreadsheet alone and NOT write the value of the solution into the Changing Cells (i.e., Restore Original Values). When Excel reports a successful run, you would usually want it to Keep the Solver Solution.

On the right-hand side of the Solver Results dialog box, Excel presents a series of reports. The Answer, Sensitivity, and Limits reports are additional sheets inserted into the current workbook. They contain diagnostic and other information and should be selected if Solver is having trouble finding a solution.

*It is important to understand that a saved Excel workbook will remember the information included in the last Solver run.*

**Save Scenario...** enables the user to save particular solutions for given configurations.

### **e) using the trained model for forecasting**

After all the training and the MSE is below 0.01, its now time for us to predict. Here, I’ve train the model and the MSE is 0.0086.

Goto the row 409 of the *Credit Approval* spreadsheet. Remember, we have save the last 2 rows for testing, row 410 and 411.

After that goto the hidden layer. Select AD409:AK409 (see Figure 5.24 below)

	AD	AE	AF	AG	AH	AI	AJ	AK	AL
1	Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5	Hidden 6		Output	Desire
2									
3									
4									
5									
6									
7									
8									
9									
10									
407	0.9801707	1	2.99E-26	5.96E-05	2.94E-08	1		1	1
408	1.434E-06	0.189955	6.9E-48	3.19E-05	2.71E-12	1		1	1
409	1.659E-05	0.074608	3.75E-45	7.63E-05	7.98E-12	1		1	1
410									
411									
412									
413									

Figure 5.24

Fill down the formula to row 410 to 411. (see Figure 5.25)

	AD	AE	AF	AG	AH	AI	AJ	AK
1	Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5	Hidden 6		Output
2								
3								
4								
5								
6								
7								
8								
9								
10								
407	0.9801707	1	2.99E-26	5.96E-05	2.94E-08	1		1
408	1.434E-06	0.189955	6.9E-48	3.19E-05	2.71E-12	1		1
409	1.659E-05	0.074608	3.75E-45	7.63E-05	7.98E-12	1		1
410	1.524E-05	0.851871	1.88E-50	4.6E-05	3.22E-12	1		1
411	3.396E-15	0.139026	1.5E-21	0.003956	2.79E-07	1		1
412								

Figure 5.25

On the Output cell i.e. AK410, the predicted value is 1 and AK411 is also 1 (see Figure 5.26 below). Thus, both results we got are High Risk. When you compare to the actual result in L410 and L411, we are spot on. That's it. You have successfully use neural network to predict the risk involve when granting a loan to an applicant.

	AH	AI	AJ	AK	AL
1	Hidden 5	Hidden 6		Output	Desire
2					
3					
4					
5					
6					
7					
8					
9					
10					
408	2.71E-12	1		1	1
409	7.98E-12	1		1	1
410	3.22E-12	1		1	
411	2.79E-07	1		1	
412					

Figure 5.26

## 2) The Sales Forecasting Model

Forecasting future retail sales is one of the most important activities that form the basis for all strategic and planning decisions in effective operations of retail businesses as well as retail supply chains.

Accurate forecasts of consumer retail sales can help improve retail supply chain operation, especially for larger retailers who have a significant market share. For profitable retail operations, accurate demand forecasting is crucial in organizing and planning purchasing, production, transportation, and labor force, as well as after-sales services. A poor forecast would result in either too much or too little inventory, directly affecting the profitability of the supply chain and the competitive position of the organization.

In this example, we will use neural network to forecast the weekly and daily sales of a fashion store.

### a) Selecting and transforming data

Open the *Workbook(Sales\_Forecasting)* in folder Chapter 5 and bring up *Worksheet (Raw Data)*. There are 7 input factors and 2 desired output (end result). We also have 104 rows of input patterns in this model.

We need to transform all the 104 rows of data into range between the values 0 to 1. The first 102 rows will be used as training data. The last 2 rows will be used for testing our prediction later. Here we can see that, the data are still in alphabet form. NN can only be fed with numeric data for training. So we need to transform these raw data into numeric form. (see Figure 5a.1) This worksheet is self explanatory. For example, in column B (Input 2), we have the Season Influence. NN cannot take or understand “Low, Medium, High, Very High”. So we transform them to Low = 0.25, Medium = 0.5, High = 0.75, Very High = 0.9. We have to do this one by one manually.

	A	B	C	D
1	Input 1	Input 2	Input 3	Input 4
2	Week No.	Season Influence	Holiday Influence	Competitors Sales
3	1	High	High	3
4	2	Medium	Medium	6
5	3	Low	None	8
6	4	Low	None	1
7	5	Low	None	5
8	6	Low	None	0
9	7	Low	None	7
10	8	Low	Medium	9
11	9	Low	None	3
12	10	Low	None	5
13	11	Low	None	2
14	12	Low	None	1
15	13	Low	None	6
16	14	Low	None	5
17	15	Low	None	8
18	16	Low	None	7
19	17	Low	None	5
20	18	Medium	None	7
21	19	Medium	None	9

Figure 5a.1

If you select the *worksheet(Transform Data)*, it contains exactly what has been transform from *worksheet(Raw Data)*. (see Figure 5a.2)

	A	B	C	D	E	F	G
1	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7
2	Week No.	Season Influence	Holiday Influence	Competitors Sales	No. of Special Offerings	Advertising Budget	Number of Ads
3	Max = 104	Low = 0.25	Low = 0.25	Max = 10	Max = 5	Max = 49800	Max = 27
4	Min = 1	Medium = 0.5	Medium = 0.5	Min = 0	Min = 0	Min = 3400	Min = 1
5		High = 0.75	High = 0.75				
6		Very High = 0.9	Very High = 0.9				
7	1	0.9	0.9	3	0	47900	20
8	2	0.5	0.5	6	2	45600	23
9	3	0.25	0.25	8	2	10670	7
10	4	0.25	0.25	1	1	9900	4
11	5	0.25	0.25	5	2	36200	12
12	6	0.25	0.25	0	0	4300	2
13	7	0.25	0.5	7	1	7700	4
14	8	0.25	0.5	9	2	12300	6
15	9	0.25	0.5	3	1	5700	3
16	10	0.25	0.5	5	2	19100	8
17	11	0.25	0.5	2	0	17300	11
18	12	0.25	0.5	1	2	4200	2

Figure 5a.2

Now, we can see that Column A to column J in the *worksheet (Transform Data)* are all numerical. Apart from this transformation, we also need to “massage” the numeric data a little bit. This is because NN will learn better if there is uniformity in the data.

Thus we need to scale all the data into the value between 0 to 1. How do we do that?

Copy all data from Column A to J and paste them to Column L To U

Select *Scale Data* on the **nn\_Solve** menu (Figure 5a.3)

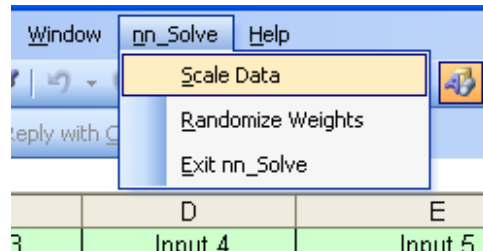


Figure 5a.3

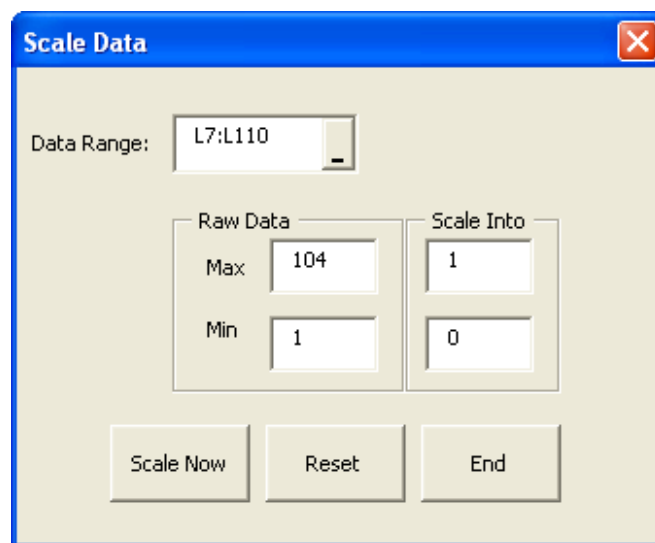


Figure 5a.4

Enter the reference for Input 1 (L7:L110) in the **Data Range**. Press the Tab key on your keyboard to exit. When you press Tab, **nn\_Solve** will automatically load the maximum (104) and the minimum (1) in the *Raw Data* frame *Min* and *Max* textbox. (see Figure 5a.4)

Enter the value 1 for maximum and 0 for minimum for the *Scale Into* frame. Of course you can change this.

Click on the **Scale Now** button. The raw data will be scaled

**nn\_Solve** will also automatically store the minimum (in cell L113) and the maximum (L112) value of the raw data in the last row and first column of the raw data. (see Figure 5a.5 below)

106	0.9612	0.25
107	0.9709	0.5
108	0.9806	0.5
109	0.9903	0.5
110	1.0000	0.75
111		
112	104	
113	1	
114		

Figure 5a.5

We also need to scale Input 4 in column O, Input 5 in column P, Input 6 in column Q, Input 7 in column R and the 2 Desire outputs in column T and U. I've scale all the input data for your convenience. See Figure 5a.6 below.

	M	N	O	P	Q	R	S	T	U
1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7		Target 1	Target 2
2	Season Influence	Holiday Influence	No. of Competitors Sales	No of Special Offers	Advertising Budget	Number of Ads		(Weekly Sales)	(Daily Sales)
3	Low = 0.25	Low = 0.25	Max = 10	Max = 5	Max = 49800	Max = 27			
4	Medium = 0.5	Medium = 0.5	Min = 0	Min = 0	Min = 3400	Min = 1			
5	High = 0.75	High = 0.75							
6	Very High = 0.9	Very High = 0.9							
97	0.5	0.5	0.55	1	0.319181034	0.584615385		0.679332929	0.15
98	0.5	0.5	0.64	1	0.292025862	0.480769231		0.348223166	0.10
99	0.5	0.5	0.46	0.28	0.105818966	0.134615385		0.736895088	0.16
100	0.25	0.5	0.37	0.28	0.435560345	0.480769231		0.388975136	0.11
101	0.25	0.25	0.64	0.46	0.138793103	0.169230769		0.384899939	0.11
102	0.25	0.5	0.55	0.28	0.121336207	0.169230769		0.421576713	0.11
103	0.25	0.75	0.46	0.46	0.15237069	0.169230769		0.435330503	0.11
104	0.25	0.75	0.64	0.46	0.15237069	0.169230769		0.413681019	0.11
105	0.25	0.75	0.46	0.46	0.142672414	0.169230769		0.689266222	0.15
106	0.25	0.9	0.73	0.46	0.412284483	0.342307692		0.345166768	0.10
107	0.5	0.9	0.46	0.46	0.191163793	0.169230769		0.773062462	0.16
108	0.5	0.5	0.64	0.28	0.804094828	1		0.73409339	0.16
109	0.5	0.9	0.28	0.28	0.740086207	0.861538462		0.98624621	0.19
110	0.75	0.9	0.55	0.1	0.709051724	0.792307692		0.95822923	0.19
111									
112			10	5	49800	27		369400	
113			0	0	3400	1		16042.85714	
114									

Figure 5a.6

We don't need to scale Input 2 and 3 as those values are already between 0 and 1.



## b) the neural network architecture

A neural network is a group of neurons connected together. Connecting neurons to form a NN can be done in various ways. This worksheet; column L to column AF actually contain the NN architecture shown below:

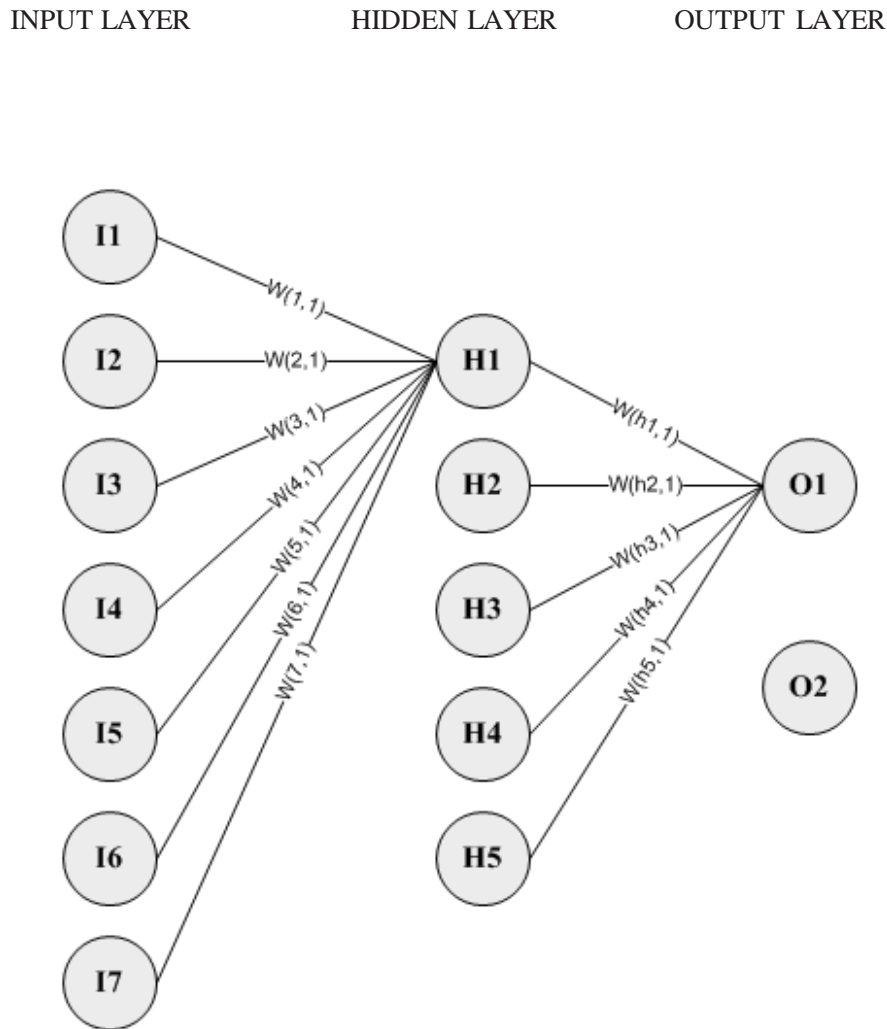


Figure 5a.7

There are 7 nodes or neuron on the input layer. 5 neurons on the hidden layer and 2 neurons on the output layer.

Those lines that connect the nodes are call weights. I only connect part of them. In reality all the weights are connected layer by layer, like Figure 5.1.

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired

outputs. Here we have 104 input patterns map to 104 desired or target outputs. We reserve 2 input patterns for testing later.

Like what you see from the Figure 5a.7 above, this NN model consists of three layers:

Input layer with 7 neurons.

Column L = Input 1 (I1); Column M = Input 2 (I2); Column N = Input 3 (I3);

Column O = Input 4 (I4) ; Column P = Input 5 (I5) ; Column Q = Input 6 (I6);

Column R = Input 7 (I7)

Hidden layer with 5 neurons.

Column Y = Hidden Node 1 (H1); Column Z = Hidden Node 2 (H2);

Column AA = Hidden Node 3 (H3); Column AB = Hidden Node 4 (H4)

Column AC = Hidden Node 5 (H5)

Output layer with 2 neurons.

Column AE = Output Node 1 (O1)

Column AF = Output Node 2 (O2)

Now let's talk about the weights that connection all the neurons together

Note that:

- The output of a neuron in a layer goes to all neurons in the following layer.
- In Figure 5a.7 we have 7 inputs node and 5 hidden nodes and 2 output nodes. Here the number of weights are  $(7 \times 5) + (5 \times 2) = 45$
- Each neuron has its own input weights.
- The output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input.

I have put the weights vector in one column W. So the weights are contain in cells:

From Input Layer to Hidden Layer

$w(1,1) = \$W\$7$  -> connecting I1 to H1

$w(2,1) = \$W\$8$  -> connecting I2 to H1

$w(3,1) = \$W\$9$  -> connecting I3 to H1

$w(4,1) = \$W\$10$  -> connecting I4 to H1

$w(5,1) = \$W\$11$  -> connecting I5 to H1

$w(6,1) = \$W\$12$  -> connecting I6 to H1

$w(7,1) = \$W\$13$  -> connecting I7 to H1

$w(1,2) = \$W\$14$  -> connecting I1 to H2

$w(2,2) = \$W\$15$  -> connecting I2 to H2

$w(3,2) = \$W\$16$  -> connecting I3 to H2

$w(4,2) = \$W\$17$  -> connecting I4 to H2

$w(5,2) = \$W\$18$  -> connecting I5 to H2

$w(6,2) = \$W\$19$  -> connecting I6 to H2

$w(7,2) = \$W\$20$  -> connecting I7 to H2

“ and

“ so

“ on

“

$w(1,5) = \$W\$35$  -> connecting I1 to H5

$w(2, 5) = \$W\$36$  -> connecting I2 to H5

$w(3, 5) = \$W\$37$  -> connecting I3 to H5

$w(4, 5) = \$W\$38$  -> connecting I4 to H5

$w(5, 5) = \$W\$39$  -> connecting I5 to H5

$w(6, 5) = \$W\$40$  -> connecting I6 to H5

$w(7, 5) = \$W\$41$  -> connecting I7 to H5

From Hidden Layer to Output Layer

$w(h1,1) = \$W\$42$  -> connecting H1 to O1

$w(h2, 1) = \$ W\$43$  -> connecting H2 to O1

$w(h3, 1) = \$ W\$44$  -> connecting H3 to O1

$w(h4, 1) = \$W\$45$  -> connecting H4 to O1

$w(h5, 1) = \$W\$46$  -> connecting H5 to O1

$w(h1,2) = \$W\$47$  -> connecting H1 to O2

$w(h2, 2) = \$ W\$48$  -> connecting H2 to O2

$w(h3, 2) = \$ W\$49$  -> connecting H3 to O2

$w(h4, 2) = \$W\$50$  -> connecting H4 to O2

$w(h5, 2) = \$W\$51$  -> connecting H5 to O2

After mapping the NN architecture to the worksheet and entering the input and desired

output data., it is time to see what is happening inside those nodes.

### c) simple mathematic operations inside the neural network model

The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, and could change per network configuration and type of data. In general the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance. You could start a network configuration using a single hidden layer, and add more hidden layers if you notice that the network is not learning as well as you like.

For this Credit Approval model, 1 hidden layer is sufficient. Select the cell Y7 (H1), you can see Figure 5a.8

X

X

X

X

X

X

X

X

X

X

X

X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X

X

X

X

X

X

X

X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

$$MSE = (SUM(AJ7:AK108)/102)/2$$

	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN
1	Output 1	Output 2	Desire 1	Desire 2		Error 1	Error 2		MSE	0.047275
2			Sales	Sales						
3			(WeeklySales)	(Daily Sales)						
4										
5										
6										
104	0.4802026	0.3341035	0.413681019	0.11		0.0044251	0.0503176			
105	0.4860332	0.3324723	0.689266222	0.15		0.0413036	0.0336045			
106	0.4849816	0.3522596	0.345166768	0.10		0.0195482	0.0636349			
107	0.4846318	0.3482774	0.773062462	0.16		0.0831923	0.0350249			
108	0.507527	0.3942475	0.73409339	0.16		0.0513323	0.0569713			
109				0.19						
110				0.19						
111										

Figure 5a.10

Our objective is to minimize the MSE. We need to change the weight of each connection so that the network produces a better approximation of the desired output.

In NN technical term, we call this step of changing the weights as TRAINING THE NEURAL NETWORK.

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection.

Phew, what craps is this back propagation!!!. Fortunately, you don't need to understand

this, if you use MS Excel Solver to build and train a neural network model.

#### d) Training NN as an Optimization Task Using Excel Solver

Training a neural network is, in most cases, an exercise in numerical optimization of a usually nonlinear function. Methods of nonlinear optimization have been studied for hundreds of years, and there is a huge literature on the subject in fields such as numerical analysis, operations research, and statistical computing, e.g., Bertsekas 1995, Gill, Murray, and Wright 1981. There is no single best method for nonlinear optimization. You need to choose a method based on the characteristics of the problem to be solved.

MS Excel's Solver is a numerical optimization add-in (an additional file that extends the capabilities of Excel). It can be fast, easy, and accurate.

For a medium size neural network model with moderate number of weights, various quasi-Newton algorithms are efficient. For a large number of weights, various conjugate-gradient algorithms are efficient. These two optimization method are available with Excel Solver

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence. Values between -1 to 1 will be the best starting weights.

Let's fill out the weight vector. The weights are contain in W7:W51. From the **nn\_Solve** menu, select *Randomize Weights* (see Figure 5a.11 below)

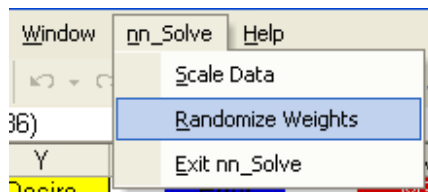


Figure 5a.11

Enter W7:W51 and click on the *Randomize Weights* button. W7:W51 will be filled out with values between -1 to 1. (see Figure 5a.12 below)

	W	X	Y	Z	AA	AB
1	Weights Vect		Hidden 1	Hidden 2	Hidden 3	Hidden 4
2						
3						
4						
5						
6						
7	0.157012224	w(1,1)				
8	-0.589895368	w(2,1)				
9	-0.508718491	w(3,1)				
10	0.351388574	w(4,1)				
11	0.585154295	w(5,1)				
12	-0.933403611	w(6,1)				
13	-0.800947666	w(7,1)				
14	0.696337819	w(1,2)				
15	0.645521402	w(2,2)				
16	-0.900149465	w(3,2)				
17	-0.043776512	w(4,2)				
18	-0.645976424	w(5,2)				

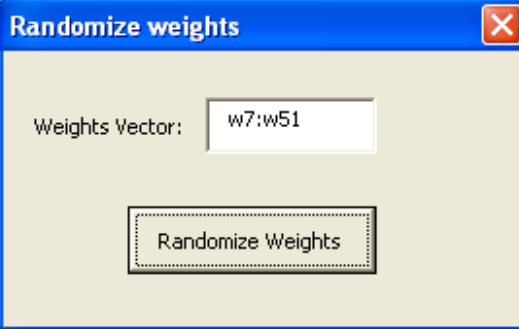


Figure 5a.12

The learning algorithm improves the performance of the network by gradually changing each weight in the proper direction. This is called an **iterative** procedure. Each iteration makes the weights slightly more efficient at separating the target from the nontarget examples. The iteration loop is usually carried out until no further improvement is being made. In typical neural networks, this may be anywhere from ten to ten-thousand iterations. Fortunately, we have Excel Solver. This tool has simplified neural network training so much ....

### Accessing Excel's Solver

To invoke Solver see page 134. After executing Tools: Solver . . . , you will be presented with the Solver Parameters dialog box below:

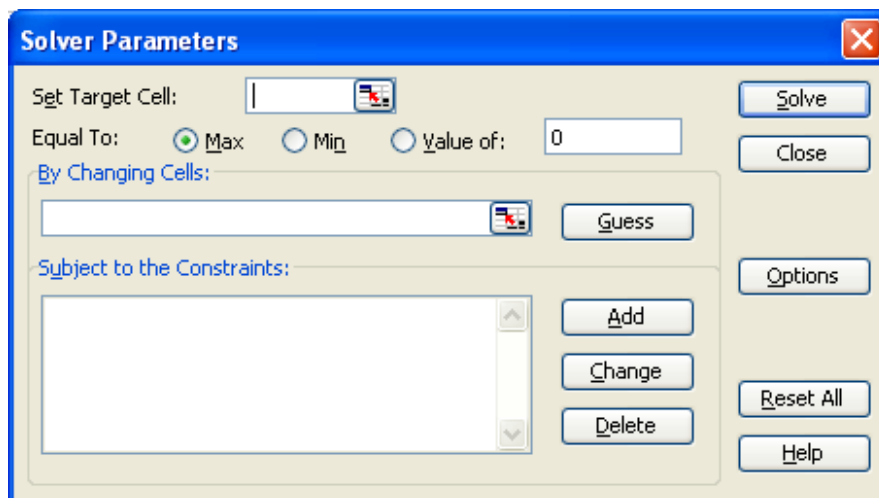


Figure 5a.16



Let us review each part of this dialog box, one at a time.

**Set Target Cell** is where you indicate the objective function (or goal) to be optimized. This cell must contain a formula that depends on one or more other cells (including at least one “changing cell”). You can either type in the cell address or click on the desired cell. Here we enter cell AN1.

In our NN model, the objective function is to minimize the Mean Squared Error. See Figure 5a.17 below

	AN1	fx =(SUM(AJ7:AK108)/102)/2			
	AJ	AK	AL	AM	AN
1	Error 1	Error 2		MSE	0.047275
2					
3					
4					

Figure 5a.17

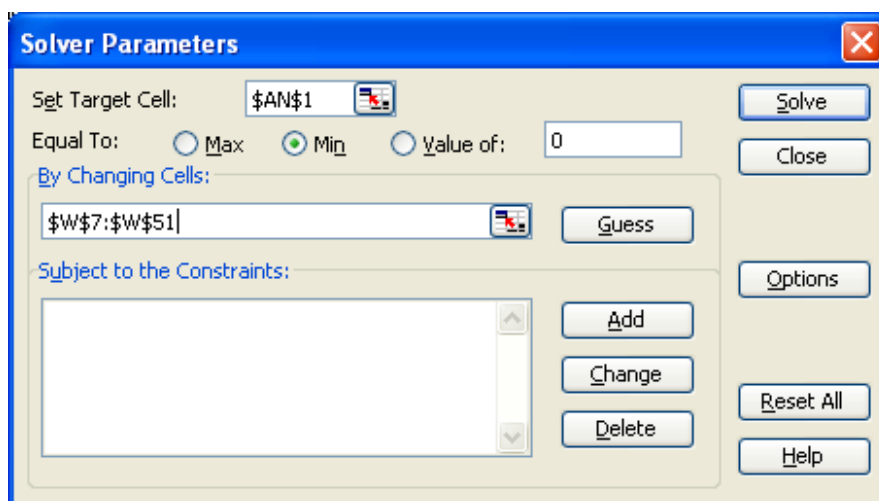


Figure 5a.18

**Equal to:** gives you the option of treating the Target Cell in three alternative ways. **Max** (the default) tells Excel to maximize the Target Cell and **Min**, to minimize it, whereas **Value** is used if you want to reach a certain particular value of the Target Cell by choosing a particular value of the endogenous variable.

Here, we select **Min** as we want to minimize the MSE

**By Changing Cells** permits you to indicate which cells are the adjustable cells (i.e., endogenous variables). As in the Set Target Cell box, you may either type in a cell address or click on a cell in the spreadsheet. Excel handles multivariable optimization problems by allowing you to include additional cells in the By Changing Cells box. Each noncontiguous choice variable is separated by a comma. If you use the mouse technique (clicking on the cells), the comma separation is automatic.

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

157

**After that select the Options.** This will allow you to adjust the way in which Solver approaches the solution.. (see Figure 5a.21)

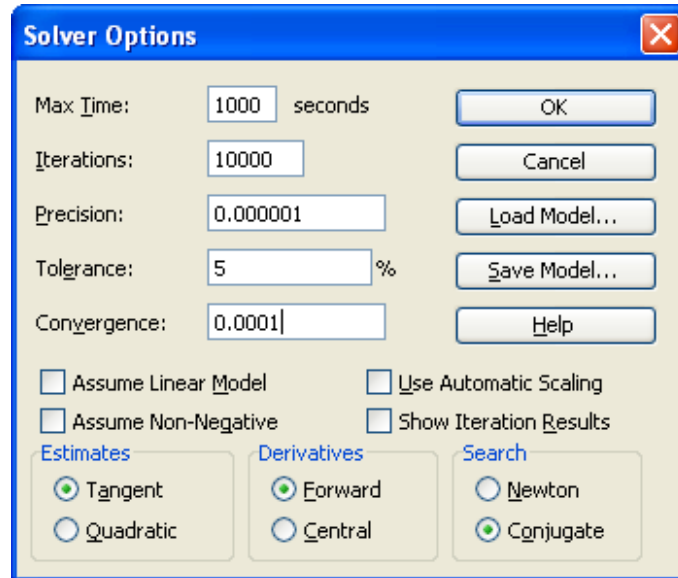


Figure 5a.21

As you can see, a series of choices are included in the Solver Options dialog box that direct Solver's search for the optimum solution and for how long it will search. These options may be changed if Solver is having difficulty finding the optimal solution. Lowering the Precision, Tolerance, and Convergence values slows down the algorithm but may enable Solver to find a solution.

For a neural network model, you can set :

- i. Max Time: 1000 seconds
- ii. Iterations: 10000
- iii. Precision: 0.000001
- iv. Tolerance: 5%
- v. Convergence: 0.0001

Select **Conjugate** as the *Search* method. This proves to be very effective in minimizing the Mean Squared Error.

The Load and Save Model buttons enable you to recall and keep a complicated set of constraints or choices so that you do not have to reenter them every time.

Clicking OK to return to the Solver Parameters dialog box.

X  
X  
X

This part of the book is not available for viewing

[illegible]

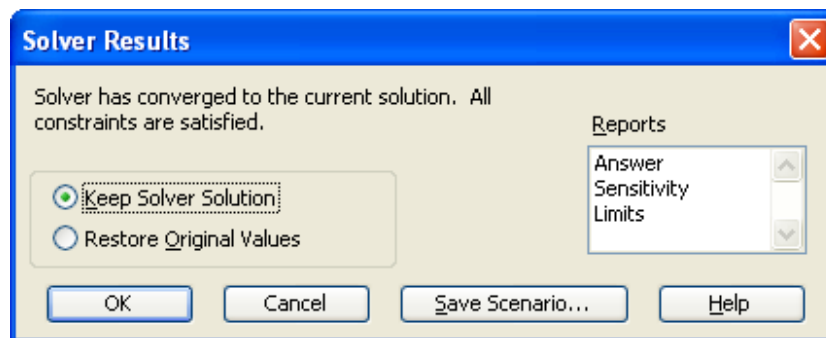


Figure 5a.24

A message will appear after Solver has converged (see Figure 5a.24). In this case, Excel reports that “Solver has converged to the current solution. All constraints are satisfied.” This is good news!

Sometime, the Mean Square Error is not satisfactory and Solver unable to find the solution at one go. If this is the case then you, Keep the Solver solution and run Solver again. Follow the step discussed above. From experience, usually you will need to run Solver a few times before Solver arrive at a satisfactory Mean Square Error. (Note: value less than 0.01 will be very good)

Bad news is a message like, “Solver could not find a solution.” If this happens, you must diagnose, debug, and otherwise think about what went wrong and how it could be fixed. The two quickest fixes are to try different initial weights values and to add bigger or smaller constraints to the weights.

Or you may change the network architecture by adding more hidden nodes.

From the Solver Results dialog box, you elect whether to have Excel write the solution it has found into the Changing Cells (i.e., Keep Solver Solution) or whether to leave the spreadsheet alone and NOT write the value of the solution into the Changing Cells (i.e., Restore Original Values). When Excel reports a successful run, you would usually want it to Keep the Solver Solution.

On the right-hand side of the Solver Results dialog box, Excel presents a series of reports. The Answer, Sensitivity, and Limits reports are additional sheets inserted into the current workbook. They contain diagnostic and other information and should be selected if Solver is having trouble finding a solution.

*It is important to understand that a saved Excel workbook will remember the information included in the last Solver run.*

### **e) using the trained model for forecasting**

After all the training and the MSE is below 0.01, its now time for us to predict. Goto the row 109 of the Sales Forecasting spreadsheet. Remember, we have save 2 rows of data for testing i.e. row 109 and 110.

	L	M	N	O	P	
1	Input 1	Input 2	Input 3	Input 4	Input 5	
2	Wks No.	Season Influence	Holiday Influence	No. of Competitors Sales	No of Special Offers	Ac
3	Max = 104	Low = 0.25	Low = 0.25	Max = 10	Max = 5	
4	Min = 1	Medium = 0.5	Medium = 0.5	Min = 0	Min = 0	
5		High = 0.75	High = 0.75			
6		Very High = 0.9	Very High = 0.9			
106	0.9612	0.25	0.9	0.73	0.46	
107	0.9709	0.5	0.9	0.46	0.46	
108	0.9806	0.5	0.5	0.64	0.28	
109	0.9903	0.5	0.9	0.28	0.28	
110	1.0000	0.75	0.9	0.55	0.1	
111						

Figure 5a.25

Then goto Y108. Select Y108:AF108. (see Figure 5a.26 below).

	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1		Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5		Output 1	Output 2	Desire 1	Desire 2
2										Sales	Sales
3										(WeeklySales)	(Daily Sales)
4											
5											
6											
105		0.99997	3.4E-06	0.07214	0.99998	0.99551		0.572526	0.040036	0.654740247	0.05
106		1	0.51294	0.78196	1	0.99663		0.353278	0.019952	0.27240752	0.00
107		0.99999	1.2E-08	0.00176	0.99824	0.99935		0.628927	0.046036	0.74784718	0.07
108		1	1	0.00224	1	0.89486		0.682341	0.050544	0.704548211	0.06
109										0.984718011	0.10
110										0.953588033	0.10

Figure 5a.26

After that, you fill down until row 110. (see Figure 5a.27)

	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1		Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5		Output 1	Output 2	Desire 1	Desire 2
2										Sales	Sales
3										(WeeklySales)	(Daily Sales)
4											
5											
6											
105		0.99997	3.4E-06	0.07214	0.99998	0.99551		0.572526	0.040036	0.654740247	0.05
106		1	0.51294	0.78196	1	0.99663		0.353278	0.019952	0.27240752	0.00
107		0.99999	1.2E-08	0.00176	0.99824	0.99935		0.628927	0.046036	0.74784718	0.07
108		1	1	0.00224	1	0.89486		0.682341	0.050544	0.704548211	0.06
109		1	0.99761	1E-05	0.98292	0.99557		0.903441	0.102692	0.984718011	0.10
110		1	0.78521	1E-05	0.94181	0.99172		0.84799	0.080876	0.953588033	0.10
111											

Figure 5a.27

	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1		Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5		Output 1	Output 2	Desire 1	Desire 2
2										Sales	Sales
3										(Weekly Sales)	(Daily Sales)
4											
5											
6											
105		0.99997	3.4E-06	0.07214	0.99998	0.99551		0.572526	0.040036	0.654740247	0.05
106		1	0.51294	0.78196	1	0.99663		0.353278	0.019952	0.27240752	0.00
107		0.99999	1.2E-08	0.00176	0.99824	0.99935		0.628927	0.046036	0.74784718	0.07
108		1	1	0.00224	1	0.89486		0.682341	0.050544	0.704548211	0.06
109		1	0.99761	1E-05	0.98292	0.99557		0.903441	0.102692	0.984718011	0.10
110		1	0.78521	1E-05	0.94181	0.99172		0.84799	0.080876	0.953588033	0.10
111											

Figure 5a.28

So, for row 109 we have 0.903441 for predicted Output 1 (AE109) and 0.102692 for predicted Output 2 (AF109). (see Figure 5a.28).

Row 110 we have 0.84799 for predicted Output 1 (AE110) and 0.080876 for predicted Output 2 (AF110).

We need to scale these number back to raw data before they have any meaning to us.

Select *Scale Data* for the **nn\_Solve** menu (see Figure 5a.29 below)

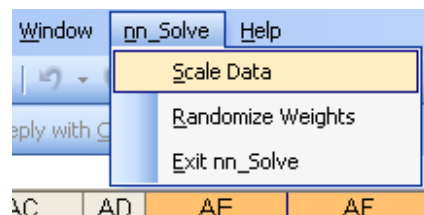


Figure 5a.29

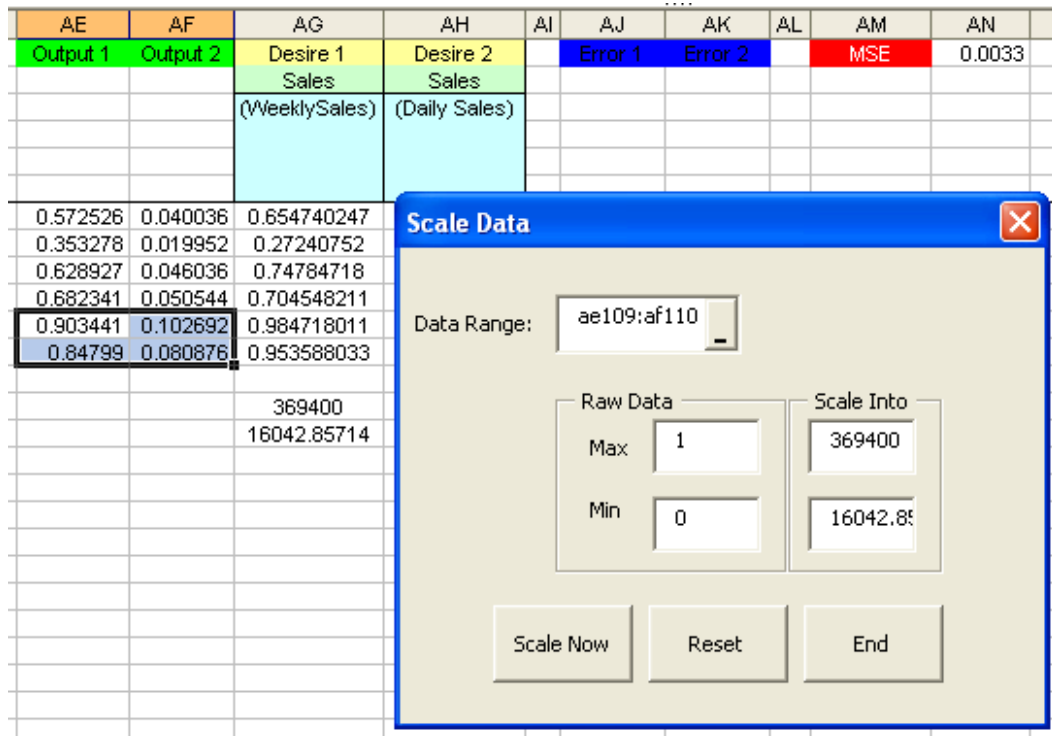


Figure 5a.30

Enter AE109:AF110 in the *Data Range*. As we are reversing what we did just now when we scale the raw data to the range 0 to 1. Now the raw data maximum become 1 and minimum become 0.

As we have automatically save the maximum and minimum of the raw data initially when we scale them, now we use them as the maximum and minimum values to be scaled into (See above Figure 5a.30). Enter 369400 (in cell AG112) as the maximum and 16042.85714 (in cell AG113) as the minimum. Click on *Scale Now*.

	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH
1		Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5		Output 1	Output 2	Desire 1	Desire 2
2										Sales	Sales
3										(WeeklySales)	(Daily Sales)
4											
5											
6											
105		0.99997	3.4E-06	0.07214	0.99998	0.99551		0.572526	0.040036	0.654740247	0.05
106		1	0.51294	0.78196	1	0.99663		0.353278	0.019952	0.27240752	0.00
107		0.99999	1.2E-08	0.00176	0.99824	0.99935		0.628927	0.046036	0.74784718	0.07
108		1	1	0.00224	1	0.89486		0.682341	0.050544	0.704548211	0.06
109		1	0.99761	1E-05	0.98292	0.99557		335280	52329.66	0.984718011	0.10
110		1	0.78521	1E-05	0.94181	0.99172		315686.2	44620.95	0.953588033	0.10
111											
112								1		369400	
113								0		16042.85714	

Figure 5a.31



So our predicted first weekly sales is 335280 as in AE109 and first daily sales is 52329.66 as in AF109. And our predicted second weekly sales is 315686.2 as in AE110 and second daily sales is 44620.95 as in AF110. (see Figure 5a.31 above)

The desire weekly sales is 364000 (see I109) and 353000 (see I110) respectively. Whereas

desire daily sales is 52000 (see cell J109) and 50428.57 (see cell J110). The predicted values that we have are within 10% error tolerance of these desire values. So it is acceptable.

Of course when you do the training on your own, you will get slightly different result because of the MSE error that you have derived. The results here are based on the MSE of 0.00391.

There you go. You have successfully use neural network to predict the weekly and daily sales.

### 3) Predicting the DJIA weekly price.

Neural networks are an emerging and challenging computational technology and they offer a new avenue to explore the dynamics of a variety of financial applications. They can simulate fundamental and technical analysis methods using fundamental and technical indicators as inputs. Consumer price index, foreign reserve, GDP, export and import volume, etc., could be used as inputs. For technical methods, the delayed time series, moving averages, relative strength indices, etc., could be used as inputs of neural networks to mine profitable knowledge.

In this example, I have built a neural network to forecast the weekly prices of the DJIA by using the moving averages.

#### a) Selecting and transforming data

Open the *Workbook(Dow\_Weekly)* in folder Chapter 5 and bring up *worksheet (Raw Data)*. Here, a real life example is use. This data is taken from the DJIA weekly prices, from the period from 22 April 2002 to 15 Oct 2007. There are 5 input factors and 1 desire output (end result), The input factors consist of 5-days Moving Average, 10-days Moving Average, 20-days Moving Average, 60-days Moving Average and 120-days Moving Average. The desire output is the next week DJIA price. There are 287 rows of input patterns in this model. (see Figure 5b.1a)

	A	B	C	D	E	F	G	
1		Input 1	Input 2	Input 3	Input 4	Input 5	Desire	
2	Date	MA 5	MA 10	MA 20	MA 60	MA 120		
3	4/22/2002	10206.85	10297.86	10118.79	10127.81	10409.398	10006.63	
4	4/29/2002	10127.38	10301.71	10128.57	10120.15	10396.765	9939.92	
5	5/6/2002	10061.04	10258.82	10123.8	10108.41	10381.907	10353.08	
6	5/13/2002	10093.49	10236.88	10134.6	10117.24	10374.418	10104.26	
7	5/20/2002	10062.92	10186.58	10126.83	10127.23	10369.13	9925.25	
8	5/28/2002	10065.83	10136.34	10123.71	10128	10360.475	9589.67	
9	6/3/2002	9982.436	10054.91	10114.6	10124.64	10353.512	9474.21	
10	6/10/2002	9889.294	9975.167	10096.31	10113.77	10347.301	9253.79	
11	6/17/2002	9669.436	9881.464	10063.64	10091.66	10342.232	9243.26	
12	6/24/2002	9497.236	9780.079	10038.59	10065.55	10332.866	9379.5	
13	7/1/2002	9388.086	9726.957	10012.41	10039.36	10328.288	8684.53	
14	7/8/2002	9207.058	9594.747	9948.229	10003.74	10312.365	8019.26	
15	7/15/2002	8916.068	9402.681	9830.749	9949.035	10286.587	8264.39	

Figure 5b.1a

There are 287 rows of data. The first 286 rows will be used as training data. The last 1 row will be used for testing our prediction later.

We need to “massage” the numeric data a little bit. This is because NN will learn better if there is uniformity in the data.

Thus we need to scale all the data into the value between 0 to 1. How do we do that?

Goto *worksheet(Transform Data)*. Copy all data from Column B to G and paste them to Column J To O.

Select *Scale Data* on the **nn\_Solve** menu. (see Figure 5b.1)

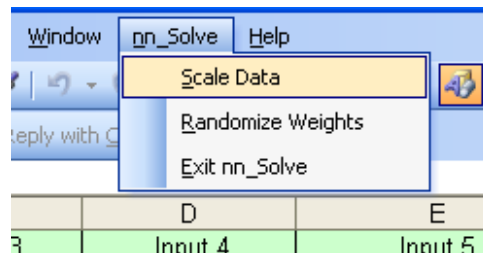


Figure 5b.1

Thus we need to convert the data in the range J3:O289 (see Figure 5b.2 ). That is 6 columns times 287 rows in the worksheet (Transform Data). Enter J3:O289 into the **Data Range** edit box. Press the Tab key on your keyboard to exit. When you press Tab, **nn\_Solve** will automatically load the maximum (14,093.08) and the minimum (7528.4) in the *Raw Data* frame *Min* and *Max* textbox.

Enter the value 1 for maximum and 0.1 for minimum for the *Scale Into*. Of course you can change this. It is advisable not to enter the value 0 as the minimum as it represent nothing. (see Figure 5b.2 below)

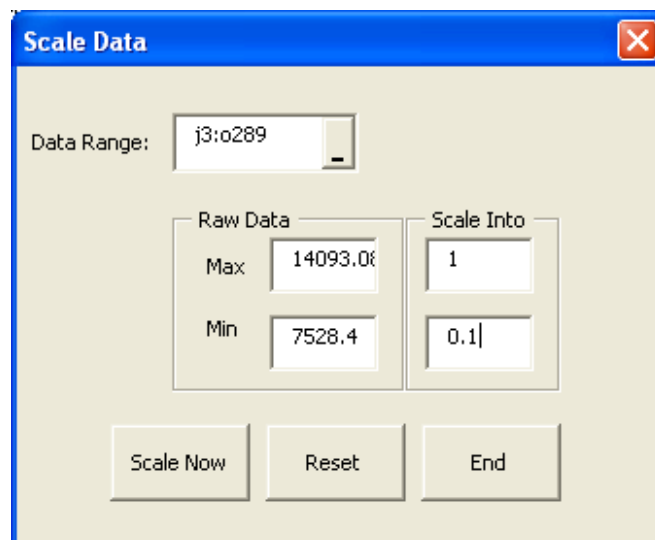


Figure 5b.2

Click on the *Scale Now* button, **nn\_Solve** will also automatically store the minimum (in cell J292) and the maximum (cell J291) value of the raw data in the last row and first column of the raw data. (see Figure 5b.3 below). We may need these numbers later. I've convert all the values for you already in the worksheet (Transform Data).

	I	J	K	L	M	N	O
1		Input 1	Input 2	Input 3	Input 4	Input 5	Desire
2	Date	MA 5	MA 10	MA 20	MA 60	MA 120	
275	7/9/2007	0.922674	0.9158987	0.843137	0.715143	0.60005686	0.9631361
276	7/16/2007	0.92912	0.9238939	0.856367	0.721674	0.60443227	0.8739299
277	7/23/2007	0.926233	0.9194602	0.863901	0.726796	0.60799188	0.8612012
278	7/30/2007	0.919326	0.9145038	0.872062	0.73261	0.61191998	0.86998
279	8/6/2007	0.907988	0.9079754	0.877839	0.738259	0.61583211	0.845537
280	8/13/2007	0.882757	0.9027153	0.883359	0.743565	0.61949638	0.8912041
281	8/20/2007	0.86837	0.8987454	0.88959	0.749223	0.62334713	0.8879854
282	8/27/2007	0.871182	0.898707	0.895269	0.754979	0.62743164	0.850762
283	9/4/2007	0.869094	0.8942096	0.896422	0.761006	0.63078477	0.9009
284	9/10/2007	0.875278	0.8916328	0.898871	0.767542	0.63446605	0.9584306
285	9/17/2007	0.897856	0.8903066	0.903103	0.774144	0.63873031	0.9699224
286	9/24/2007	0.9136	0.8909852	0.90744	0.780885	0.64302476	0.9958764
287	10/1/2007	0.935178	0.9031799	0.91132	0.788446	0.64739529	1
288	10/8/2007	0.965026	0.9170598	0.915782	0.79533	0.65221306	0.9130102
289	10/15/2007	0.967448	0.9213628	0.914669	0.801012	0.65629878	0.9365011
290							
291		14093.08					
292		7528.4					
293							

Figure 5b.3

After converting the values, its now time to build the neural network infrastructure.

## b) the neural network architecture

A neural network is a group of neurons connected together. Connecting neurons to form a NN can be done in various ways. This worksheet; column J to column X actually contain the NN architecture shown below:

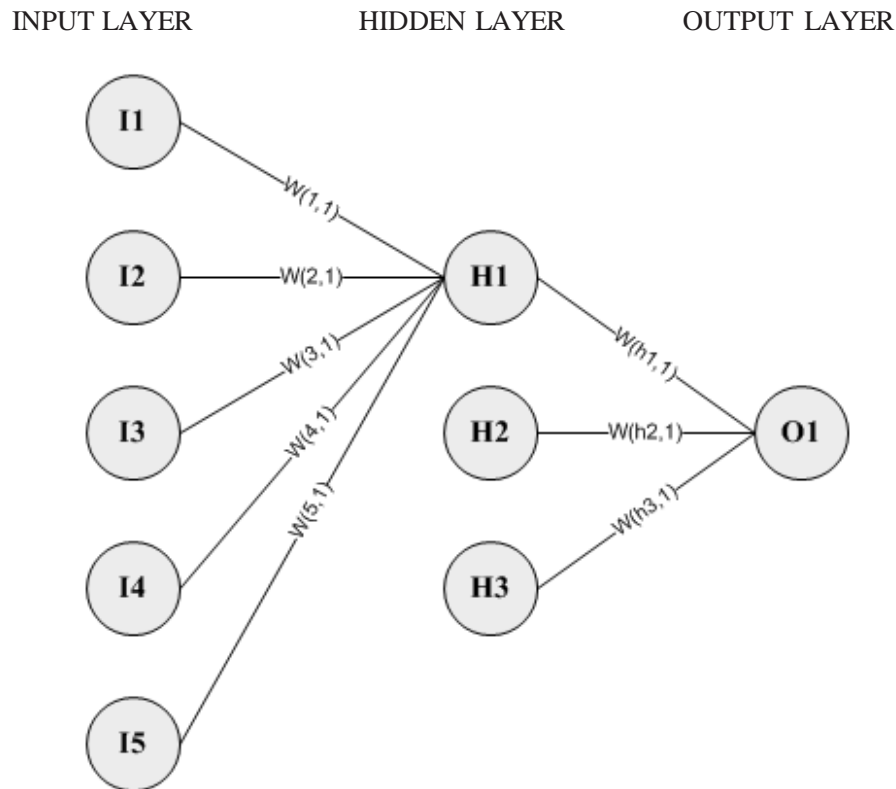


Figure 5b.4

There are 5 nodes or neuron on the input layer. 3 neurons on the hidden layer and 1 neuron on the output layer.

Those lines that connect the nodes are call weights. I only connect part of them. In reality all the weights are connected layer by layer, like Figure 5.1

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. Here we have 286 input patterns map to 286 desired or target outputs. We reserve 1 input pattern for testing later.

Like what you see from the Figure 5b.4 above, this NN model consists of three layers:

Input layer with 5 neurons.

Column J = Input 1 (I1); Column K = Input 2 (I2); Column L = Input 3 (I3);

Column M = Input 4 (I4) ; Column N = Input 5 (I5)

Hidden layer with 3 neurons.

Column T = Hidden Node 1 (H1)

Column U = Hidden Node 2 (H2)

Column V = Hidden Node 3 (H3)

Output layer with 1 neuron.

Column X = Output Node 1 (O1)

Now let's talk about the weights that connection all the neurons together

Note that:

- The output of a neuron in a layer goes to all neurons in the following layer.
- We have 5 inputs node and 3 hidden nodes and 1 output node. Here the number of weights are  $(5 \times 3) + (3 \times 1) = 18$
- Each neuron has its own input weights.
- The output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input.

I have put the weights vector in one column Q. So the weights are contain in cells:

From Input Layer to Hidden Layer

$w(1,1) = Q_3 \rightarrow$  connecting I1 to H1

$w(2,1) = Q_4 \rightarrow$  connecting I2 to H1

$w(3,1) = Q_5 \rightarrow$  connecting I3 to H1

$w(4,1) = Q_6 \rightarrow$  connecting I4 to H1

$w(5,1) = Q_7 \rightarrow$  connecting I5 to H1

$w(1,2) = Q_8 \rightarrow$  connecting I1 to H2

$w(2,2) = Q_9 \rightarrow$  connecting I2 to H2

$w(3,2) = Q_{10} \rightarrow$  connecting I3 to H2

$w(4,2) = Q_{11} \rightarrow$  connecting I4 to H2

$w(5,2) = Q_{12} \rightarrow$  connecting I5 to H2

“ and

“ so

“ on

“

$w(1,5) = Q_{13} \rightarrow$  connecting I1 to H5

$w(2,5) = Q_{14} \rightarrow$  connecting I2 to H5

$w(3,5) = Q_{15} \rightarrow$  connecting I3 to H5

$w(4,5) = Q_{16} \rightarrow$  connecting I4 to H5

$w(5, 5) = 17$  -> connecting I5 to H5

From Hidden Layer to Output Layer

$w(h1, 1) = 18$  -> connecting H1 to O1

$w(h2, 1) = 19$  -> connecting H2 to O1

$w(h3, 1) = 20$  -> connecting H3 to O1

After mapping the NN architecture to the worksheet and entering the input and desired output data., it is time to see what is happening inside those nodes.

### c) simple mathematic operations inside the neural network model

The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, and could change per network configuration and type of data. In general the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance. You could start a network configuration using a single hidden layer, and add more hidden layers if you notice that the network is not learning as well as you like.

For this model, 1 hidden layer is sufficient. Select the cell T3 (H1), you can see

X

X

X

X

X

X

X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X

X

X

X

X

X

[illegible]

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X



Figure 5b.5 above indicate that the desire Output 1 is 0.4397587 i.e. Y3. The actual output is 0.529467 i.e. X3. Since 0.529467 is not what we want, we minus this with the desire output. After that we square the error to get a positive value

Error =  $(Y3 - X3)^2$  ; we get 0.008048 (in AA3)

The closer the actual output of the network matches the desired output, the better. This is only one pattern error. Since we have using 286 rows of patterns, we fill down the formula again from row 3 until row 288 in this spreadsheet. Remember, we save the last 1 row for testing our model later. Sum up all the error and take the average.

MSE =  $(\text{SUM}(\text{AA3:AA288})/286)$

Our objective is to minimize the MSE. We need to change the weight of each connection so that the network produces a better approximation of the desired output. In NN technical term, we call this step of changing the weights as TRAINING THE NEURAL NETWORK.

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW. The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection.

Phew, what craps is this back propagation!!!. Fortunately, you don't need to understand this, if you use MS Excel Solver to build and train a neural network model.

#### **d) Training NN as an Optimization Task Using Excel Solver**

Training a neural network is, in most cases, an exercise in numerical optimization of a usually nonlinear function. Methods of nonlinear optimization have been studied for hundreds of years, and there is a huge literature on the subject in fields such as numerical analysis, operations research, and statistical computing, e.g., Bertsekas 1995, Gill,

Murray, and Wright 1981. There is no single best method for nonlinear optimization. You need to choose a method based on the characteristics of the problem to be solved.

MS Excel's Solver is a numerical optimization add-in (an additional file that extends the capabilities of Excel). It can be fast, easy, and accurate. For a medium size neural network model with moderate number of weights, various quasi-Newton algorithms are efficient. For a large number of weights, various conjugate-gradient algorithms are efficient. These two optimization methods are available with Excel Solver. To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence. Values between -1 to 1 will be the best starting weights. Let's fill out the weight vector. The weights are contained in Q3:Q20. From the **nn\_Solve** menu, select *Randomize Weights* (see Figure 5b.6)

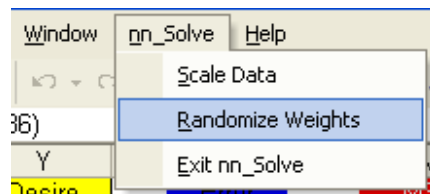


Figure 5b.6

Enter Q3:Q20 and click on the *Randomize Weights* button. Q3:Q20 will be filled out with values between -1 to 1. (see Figure 5b.7 below)

	Q	R	S	T	U	V
1	Weights			Hidden 1	Hidden 2	Hidden 3
2						
3	0.7181352	w(1,1)		0.604287	0.336352	0.524511
4	0.482717	w				
5	0.1305	w				
6	0.4875662	w				
7	-0.8598777	w				
8	-0.8638105	w				
9	-0.0882181	w				
10	-0.5813773	w				
11	-0.9777542	w				
12	0.9639401	w				
13	-0.4938334	w				
14	0.9025586	w				
15	-0.7308313	w(3,3)		0.558335	0.381022	0.524775
16	-0.3730202	w(4,3)		0.549486	0.390109	0.524527
17	0.8056217	w(5,3)		0.54144	0.398764	0.524934
18	0.6384704	w(h1,1)		0.535786	0.405114	0.525652
19	0.6586012	w(h2,1)		0.53454	0.406825	0.524781
20	-0.8328214	w(h3,1)		0.537394	0.403908	0.521731
21				0.537996	0.403751	0.52042

Figure 5b.7

The learning algorithm improves the performance of the network by gradually changing each weight in the proper direction. This is called an **iterative** procedure. Each iteration makes the weights slightly more efficient at separating the target from the nontarget examples. The iteration loop is usually carried out until no further improvement is being made. In typical neural networks, this may be anywhere from ten to ten-thousand iterations. Fortunately, we have Excel Solver. This tool has simplified neural network training so much.

### *Accessing Excel's Solver*

To invoke Solver goto page 134. After executing Tools: Solver . . . , you will be presented with the Solver Parameters dialog box below:

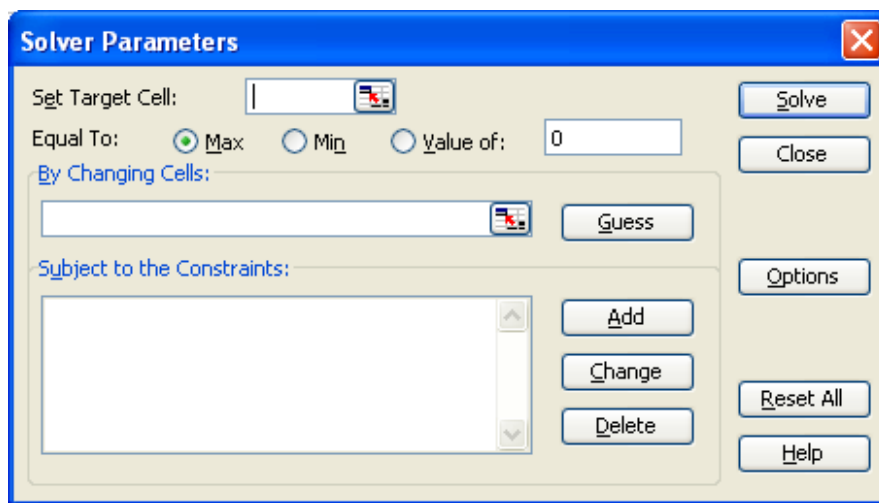


Figure 5b.11

Let us review each part of this dialog box, one at a time.

**Set Target Cell** is where you indicate the objective function (or goal) to be optimized. This cell must contain a formula that depends on one or more other cells (including at least one “changing cell”). You can either type in the cell address or click on the desired cell. Here we enter cell AD1.

In our NN model, the objective function is to minimize the Mean Squared Error (AD1). See Figure 5b.12 and 5b.13 below

X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

[illegible]

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X

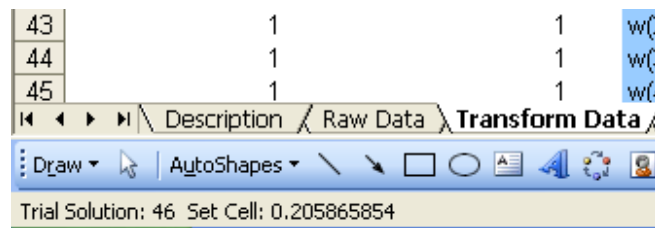


Figure 5b.19

When Solver start optimizing, you will see the *Trial Solution* at the bottom left of your spreadsheet. See Figure 5b.19 above.

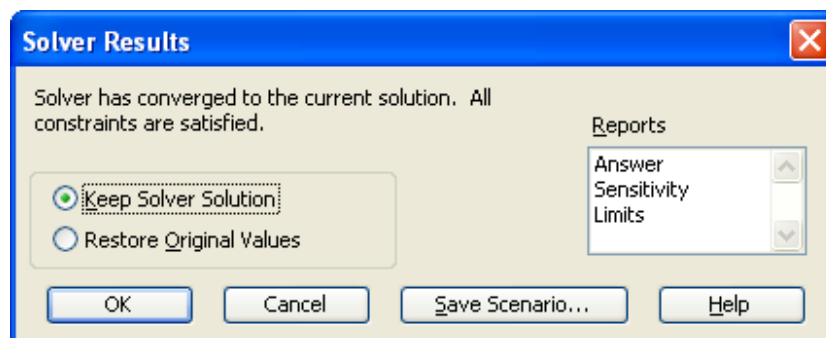


Figure 5b.20

A message appears when Solver converged (see Figure 5b.20). In this case, Excel reports that “Solver has converged to the current solution. All constraints are satisfied.” This is good news!

Sometime, the Mean Square Error is not satisfactory and Solver unable to find the solution at one go. If this is the case then you, Keep the Solver solution and run Solver again. Follow the step discussed above. From experience, usually you will need to run Solver a few times before Solver arrive at a satisfactory Mean Square Error. (Note: value less than 0.01 will be satisfactory)

Bad news is a message like, “Solver could not find a solution.” If this happens, you must diagnose, debug, and otherwise think about what went wrong and how it could be fixed. The two quickest fixes are to try different initial weights values and to add bigger or smaller constraints to the weights.

Or you may change the network architecture by adding more hidden nodes.

From the Solver Results dialog box, you elect whether to have Excel write the solution it has found into the Changing Cells (i.e., Keep Solver Solution) or whether to leave the spreadsheet alone and NOT write the value of the solution into the Changing Cells (i.e., Restore Original Values). When Excel reports a successful run, you would usually want it to Keep the Solver Solution.

On the right-hand side of the Solver Results dialog box, Excel presents a series of reports. The Answer, Sensitivity, and Limits reports are additional sheets inserted into the current

workbook. They contain diagnostic and other information and should be selected if Solver is having trouble finding a solution.

**e) using the trained model for forecasting**

After all the training and the MSE is below 0.01, its now time for us to predict. Goto the row 288 of the Dow\_Weekly spreadsheet. Remember, we have save 1 row of data for testing i.e. row 289

	S	T	U	V	W	X	Y
1		Hidden 1	Hidden 2	Hidden 3		Output	Desire
2							
285		0.992519	5.04E-09	0.790843		0.922078	0.9729301
286		0.99295	4.36E-09	0.79402		0.928817	0.9962888
287		0.993345	3.52E-09	0.797625		0.93595	1
288		0.993777	2.71E-09	0.80205		0.943863	0.9217092
289							
290							

Figure 5b.21

Then goto T288. Select T288:X288. See Figure 5b.22 below.

	S	T	U	V	W	X	Y
1		Hidden 1	Hidden 2	Hidden 3		Output	Desire
2							
285		0.992519	5.04E-09	0.790843		0.922078	0.9729301
286		0.99295	4.36E-09	0.79402		0.928817	0.9962888
287		0.993345	3.52E-09	0.797625		0.93595	1
288		0.993777	2.71E-09	0.80205		0.943863	0.9217092
289							
290							
291							

Figure 5b.22

After that, you fill down until row 289 (see Figure 5b.23 below)

	S	T	U	V	W	X	Y
1		Hidden 1	Hidden 2	Hidden 3		Output	Desire
2							
285		0.992519	5.04E-09	0.790843		0.922078	0.9729301
286		0.99295	4.36E-09	0.79402		0.928817	0.9962888
287		0.993345	3.52E-09	0.797625		0.93595	1
288		0.993777	2.71E-09	0.80205		0.943863	0.9217092
289		0.993988	2.55E-09	0.803408		0.946003	
290							
291							

Figure 5b.23



So, for row 109 we have 0.946003 for predicted Output 1 (X289).

We need to scale this number back to raw data before they have any meaning to us.

Select *Scale Data* from the **nn\_Solve** menu.

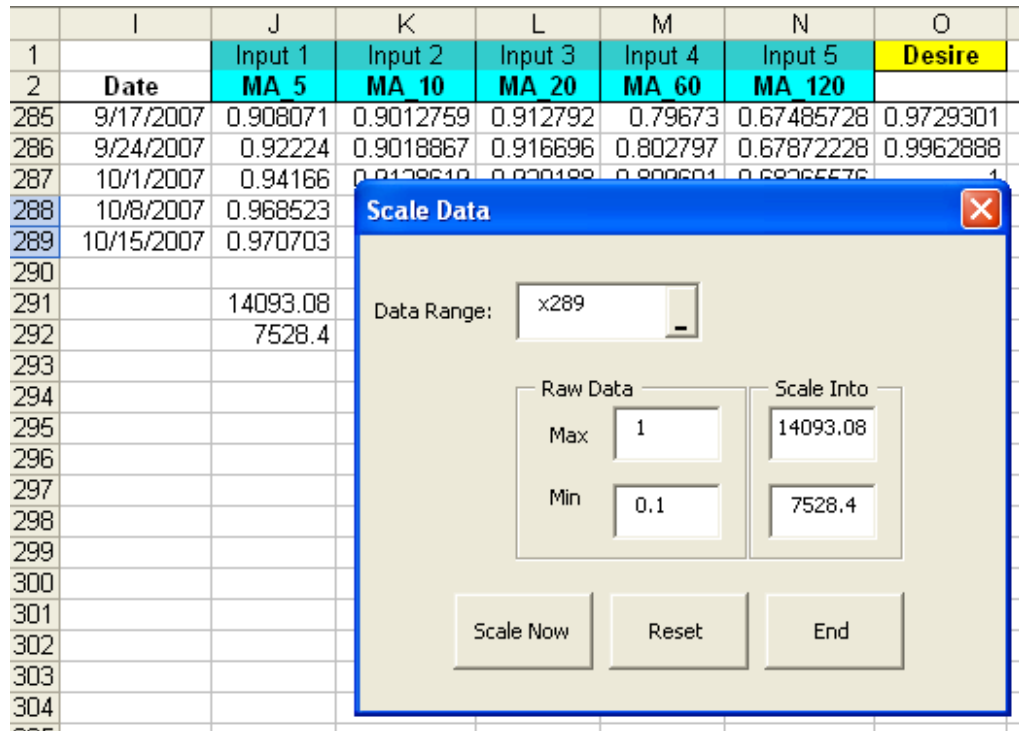


Figure 5b.24

Enter X289 in the *Data Range*. As we are reversing what we did just now when we scale the raw data to the range 0 to 1. Now the raw data maximum become 1 and minimum become 0.1 (see Figure 5b.24)

As we have save the maximum (14093.080 and minimum (7528.4) of the raw data, now we use them as the maximum and minimum values to be scaled into. Click on *Scale Now*.

	T	U	V	W	X	Y
1	Hidden 1	Hidden 2	Hidden 3		Output	Desire
2						
285	0.992519	5.04E-09	0.790843		0.922078	0.9729301
286	0.99295	4.36E-09	0.79402		0.928817	0.9962888
287	0.993345	3.52E-09	0.797625		0.93595	1
288	0.993777	2.71E-09	0.80205		0.943863	0.9217092
289	0.993988	2.55E-09	0.803408		13860.05	
290						

Figure 5b.25

So our predicted DJIA weekly price is 13860.05 as in X289 (see Figure 5b.25). The actual price is 13879.39 (cell G289). Of course when you do the training on your own, you will get slightly different result because of the MSE error that you have derived. The results here are based on the MSE of 0.001265

There you go. You have successfully use neural network to predict the next week price of the DJIA.. You can also use other factors as inputs data. For example, like Volume, Bollinger bands, RSI, ADX and etc. The sky is the limit and use your creativity.

#### 4) Predicting Real Estate Value

Our objective is to use neural network to forecast the value of a residential property in a suburban area.

##### a) Selecting and transforming data

Open the *workbook(Real\_Estate)* in Chapter 5 and bring up *worksheet (Raw Data)*. Here we have 499 inputs patterns and desire outputs. There are 13 input factors and 1 desire output (end result). Goto the *worksheet (Description)* to see the explanation of each of the input factors.

We need to “massage” these numeric data a little bit. This is because NN will learn better if there is uniformity in the data. Goto the *worksheet (Transform Data)*

Thus we need to scale all the data into the value between 0 to 1. How do we do that?

Copy all data from Column A to O and paste them to Column Q To AE. The first column we need to scale is Column Q (Input 1)

Select *Scale Data* on the **nn\_Solve** menu

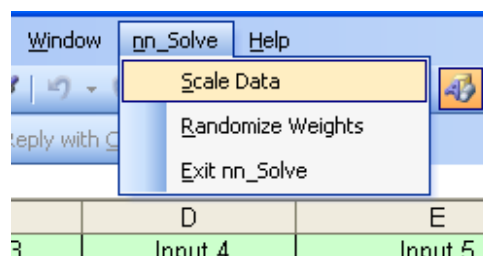
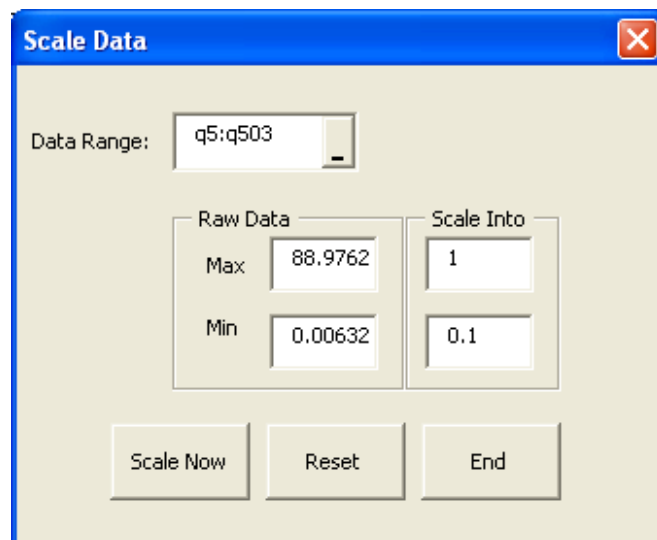


Figure 5c.1

Enter the reference for Input 1 (Q5:Q503) in the *Data Range*. Press the Tab key on your keyboard to exit. When you press Tab, **nn\_Solve** will automatically load the maximum (88.9762) and the minimum (0.00632) in the *Raw Data* frame **Min** and **Max** textbox. (see Figure 5c.2)



The 'Scale Data' dialog box has a title bar with a close button. It contains a 'Data Range' field with the text 'q5:q503'. Below this are two columns of input fields. The 'Raw Data' column has 'Max' set to 88.9762 and 'Min' set to 0.00632. The 'Scale Into' column has a top field set to 1 and a bottom field set to 0.1. At the bottom are three buttons: 'Scale Now', 'Reset', and 'End'.

Figure 5c.2

Enter the value 1 for maximum and 0.1 for minimum for the Scale Into. Of course you can change this. It is advisable not to use the value 0 as the minimum as it represent nothing. Click on the *Scale Now* button. The raw data will be scaled.

**nn\_Solve** will also automatically store the minimum (in cell Q506) and the maximum (Q505) value of the raw data in the last row and first column of the raw data. (see Figure 5c.3 below)

	Q	R
1	Input 1	Input
2	CRIM	ZN
3		
4		
492	0.101367	
493	0.101209	
494	0.100352	
495	0.100068	
496	0.105905	
497	0.100125	
498	0.170975	
499	0.100659	
500	0.106122	
501	0.100735	
502	0.193328	
503	0.174832	
504		
505	88.9762	
506	0.00632	
507		

Figure 5c.3

We also need to scale Input 2 in column R, Input 3 in column S, Input 6 in column V,

Input 7 in column W, Input 8 in column X, Input 9 in column Y, Input 10 in column Z, Input 11 in column AA, Input 12 in column AB, Input 13 in column AC and Desire Output in Column AE. I've scale all the input data for your convenience. See Figure 5c.4 below

We don't need to scale Input 4 in column T and Input 5 in column U as those values are already between 0 and 1.

	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
1	Input 1	Input 2	Input 3	Input 4	Input 5	Input 6	Input 7	Input 8	Input 9	Input 10	Input 11	Input 12	Input 13		Desire
2	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT		PRICE
3															
4															
476	0.1661	0	0.6466	1	0.631	0.662	0.9743	0.0066	1	0.9141	0.8085	0.9878	0.0339		1
477	0.1949	0	0.6466	0	0.74	0.3959	0.9372	0.0625	1	0.9141	0.8085	1	0.5836		0.1733
478	0.1063	0	0.2815	0	0.538	0.4355	0.552	0.3064	0.1304	0.229	0.8936	0.9968	0.186		0.3311
479	0.1575	0	0.6466	0	0.583	0.4892	0.792	0.2197	1	0.9141	0.8085	0.9894	0.3656		0.3133
480	0.1004	0.525	0.1782	0	0.405	0.5756	0.206	0.5627	0.2174	0.2023	0.4255	0.9365	0.2147		0.44
481	0.2605	0	0.6466	0	0.671	0.5718	0.9907	0.0354	1	0.9141	0.8085	1	0.5339		0.1311
482	0.1007	0	0.2016	0	0.499	0.4386	0.3975	0.255	0.1739	0.1756	0.7021	1	0.1943		0.3556
483	0.1099	0	0.2815	0	0.538	0.4315	1	0.2697	0.1304	0.229	0.8936	0.994	0.5008		0.2111
484	0.1203	0	0.7009	0	0.605	0.8369	0.9609	0.0833	0.1739	0.4122	0.2234	0.9304	0.0544		1
485	0.1023	0	0.2364	0	0.448	0.4731	0.8507	0.4146	0.087	0.0878	0.5638	0.9895	0.471		0.2578
486	0.2458	0	0.6466	0	0.74	0.5557	0.931	0.0794	1	0.9141	0.8085	0.0685	0.4503		0.1022
487	0.1136	0	0.2815	0	0.538	0.4811	1	0.2769	0.1304	0.229	0.8936	0.9491	0.3121		0.2111
488	0.1373	0	0.6466	0	0.718	0.2686	0.9114	0.0566	1	0.9141	0.8085	0.7961	0.3386		0.3756
489	0.2807	0	0.6466	0	0.671	0.5101	1	0.0233	1	0.9141	0.8085	0.992	0.5533		0.1156
490	0.1029	0	0.3713	0	0.489	0.3547	0.0711	0.2235	0.1304	0.1718	0.6383	0.879	0.7677		0.4156
491	0.1226	0	0.7009	0	0.605	0.4394	0.9156	0.1175	0.1739	0.4122	0.2234	0.9955	0.2735		0.3933
492	0.1014	0	0.2364	0	0.448	0.4997	0.0381	0.4175	0.087	0.0878	0.5638	0.9659	0.1126		0.4511
493	0.1012	0.45	0.1092	0	0.437	0.5739	0.2698	0.3126	0.1739	0.4027	0.2766	0.9645	0.0781		0.5511
494	0.1004	0.25	0.1613	0	0.426	0.6066	0.3151	0.3884	0.1304	0.1794	0.6809	1	0.0982		0.5111
495	0.1001	0.35	0.0389	0	0.442	0.7051	0.4779	0.5373	0	0.1851	0.3085	0.9946	0.1038		0.6156
496	0.1059	0	0.7856	0	0.624	0.5386	0.9784	0.1089	0.1304	0.4771	0.9149	0.9719	0.2591		0.4
497	0.1001	0.85	0.1353	0	0.429	0.5662	0.2554	0.6734	0.1304	0.313	0.5638	0.9887	0.1278		0.4022
498	0.171	0	0.6466	0	0.718	0.4685	0.9516	0.0677	1	0.9141	0.8085	0.806	0.3855		0.2044
499	0.1007	0	0.1477	0	0.449	0.4905	0.5551	0.2381	0.087	0.1145	0.6277	0.9956	0.1852		0.3822
500	0.1061	0.2	0.1287	0	0.647	0.9854	0.8651	0.0611	0.1739	0.1469	0.0426	0.9818	0.0935		1
501	0.1007	0	0.4534	0	0.437	0.5196	0.0319	0.2839	0.1739	0.4027	0.6489	0.995	0.1393		0.4244
502	0.1933	0	0.6466	0	0.631	0.5087	1	0.0036	1	0.9141	0.8085	0.9225	0.2152		1
503	0.1748	0	0.6466	0	0.597	0.3939	0.9784	0.0296	1	0.9141	0.8085	0.7926	0.6807		0.2711
504															
505	88.976	100	27.74			8.78	100	12.127	24	711	22	396.9	37.97		50
506	0.0063	0	0.46			3.561	2.9	1.1296	1	187	12.6	0.32	1.73		5
507															

Figure 5c.4

## b) the neural network architecture

A neural network is a group of neurons connected together. Connecting neurons to form a NN can be done in various ways. This worksheet; column Q to column AR actually contain the NN architecture shown below:

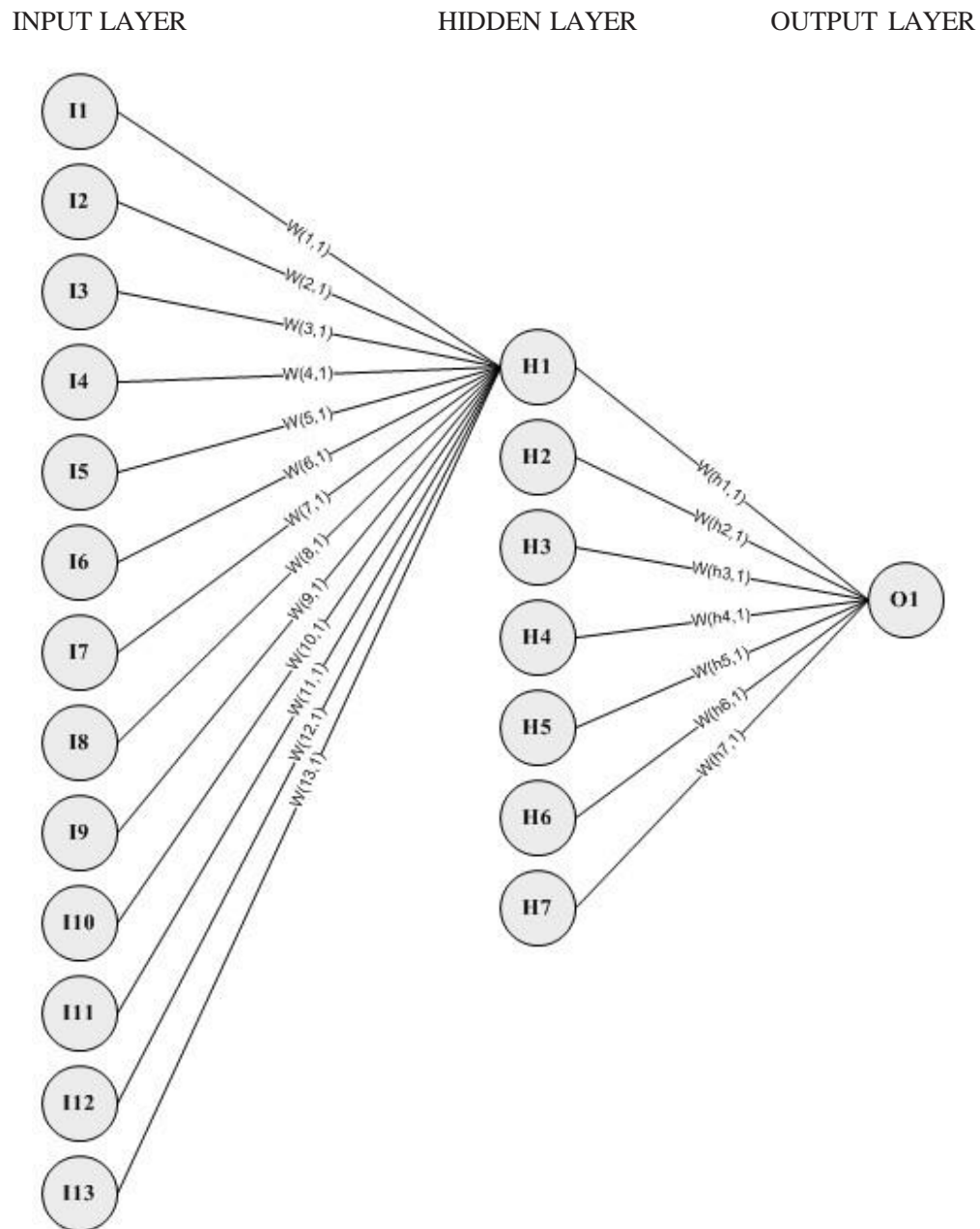


Figure 5c.5

There are 13 (column Q:AE) nodes or neuron on the input layer. 7 neurons on the hidden layer (column AJ:AP) and 1 neuron on the output layer (column AR).

Those lines that connect the nodes are call weights. I only connect part of them. In reality all the weights are connected layer by layer, like Figure 5.1

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. Here we have 499 input patterns map to 499 desired or target outputs. We reserve 1 input pattern for testing later.

Like what you see from the Figure 5c.5 above, this NN model consists of three layers:

Input layer with 13 neurons.

Column Q = Input 1 (I1); Column R = Input 2 (I2); Column S = Input 3 (I3);  
 Column T = Input 4 (I4) ; Column U = Input 5 (I5) ; Column V = Input 6 (I6);  
 Column W = Input 7 (I7); Column X = Input 8 (I8); Column Y = Input 9 (I9);  
 Column Z = Input 10 (I10); Column AA = Input 11 (I11); Column AB = Input 12 (I12);  
 Column AC = Input 13 (I13)

Hidden layer with 7 neurons.

Column AJ = Hidden Node 1 (H1); Column AK = Hidden Node 2 (H2);  
 Column AL = Hidden Node 3 (H3); Column AM = Hidden Node 4 (H4)  
 Column AN = Hidden Node 5 (H5); Column AO = Hidden Node 6 (H6)  
 Column AP = Hidden Node 7 (H7)

Output layer with 1 neuron.

Column AR = Output Node 1 (O1)

Now let's talk about the weights that connection all the neurons together

Note that:

- The output of a neuron in a layer goes to all neurons in the following layer. See Figure 5c.5
- We have 13 inputs node and 7 hidden nodes and 1 output node. Here the number of weights are  $(13 \times 7) + (7 \times 1) = 98$
- Each neuron has its own input weights.
- The output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input.

I have put the weights vector in one column. So the weights are contain in cells:

From Input Layer to Hidden Layer

$w(1,1) = \$AG\$5$  -> connecting I1 to H1

$w(2,1) = \$AG\$6$  -> connecting I2 to H1

$w(3,1) = \$AG\$7$  -> connecting I3 to H1

$w(4,1) = \$AG\$8$  -> connecting I4 to H1

$w(5,1) = \$AG\$9$  -> connecting I5 to H1

$w(6,1) = \$AG\$10 \rightarrow$  connecting I6 to H1  
 $w(7,1) = \$AG\$11 \rightarrow$  connecting I7 to H1  
 $w(8,1) = \$AG\$12 \rightarrow$  connecting I8 to H1  
 $w(9,1) = \$AG\$13 \rightarrow$  connecting I9 to H1  
 $w(10,1) = \$AG\$14 \rightarrow$  connecting I10 to H1  
 $w(11,1) = \$AG\$15 \rightarrow$  connecting I11 to H1  
 $w(12,1) = \$AG\$16 \rightarrow$  connecting I12 to H1  
 $w(13,1) = \$AG\$17 \rightarrow$  connecting I13 to H1

$w(1,2) = \$AG\$18 \rightarrow$  connecting I1 to H2  
 $w(2,2) = \$AG\$19 \rightarrow$  connecting I2 to H2  
 $w(3,2) = \$AG\$20 \rightarrow$  connecting I3 to H2  
 $w(4,2) = \$AG\$21 \rightarrow$  connecting I4 to H2  
 $w(5,2) = \$AG\$22 \rightarrow$  connecting I5 to H2  
 $w(6,2) = \$AG\$23 \rightarrow$  connecting I6 to H2  
 $w(7,2) = \$AG\$24 \rightarrow$  connecting I7 to H2  
 $w(8,2) = \$AG\$25 \rightarrow$  connecting I8 to H2  
 $w(9,2) = \$AG\$26 \rightarrow$  connecting I9 to H2  
 $w(10,2) = \$AG\$27 \rightarrow$  connecting I10 to H2  
 $w(11,2) = \$AG\$28 \rightarrow$  connecting I11 to H2  
 $w(12,2) = \$AG\$29 \rightarrow$  connecting I12 to H2  
 $w(13,2) = \$AG\$30 \rightarrow$  connecting I13 to H2

“

“ and

“ so

“ on

$w(1,7) = \$AG\$83 \rightarrow$  connecting I1 to H7  
 $w(2,7) = \$AG\$84 \rightarrow$  connecting I2 to H7  
 $w(3,7) = \$AG\$85 \rightarrow$  connecting I3 to H7  
 $w(4,7) = \$AG\$86 \rightarrow$  connecting I4 to H7  
 $w(5,7) = \$AG\$87 \rightarrow$  connecting I5 to H7  
 $w(6,7) = \$AG\$88 \rightarrow$  connecting I6 to H7  
 $w(7,7) = \$AG\$89 \rightarrow$  connecting I7 to H7

$w(8,7) = \$AG\$90$  -> connecting I8 to H7  
 $w(9,7) = \$AG\$91$  -> connecting I9 to H7  
 $w(10,7) = \$AG\$92$  -> connecting I10 to H7  
 $w(11,7) = \$AG\$93$  -> connecting I11 to H7  
 $w(12,7) = \$AG\$94$  -> connecting I12 to H7  
 $w(13,7) = \$AG\$95$  -> connecting I13 to H7

#### From Hidden Layer to Output Layer

$w(h1,1) = \$AG\$96$  -> connecting H1 to O1  
 $w(h2, 1) = \$AG\$97$  -> connecting H2 to O1  
 $w(h3, 1) = \$AG\$98$  -> connecting H3 to O1  
 $w(h4, 1) = \$AG\$99$  -> connecting H4 to O1  
 $w(h5, 1) = \$AG\$100$  -> connecting H5 to O1  
 $w(h6, 1) = \$AG\$101$  -> connecting H6 to O1  
 $w(h7, 1) = \$AG\$102$  -> connecting H7 to O1

After mapping the NN architecture to the worksheet and entering the input and desired output data., it is time to see what is happening inside those nodes.

### c) simple mathematic operations inside the neural network model

The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, and could change per network configuration and type of data. In general the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance. You could start a network configuration using a single hidden layer, and add more hidden layers if you notice that the network is not learning as well as you like.

For this Real Estate model, 1 hidden layer is sufficient. Select the cell AJ5 (H1), you can see

X  
 X  
 X  
 X  
 X  
 X  
 X



X

X

X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X

X

X

X

X

X

X

X

X

X

so that the network produces a better approximation of the desired output.

In NN technical term, we call this step of changing the weights as **TRAINING THE NEURAL NETWORK**.

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (EW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the EW.

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection. Phew, what craps is this back propagation!!!. Fortunately, you don't need to understand this, if you use MS Excel Solver to build and train a neural network model.

#### **d) Training NN as an Optimization Task Using Excel Solver**

Training a neural network is, in most cases, an exercise in numerical optimization of a usually nonlinear function. Methods of nonlinear optimization have been studied for hundreds of years, and there is a huge literature on the subject in fields such as numerical analysis, operations research, and statistical computing, e.g., Bertsekas 1995, Gill, Murray, and Wright 1981. There is no single best method for nonlinear optimization. You need to choose a method based on the characteristics of the problem to be solved.

MS Excel's Solver is a numerical optimization add-in (an additional file that extends the capabilities of Excel). It can be fast, easy, and accurate.

For a medium size neural network model with moderate number of weights, various quasi-Newton algorithms are efficient. For a large number of weights, various conjugate-gradient algorithms are efficient. These two optimization method are available with Excel Solver

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections

appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence. Values between -1 to 1 will be the best starting weights.

Let' fill out the weight vector. The weights are contain in AG5:AG102. From the **nn\_Solve** menu, select *Randomize Weights* (see Figure 5c.9)

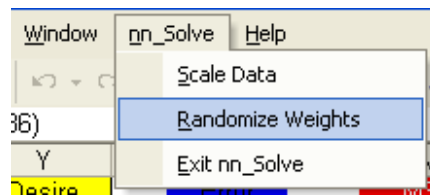


Figure 5c.9

Enter AG5:AG102 and click on the *Randomize Weights* button. AG5:AG102 will be filled out with values between -1 to 1. (see Figure 5c.10 below)

The learning algorithm improves the performance of the network by gradually changing each weight in the proper direction. This is called an **iterative** procedure. Each iteration makes the weights slightly more efficient at separating the target from the nontarget examples. The iteration loop is usually carried out until no further improvement is being made. In typical neural networks, this may be anywhere from ten to ten-thousand iterations. Fortunately, we have Excel Solver. This tool has simplified neural network training so much.

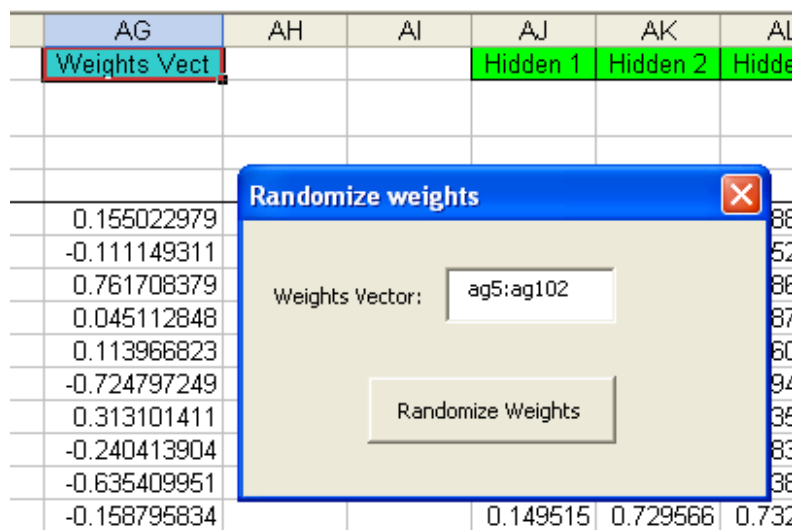


Figure 5c.10

### *Accessing Excel's Solver*

To invoke Solver see page 134. After executing Tools: Solver . . . , you will be presented with the Solver Parameters dialog box below:

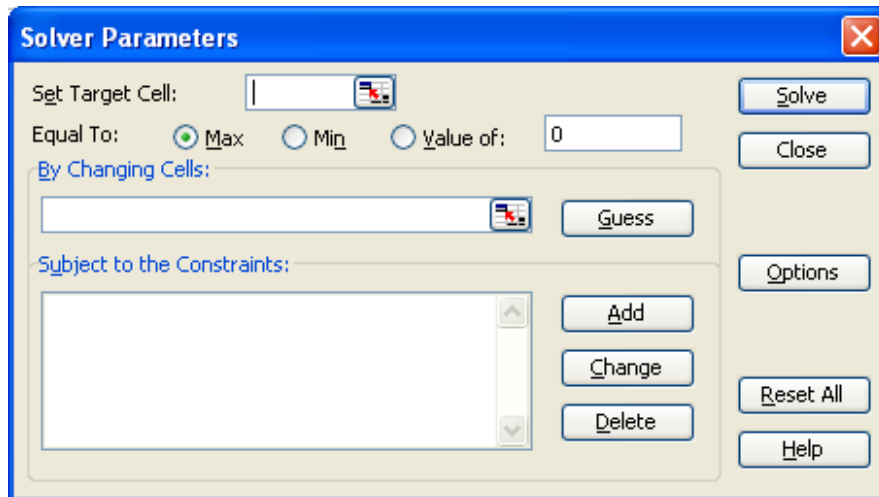


Figure 5c.14

Let us review each part of this dialog box, one at a time.

**Set Target Cell** is where you indicate the objective function (or goal) to be optimized. This cell must contain a formula that depends on one or more other cells (including at least one “changing cell”). You can either type in the cell address or click on the desired cell. Here we enter cell AV1.

In our NN model, the objective function is to minimize the Mean Squared Error. See Figure 5c.15 below

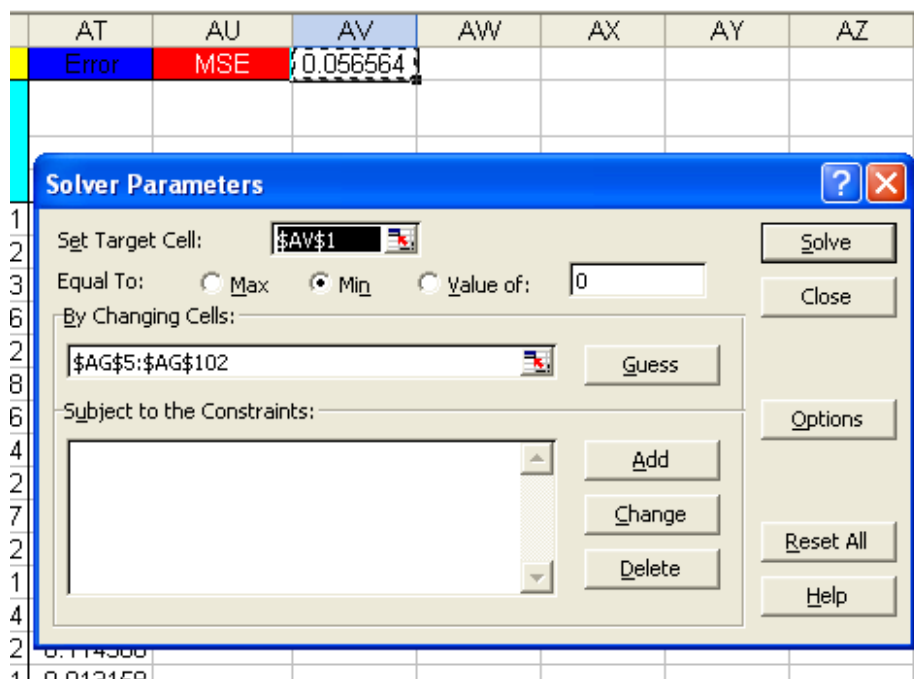


Figure 5c.15

**Equal to:** gives you the option of treating the Target Cell in three alternative ways. **Max** (the default) tells Excel to maximize the Target Cell and **Min**, to minimize it, whereas **Value** is used if you want to reach a certain particular value of the Target Cell by choosing a particular value of the endogenous variable.

Here, we select **Min** as we want to minimize MSE

**By Changing Cells** permits you to indicate which cells are the adjustable cells (i.e., endogenous variables). As in the Set Target Cell box, you may either type in a cell address or click on a cell in the spreadsheet. Excel handles multivariable optimization problems by allowing you to include additional cells in the By Changing Cells box. Each noncontiguous choice variable is separated by a comma. If you use the mouse technique (clicking on the cells), the comma separation is automatic.

Here, the cells that need to be changed is the weights vector. In our model, the weights are contained in Range AG5:AG102. So we enter, AG5:AG102. (See Figure 5c.15 above)

**Subject to the Constraints** is used to impose constraints on the endogenous variables. We will rely on this important part of Solver when we do Constrained Optimization problems. Although, training a neural network is not a Constrained Optimization problem, but it will be efficient if we limit the maximum and minimum value of the weights to a range of -100 to 100. Therefore, we put in

AG5:AG102 > -100 and  
AG5:AG102 < 100

Click on the Add button to add these constraints. (see Figure 5c.16)

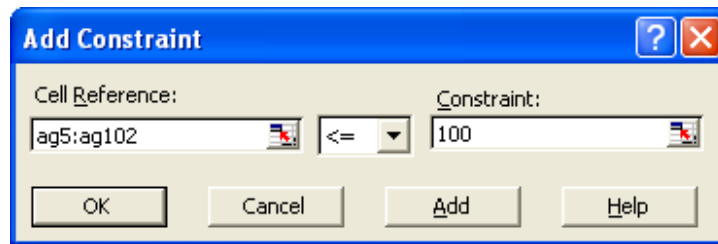


Figure 5c.16

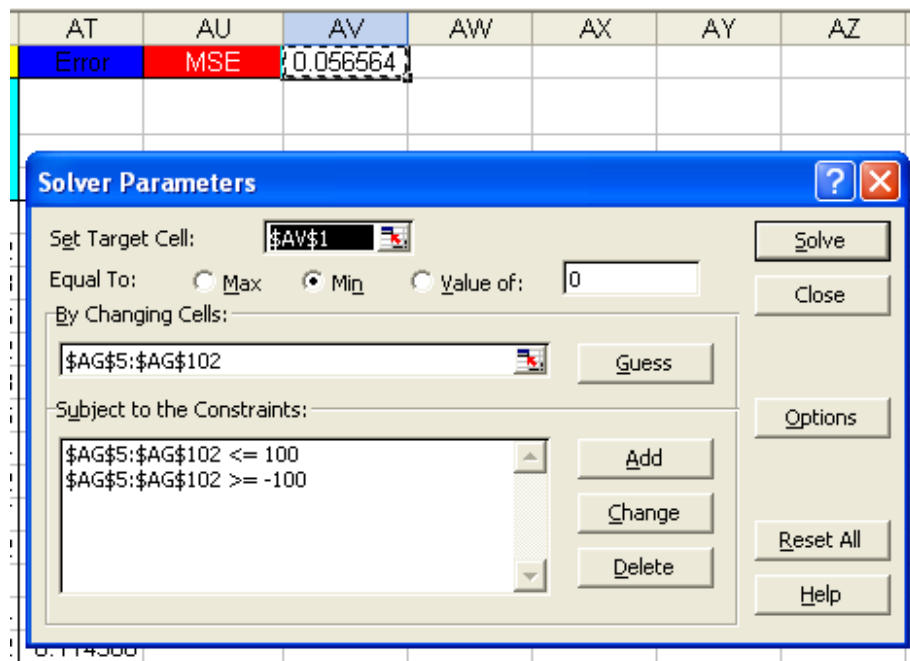


Figure 5c.17

**After that select the Options.** This will allow you to adjust the way in which Solver approaches the solution.. (see Figure 5c.18 below)

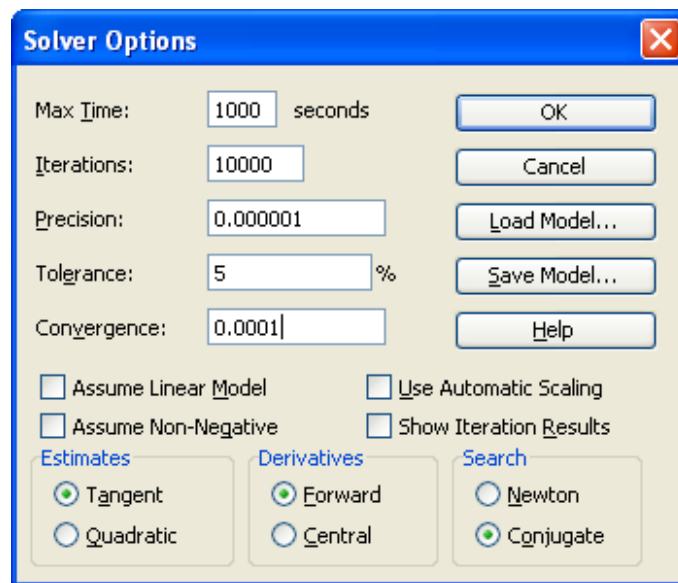


Figure 5c.18

As you can see, a series of choices are included in the Solver Options dialog box that direct Solver's search for the optimum solution and for how long it will search. These options may be changed if Solver is having difficulty finding the optimal solution. Lowering the Precision, Tolerance, and Convergence values slows down the algorithm but may enable Solver to find a solution.

For a neural network model, you can set :

- i. Max Time: 1000 seconds
- ii. Iterations: 10000
- iii. Precision: 0.000001
- iv. Tolerance: 5%
- v. Convergence: 0.0001

Select **Conjugate** as the *Search* method. This prove to be very effective in minimizing the Mean Squared Error. (see Figure 5c.18 above)

The Load and Save Model buttons enable you to recall and keep a complicated set of constraints or choices so that you do not have to reenter them every time.

Clicking OK to return to the Solver Parameters dialog box.

X  
X  
X  
X



X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

A message appears when Solver converged (see Figure 5c.21). In this case, Excel reports that “Solver has converged to the current solution. All constraints are satisfied.” This is good news!

Sometime, the Mean Square Error is not satisfactory and Solver unable to find the solution at one go. If this is the case then you, Keep the Solver solution and run Solver again. Follow the step discussed above. From experience, usually you will need to run Solver a few times before Solver arrive at a satisfactory Mean Square Error. (Note: value less than 0.01 will be satisfactory)

Bad news is a message like, “Solver could not find a solution.” If this happens, you must diagnose, debug, and otherwise think about what went wrong and how it could be fixed. The two quickest fixes are to try different initial weights values and to add bigger or smaller constraints to the weights.

Or you may change the network architecture by adding more hidden nodes.

From the Solver Results dialog box, you elect whether to have Excel write the solution it has found into the Changing Cells (i.e., Keep Solver Solution) or whether to leave the spreadsheet alone and NOT write the value of the solution into the Changing Cells (i.e., Restore Original Values). When Excel reports a successful run, you would usually want it to Keep the Solver Solution.

On the right-hand side of the Solver Results dialog box, Excel presents a series of reports. The Answer, Sensitivity, and Limits reports are additional sheets inserted into the current workbook. They contain diagnostic and other information and should be selected if Solver is having trouble finding a solution.

*It is important to understand that a saved Excel workbook will remember the information included in the last Solver run.*

**Save Scenario...** enables the user to save particular solutions for given configurations.

#### **e) using the trained model for forecasting**

After all the training and the MSE is below 0.01, its now time for us to predict. Goto the row 502 of the *RealEstate* spreadsheet. Remember, we have save 1 row of data for testing i.e. row 503

	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS
1	Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5	Hidden 6	Hidden 7		Output	Desire
2										PRICE
3										
4										
497	0.10427	0.00017	0.99264	0.56601	0.89449	0.98495	0.48396		0.46208	0.40222
498	0.63861	0.02309	0.99992	0.3774	0.90516	0.99947	0.72738		0.2431	0.20444
499	0.0655	0.00105	0.99309	0.49724	0.51741	0.97263	0.1366		0.41871	0.38222
500	0.34052	0.00178	0.34618	0.91289	0.72056	0.97912	0.10887		0.99195	1
501	0.02245	0.0002	0.98926	0.34293	0.76934	0.9695	0.64705		0.46318	0.42444
502	0.59175	0.26987	0.99983	0.46389	0.92509	0.99971	0.59706		0.87981	1
503										
504										

Figure 5c.22

Select AJ502:AR502. (See Figure 5c.22 above)

After that, you fill down until row 503 (see Figure 5c.23 below)

	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS
1	Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5	Hidden 6	Hidden 7		Output	Desire
2										PRICE
3										
4										
497	0.01844	5.3E-06	0.99001	0.59541	0.86788	0.71516	0.73304		0.46294	0.40222
498	0.51242	0.01475	0.99952	0.56173	0.86782	0.95112	0.49285		0.22367	0.20444
499	0.01778	0.00076	0.98906	0.57834	0.388	0.91129	0.18886		0.39929	0.38222
500	0.28028	8.8E-06	0.37851	0.95838	0.873	0.88476	0.05819		0.99446	1
501	0.00194	0.00025	0.98475	0.33104	0.59044	0.71141	0.82783		0.35022	0.42444
502	0.56653	0.25363	0.99896	0.55826	0.88878	0.96896	0.4313		0.91852	1
503	0.89274	0.16391	0.9999	0.2527	0.62072	0.95818	0.6885		0.26748	
504										

Figure 5c.23

So, for row 503 we have 0.26748 for predicted Output 1 (AR503). (see Figure 5c.23)

We need to scale this number back to raw data before they have any meaning to us.

Select *Scale Data* for the **nn\_Solve** menu.

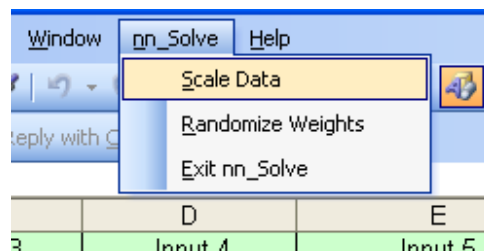


Figure 5c.24

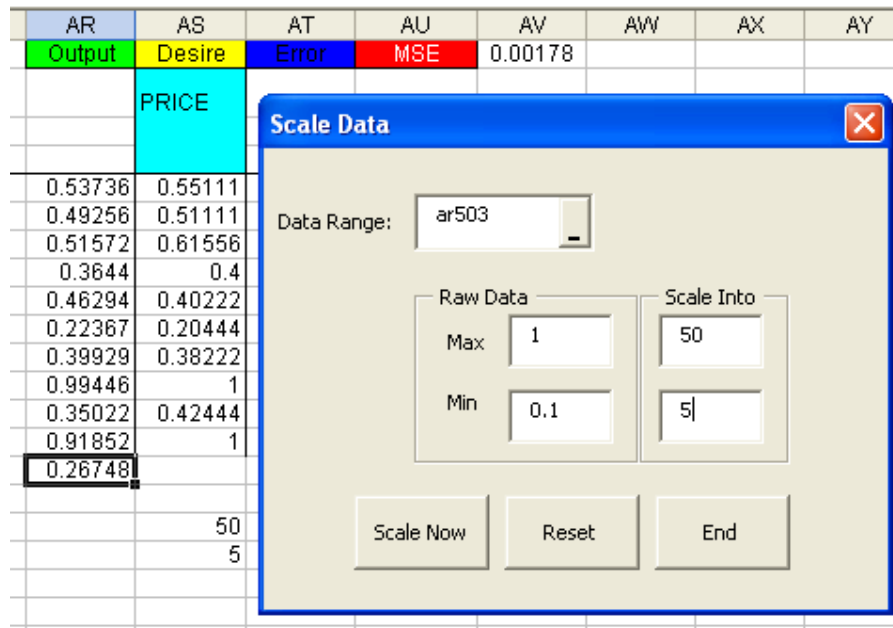


Figure 5c.25

Enter AR503 in the *Data Range*. As we are reversing what we did just now when we scale the raw data to the range 0 to 1. Now the raw data maximum become 1 and minimum become 0.1

As we have automatically saved the maximum (50) and minimum (5) of the raw data initially, now we use them as the maximum and minimum values to be scaled into. (See Figure 5c.25 above). Click on *Scale Now*.

	AR	AS
1	Output	Desire
2		PRICE
3		
4		
501	0.35022	0.42444
502	0.91852	1
503	17.0366	
504		

Figure 5c.26

So our predicted real estate price is 17.0366 as in AR503 (see Figure 5c.26 above). The actual price is 17.2 (cell O503). Of course when you do the training on your own, you will get slightly different result because of the MSE error that you have derived. The results here are based on the MSE of 0.00178

There you go. You have successfully use neural network to predict real estate price.

## 5) Classify Type Of Irises

Open the file *Irises.xls*. Using the petal and sepal sizes, we can use neural network to classify which class an Iris flower belongs to.

This is one of the standard benchmark that can be used to show how neural networks (and other techniques) can be used for classification. The neural network is train with 146 examples of three species of Iris. We reserve 1 example for testing our model later. So we have 147 records altogether. Two of the species are not linearly separable, so there is no simple rule for classification of flowers. After proper training the network is capable of classifying the flowers with a 100% accuracy.

### a) Selecting and transforming data

Open the *Workbook(Irises)* in folder Chapter 5 and bring up *worksheet (Raw Data)*. Here we, 147 inputs patterns and desire outputs. There are 4 input factors and 1 desire output (end result). Here we can see that, the data are still in alphabet form. NN can only be fed with numeric data for training. So we need to transform these raw data into numeric form.

This worksheet is self explanatory. For example, in column F (Desire), we have the Flower Type. NN cannot take or understand “setosa, versicol, virginic”. So we transform them to 1 for “setosa”, 0.5 for “versicol” and 0 for “virginic”.

We have to do this one by one manually.

If you select the *worksheet(Transform Data)*, it contains exactly what has been transform from *worksheet(Raw Data)*. (see Figure 5d.1)

	A	B	C	D	E	F
1	Input 1	Input 2	Input 3	Input 4		Desire
2	S_length	S_width	P_length	P_width		Flower Type
3						Setosa = 1
4						Versicol = 0.5
5						Virginic = 0
6						
7	5.1	3.5	1.4	0.2		1
8	4.9	3	1.4	0.2		1
9	4.7	3.2	1.3	0.2		1
10	4.6	3.1	1.5	0.2		1
11	5	3.6	1.4	0.2		1
12	5.4	3.9	1.7	0.4		1
13	4.6	3.4	1.4	0.3		1
14	5	3.4	1.5	0.2		1
15	4.4	2.9	1.4	0.2		1
16	4.9	3.1	1.5	0.1		1
17	5.4	3.7	1.5	0.2		1
18	4.8	3.4	1.6	0.2		1

Figure 5d.1

Now, we can see that Column A to column F in the worksheet (Transform Data) are all numerical. Apart from this transformation, we also need to “massage” the numeric data a little bit. This is because NN will learn better if there is uniformity in the data.

Thus we need to scale all the data into the value between 0 to 1. How do we do that?

Copy all data from Column A to F and paste them to Column H to M. The first column we need to scale is Column H (Input 1)

Select *Scale Data* on the **nn\_Solve** menu (see Figure 5d.2)

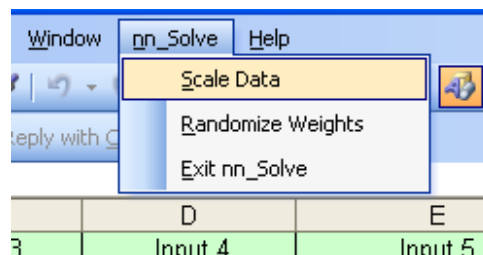


Figure 5d.2

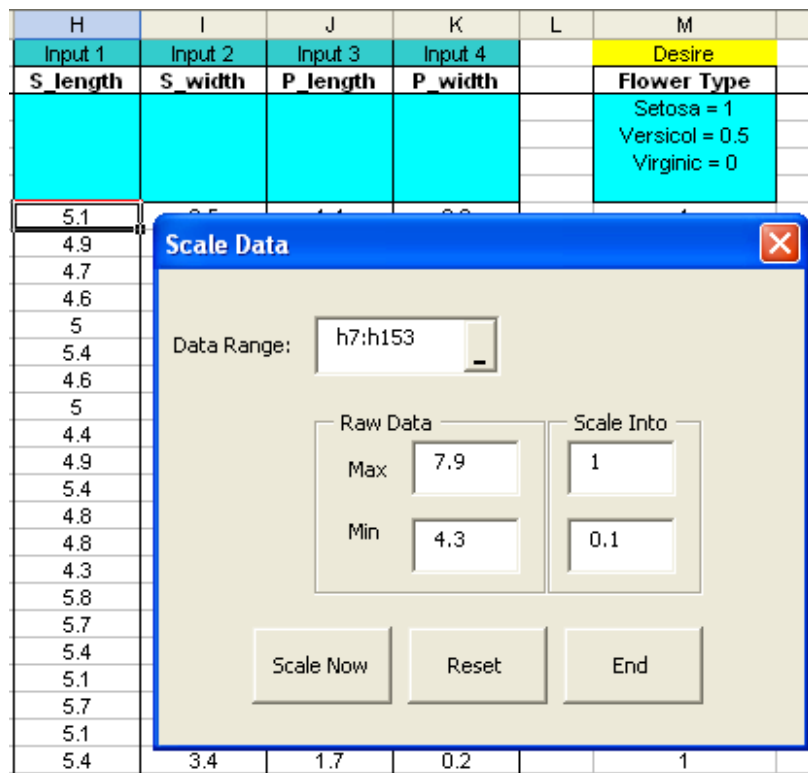


Figure 5d.3

Enter the reference for Input 1 (H7:H153) in the *Data Range*. Press the Tab key on your keyboard to exit. When you press Tab, **nn\_Solve** will automatically load the maximum (7.9) and the minimum (4.3) in the *Raw Data* frame *Min* and *Max* textbox. (see Figure 5d.2). Enter the value 1 for maximum and 0.1 for minimum for the *Scale Into*. Of course you can change this. It is advisable not to use the value 0 as the minimum as it represent nothing. Click on the *Scale Now* button. The raw data will be scaled.

**nn\_Solve** will also automatically store the minimum (in cell H156) and the maximum (H155) value of the raw data in the last row and first column of the raw data. (see Figure 5d.4 below). We may need these numbers later.

	G	H
1		Input 1
2		S_length
150		0.7
151		0.6
152		0.65
153		0.575
154		
155		7.9
156		4.3
157		

Figure 5d.4

We also need to scale Input 2 in column I, Input 3 in column J and Input 4 in column K. I've scale all the input data for your convenience. See Figure 5d.5 below.

We don't need to scale Desire column M as those values are already between 0 and 1.

	A	B	C	D	E	F	G	H	I	J	K
1	Input 1	Input 2	Input 3	Input 4		Desire		Input 1	Input 2	Input 3	Input 4
2	S_length	S_width	P_length	P_width		Flower Type		S_length	S_width	P_length	P_width
3						Setosa = 1					
4						Versicol = 0.5					
5						Virginic = 0					
6											
7	5.1	3.5	1.4	0.2		1		0.3	0.6625	0.16101695	0.1375
8	4.9	3	1.4	0.2		1		0.25	0.475	0.16101695	0.1375
9	4.7	3.2	1.3	0.2		1		0.2	0.55	0.14576271	0.1375
10	4.6	3.1	1.5	0.2		1		0.175	0.5125	0.17627119	0.1375
11	5	3.6	1.4	0.2		1		0.275	0.7	0.16101695	0.1375
12	5.4	3.9	1.7	0.4		1		0.375	0.8125	0.20677966	0.2125
13	4.6	3.4	1.4	0.3		1		0.175	0.625	0.16101695	0.175
14	5	3.4	1.5	0.2		1		0.275	0.625	0.17627119	0.1375
15	4.4	2.9	1.4	0.2		1		0.125	0.4375	0.16101695	0.1375
16	4.9	3.1	1.5	0.1		1		0.25	0.5125	0.17627119	0.1
17	5.4	3.7	1.5	0.2		1		0.375	0.7375	0.17627119	0.1375
18	4.8	3.4	1.6	0.2		1		0.225	0.625	0.19152542	0.1375
19	4.8	3	1.4	0.1		1		0.225	0.475	0.16101695	0.1
20	4.3	3	1.1	0.1		1		0.1	0.475	0.11525424	0.1
21	5.8	4	1.2	0.2		1		0.475	0.85	0.13050847	0.1375
22	5.7	4.4	1.5	0.4		1		0.45	1	0.17627119	0.2125
23	5.4	3.9	1.3	0.4		1		0.375	0.8125	0.14576271	0.2125
24	5.1	3.5	1.4	0.3		1		0.3	0.6625	0.16101695	0.175
25	5.7	3.8	1.7	0.3		1		0.45	0.775	0.20677966	0.175

Figure 5d.5

## b) the neural network architecture

A neural network is a group of neurons connected together. Connecting neurons to form a NN can be done in various ways. This worksheet; column H to column X actually contain the NN architecture shown below: It is a 4 layer neural network which is very effective in classification problems.

INPUT LAYER                      HIDDEN LAYER 1              HIDDEN LAYER 2              OUTPUT LAYER

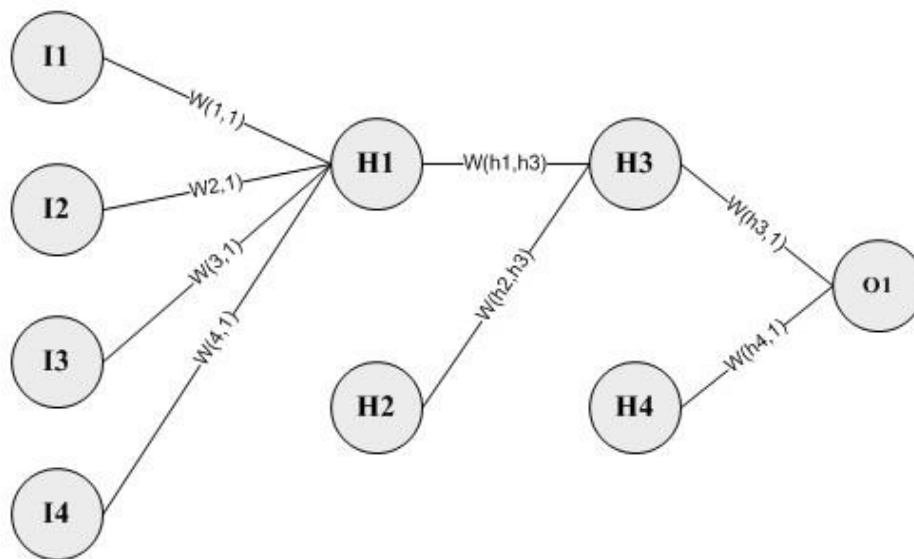


Figure 5d.6

There are 4 (column H:K) nodes or neuron on the input layer. 2 neurons on the hidden layer 1 (column R:S), 2 neurons on the hidden layer 2 (column U:V) and 1 neuron on the output layer (column X).

Those lines that connect the nodes are call weights. I only connect part of them. In reality all the weights are connected layer by layer, like Figure 5.1

The number of neurons in the input layer depends on the number of possible inputs we have, while the number of neurons in the output layer depends on the number of desired outputs. Here we have 146 input patterns map to 146 desired or target outputs. We reserve 1 input pattern for testing later.

Like what you see from the Figure 5d.6 above, this NN model consists of three layers:



Input layer with 4 neurons.

Column H = Input 1 (I1); Column I = Input 2 (I2); Column J = Input 3 (I3);  
Column K = Input 4 (I4)

Hidden layer 1 with 2 neurons.

Column R = Hidden Node 1 (H1)  
Column S = Hidden Node 2 (H2)

Hidden layer 2 with 2 neurons.

Column U = Hidden Node 1 (H3)  
Column V = Hidden Node 2 (H4)

Output layer with 1 neuron.

Column X = Output Node 1 (O1)

Now let's talk about the weights that connection all the neurons together

Note that:

- The output of a neuron in a layer goes to all neurons in the following layer. See Figure 5d.6
- We have 4 inputs node and 2 nodes for hidden layer 1, 2 nodes for hidden layer 2 and 1 output node. Here the number of weights are  $(4 \times 2) + (2 \times 2) + (2 \times 1) = 14$
- Each neuron has its own input weights.
- The output of the NN is reached by applying input values to the input layer, passing the output of each neuron to the following layer as input.

I have put the weights vector in one column O. So the weights are contain in cells:

From Input Layer to Hidden Layer 1

$w(1,1) = \$O\$7$  -> connecting I1 to H1

$w(2,1) = \$O\$8$  -> connecting I2 to H1

$w(3,1) = \$O\$9$  -> connecting I3 to H1

$w(4,1) = \$O\$10$  -> connecting I4 to H1

$w(1,2) = \$O\$11$  -> connecting I1 to H2

$w(2,2) = \$O\$12$  -> connecting I2 to H2

$w(3,2) = \$O\$13$  -> connecting I3 to H2

$w(4,2) = \$O\$14 \rightarrow$  connecting I4 to H2

“ and

“ so

“ on

“

“

From Hidden Layer 1 to Hidden Layer 2

$w(h1,h3) = \$O\$15 \rightarrow$  connecting H1 to H3

$w(h2,h3) = \$O\$16 \rightarrow$  connecting H2 to H3

$w(h1,h4) = \$O\$17 \rightarrow$  connecting H1 to H4

$w(h2,h4) = \$O\$18 \rightarrow$  connecting H2 to H4

From Hidden Layer 2 to Output Layer

$w(h3,1) = \$O\$19 \rightarrow$  connecting H3 to O1

$w(h4,1) = \$O\$20 \rightarrow$  connecting H4 to O1

After mapping the NN architecture to the worksheet and entering the input and desired output data., it is time to see what is happening inside those nodes.

### c) simple mathematic operations inside the neural network model

The number of hidden layers and how many neurons in each hidden layer cannot be well defined in advance, and could change per network configuration and type of data. In general the addition of a hidden layer could allow the network to learn more complex patterns, but at the same time decreases its performance. You could start a network configuration using a single hidden layer, and add more hidden layers if you notice that the network is not learning as well as you like.

For this Irises classification model, 2 hidden layers are needed. Select the cell R7 (H1), you can see

X

X

X

X

X

X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each EW by first computing the EA, the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the EW for each incoming connection of the unit. The EW is the product of the EA and the activity through the incoming connection.

Phew, what craps is this back propagation!!!. Fortunately, you don't need to understand this, if you use MS Excel Solver to build and train a neural network model.

#### **d) Training NN as an Optimization Task Using Excel Solver**

Training a neural network is, in most cases, an exercise in numerical optimization of a usually nonlinear function. Methods of nonlinear optimization have been studied for hundreds of years, and there is a huge literature on the subject in fields such as numerical analysis, operations research, and statistical computing, e.g., Bertsekas 1995, Gill, Murray, and Wright 1981. There is no single best method for nonlinear optimization. You need to choose a method based on the characteristics of the problem to be solved. MS Excel's Solver is a numerical optimization add-in (an additional file that extends the capabilities of Excel). It can be fast, easy, and accurate.

For a medium size neural network model with moderate number of weights, various quasi-Newton algorithms are efficient. For a large number of weights, various conjugate-gradient algorithms are efficient. These two optimization method are available with Excel Solver

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence. Values between -1 to 1 will be the best starting weights.

Let's fill out the weight vector. The weights are contain in O7:O20. From the **nn\_Solve** menu, select *Randomize Weights* (see Figure 5d.8)

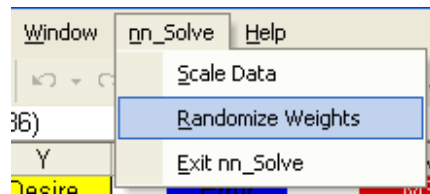


Figure 5d.8

Enter O7:O20 and click on the *Randomize Weights* button. O7:O20 will be filled out with values between -1 to 1. (see Figure 5d.9 below)

	N	O	P	Q	R	S	T
1		Weights Vect			Hidden 1	Hidden 2	
2							
3							
4							
5							
6							
7		-0.10905659	W(1,1)				
8		-0.25982857	W(2,1)				
9		0.530500054	W(3,1)				
10		-0.7295239	W(4,1)				
11		-0.65733778	W(1,2)				
12		-0.6252389	W(2,2)				
13		-0.84250724	W(3,2)		0.444301	0.35676	
14		0.297212839	W(4,2)		0.450369	0.336443	
15		-0.96069252	w(h1,h3)		0.464511	0.389242	
16		-0.54574108	w(h2,h3)		0.46509	0.35353	
17		0.910698533	w(h1,h4)		0.440455	0.306771	
18		0.562497854	w(h2,h4)		0.453724	0.340926	
19		-0.8312546	w(h3,1)		0.466179	0.365673	
20		-0.20993376	w(h4,1)		0.463531	0.394102	

Figure 5d.9

The learning algorithm improves the performance of the network by gradually changing each weight in the proper direction. This is called an **iterative** procedure. Each iteration makes the weights slightly more efficient at separating the target from the non target examples. The iteration loop is usually carried out until no further improvement is being made. In typical neural networks, this may be anywhere from ten to ten-thousand iterations. Fortunately, we have Excel Solver. This tool has simplified neural network training so much.

### *Accessing Excel's Solver*

To invoke Solver see page 134. After executing Tools: Solver . . . , you will be presented with the Solver Parameters dialog box below:

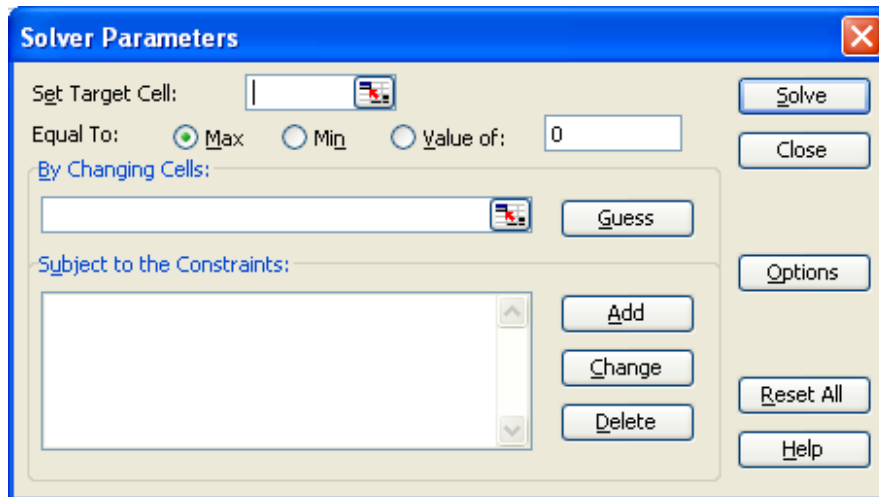


Figure 5d.13

Let us review each part of this dialog box, one at a time.

**Set Target Cell** is where you indicate the objective function (or goal) to be optimized. This cell must contain a formula that depends on one or more other cells (including at least one “changing cell”). You can either type in the cell address or click on the desired cell. Here we enter cell AB1.

In our NN model, the objective function is to minimize the Mean Squared Error. See Figure 5d.14 below

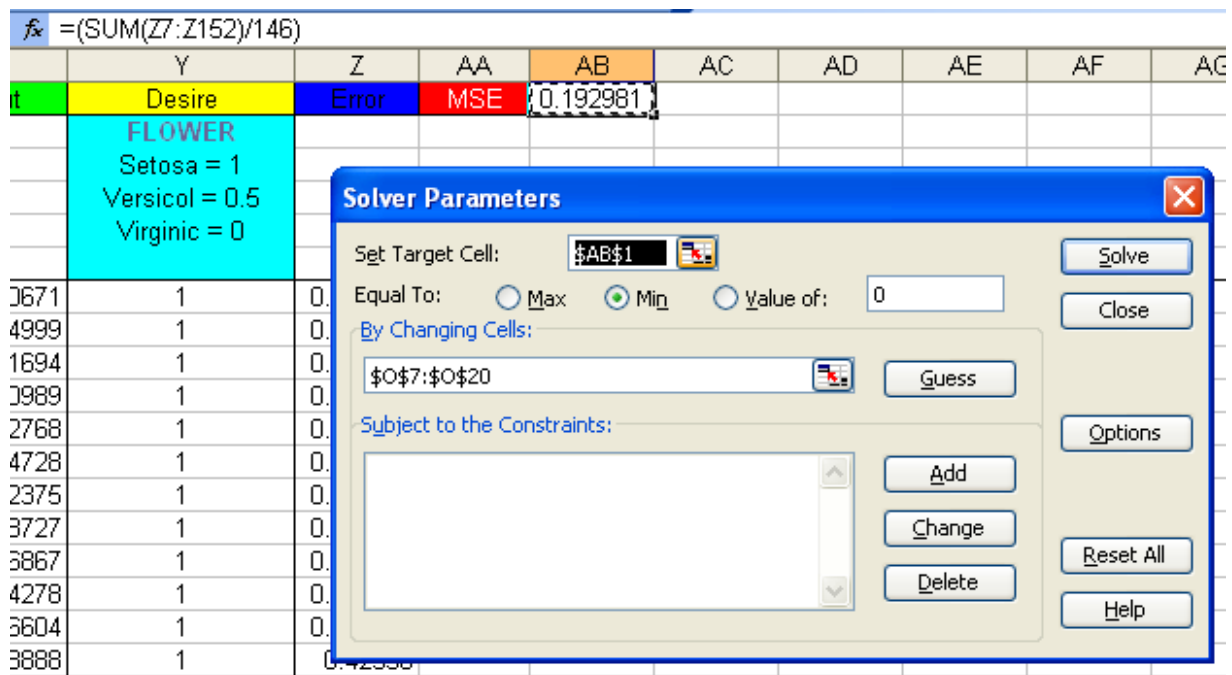


Figure 5d.14

**Equal to:** gives you the option of treating the Target Cell in three alternative ways. **Max** (the default) tells Excel to maximize the Target Cell and **Min**, to minimize it, whereas **Value** is used if you want to reach a certain particular value of the Target Cell by choosing a particular value of the endogenous variable.

Here, we select **Min** as we want to minimize MSE

**By Changing Cells** permits you to indicate which cells are the adjustable cells (i.e., endogenous variables). As in the Set Target Cell box, you may either type in a cell address or click on a cell in the spreadsheet. Excel handles multivariable optimization problems by allowing you to include additional cells in the By Changing Cells box. Each noncontiguous choice variable is separated by a comma. If you use the mouse technique (clicking on the cells), the comma separation is automatic.

Here, the cells that need to be changed is the weights vector. In our model, the weights are contain in Range O7:O20. So we enter, O7:O20. (See Figure 5d.14 above)

**Subject to the Constraints** is used to impose constraints on the endogenous variables. We will rely on this important part of Solver when we do Constrained Optimization problems. Although, training a neural network is not a Constrained Optimization problem, but it will be efficient if we limit the maximum and minimum value of the weights to a range of -100 to 100. Therefore, we put in

O7:O20 > -100 and  
O7:O20 < 100



X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

This part of the book is not available for viewing

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

A message appears when Solver converged (see Figure 5d.20). In this case, Excel reports that “Solver has converged to the current solution. All constraints are satisfied.” This is good news!

Sometime, the Mean Square Error is not satisfactory and Solver unable to find the solution at one go. If this is the case then you, Keep the Solver solution and run Solver again. Follow the step discussed above. From experience, usually you will need to run Solver a few times before Solver arrive at a satisfactory Mean Square Error. (Note: value less than 0.01 will be satisfactory)

Bad news is a message like, “Solver could not find a solution.” If this happens, you must diagnose, debug, and otherwise think about what went wrong and how it could be fixed. The two quickest fixes are to try different initial weights values and to add bigger or smaller constraints to the weights.

Or you may change the network architecture by adding more hidden nodes.

From the Solver Results dialog box, you elect whether to have Excel write the solution it has found into the Changing Cells (i.e., Keep Solver Solution) or whether to leave the spreadsheet alone and NOT write the value of the solution into the Changing Cells (i.e., Restore Original Values). When Excel reports a successful run, you would usually want it to Keep the Solver Solution.

On the right-hand side of the Solver Results dialog box, Excel presents a series of reports. The Answer, Sensitivity, and Limits reports are additional sheets inserted into the current workbook. They contain diagnostic and other information and should be selected if Solver is having trouble finding a solution.

### e) using the trained model for forecasting

After all the training and the MSE is below 0.01, its now time for us to predict. Goto the row 152 of the Irises spreadsheet. Remember, we have save 1 row of data for testing i.e. row 153

	Q	R	S	T	U	V	W	X	Y
1		Hidden 1	Hidden 2		Hidden 3	Hidden 4		Output	Desire
2									FLOWER
3									Setosa = 1
4									Versicol = 0.5
5									Virginic = 0
6									
151		0.73083	0.41937		0.08448276	0.68254		0.08448276	0
152		0.59344	0.32302		0.07374269	0.64872		0.07374269	0
153									
154									

Figure 5d.21

Select R152:X152. (See Figure 5d.21 above).After that, you fill down until row 153 (see Figure 5d.22 below)

	Q	R	S	T	U	V	W	X	Y
1		Hidden 1	Hidden 2		Hidden 3	Hidden 4		Output	Desire
2									<b>FLOWER</b> Setosa = 1 Versicol = 0.5 Virginic = 0
3									
4									
5									
6									
148		0.57719	0.28229		0.02804757	0.64146		0.02804757	0
149		0.57957	0.27677		0.02215018	0.64129		0.02215018	0
150		0.70598	0.38468		0.04758684	0.67479		0.04758684	0
151		0.73083	0.41937		0.08448276	0.68254		0.08448276	0
152		0.59344	0.32302		0.07374269	0.64872		0.07374269	0
153		0.39448	0.19557		0.08839486	0.5984		0.08839486	
154									

.Figure 5d.22

So, for row 153 we have 0.08839486 for predicted Output 1 (X153). (see Figure 5d.23 below). As the value 0.08839486 is very near to 0, we can take this result as 0.

	S	T	U	V	W	X	Y
1	Hidden 2		Hidden 3	Hidden 4		Output	Desire
2							<b>FLOWER</b> Setosa = 1 Versicol = 0.5 Virginic = 0
3							
4							
5							
6							
148	0.28229		0.02804757	0.64146		0.02804757	0
149	0.27677		0.02215018	0.64129		0.02215018	0
150	0.38468		0.04758684	0.67479		0.04758684	0
151	0.41937		0.08448276	0.68254		0.08448276	0
152	0.32302		0.07374269	0.64872		0.07374269	0
153	0.19557		0.08839486	0.5984		0.08839486	
154							

Figure 5d.23

So our predicted type of Irises price is Virginic which is represented by the value 0 as in X153 (see Figure 5d.23 above). The desire output is 0 (cell M153). We have successfully predicted the exact outcome. Of course when you do the training on your own, you will get slightly different result because of the MSE error that you have derived. The results here are based on the MSE of 0.00878

The general rules for classification are, when you have a predicted value of less than 0.4, this can be round up to the value of 0. And if you have a predicted value of more than 0.6, then you can round up to the value of 1. My suggestion is, try to rebuild and retrain the NN model if you get borderline cases like this.

There you go. You have successfully use neural network to predict the type of Irises.

## Conclusion

Neural networks' tolerance to noise makes them an excellent choice for solving real-world pattern recognition problems in which perfect data is not always available. As with any solution, there are costs. Training a network can be an arduous process. Depending on the domain, obtaining sufficient and suitable training data, sometimes called "truth", can be challenging. For example, speech transcription and natural language systems require large amounts of data to train. In addition, after the system is implemented it may not converge. That is, regardless of the amount of training, the weights may not always "settle" at particular values. In these cases, developing neural networks becomes more of an art than a science.

Each problem is unique, and so too are the solutions. Sometimes adding additional nodes or layers will stabilize a system. There are no hard rules, but there is one thing for certain; whether a neural network is computed by a computer, implemented in hardware, or propagated by hand, neural networks do not cogitate. They are simply powerful computational tools and their sophisticated mathematical computation positions them to be used to solve a broad range of problems.

Neural networks are increasingly being used in real-world business applications and, in some cases, such as fraud detection, they have already become the method of choice. Their use for risk assessment is also growing and they have been employed to visualize complex databases for marketing segmentation. This boom in applications covers a wide range of business interests — from finance management, through forecasting, to production. With this book, you have learned the neural network method to enable direct quantitative studies to be carried out without the need for rocket-science expertise.

## Chapter 6

### Markov Chain

#### Introduction

MARKOV CHAIN, a powerful term but simple to understand. *Markov chains* are used to analyze trends and predict the future (weather, stock market, genetics, *product success*, etc.)

*“A Markov Chain is a special kind of stochastic process where the outcome of an experiment depends only on the outcome of a previous experiment”*. In other words, the next state of the system depends only on the current state and not on the previous states. Stochastic processes are of interest for describing the behaviour of a system evolving over a period of time.

Suppose there is a physical or mathematical system that has  $n$  possible states and at any one time, the system is in one and only one of its  $n$  states. The state of the system at time  $t+1$  depends only on the state of the system at time  $t$ . Such a system is called **Markov Chain** or **Markov process**.

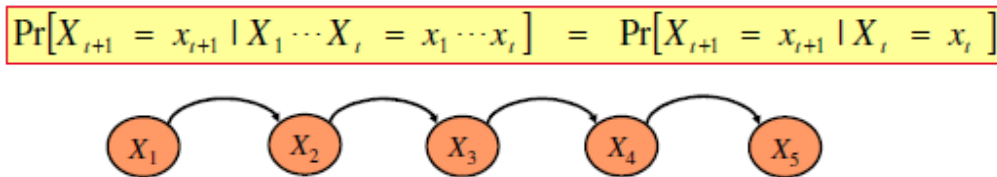


Figure 6.1

If a Markov process meets the following conditions:

1. The system can be described by a set of finite states and that the system can be in one and only one state at a given time.
- 2 The transition probability  $\Pr_{ij}$ , the probability of transition from state  $i$  to state  $j$ , is given from every possible combination of  $i$  and  $j$  (including  $i = j$ ) and the transition probabilities are assumed to be stationary (unchanging) over the time period of interest and independent of how state  $i$  was reached. and
3. Either the initial state of the system or the probability distribution of the initial state is known.

Then, is called a **finite-state first order Markov chain**.

To understand Markov Chains and do analytical calculations with Markov Chains, you must

have a basic understanding of probability and matrices. For the example in the activity on sunny and rainy days, you will write the probability of one day being rainy and the probability of one day being sunny. You should be able to take that probability and assess

how many sunny or rainy days there are in a year. Ability in matrix theory isn't necessary to complete this lesson, but is necessary if you wish to scale this lesson up, move beyond simulations, and do analytical calculations. Below is a refresher on how to do matrix multiplication.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} * \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} a*f + b*g \\ c*f + d*g \end{bmatrix}$$

An example of a Markov chain is a string with 2 nodes on it. These nodes are called node 0

and node 1. If we are at node 0 we can stay at node 0 with probability .1 and move to node 1

with probability .9. If we are at node 1 we can stay at node 1 with probability .7 or move to node 0 with probability .3. Here is how we would set up our probability matrix.

$$\begin{bmatrix} .1 & .3 \\ .9 & .7 \end{bmatrix}$$

We then select a set of 2 numbers between 0 and 1 that add to one and create a matrix with

them as follows. Let us select .5 and .5. These numbers represent a guess of the probability of being in node 0 and node 2.

$$\begin{bmatrix} .5 \\ .5 \end{bmatrix}$$

We then multiply those two matrices together to get a new 2 x 1 matrix.

$$\begin{bmatrix} .1 & .3 \\ .9 & .7 \end{bmatrix} * \begin{bmatrix} .5 \\ .5 \end{bmatrix} = \begin{bmatrix} .2 \\ .8 \end{bmatrix}$$

We multiply that onto the probability matrix again to get a new 2 x 1 matrix and we repeat until we continue to get the same matrix. For our system we ultimately get this.



$$\begin{bmatrix} .25 \\ .75 \end{bmatrix}$$

This represents that we spend 25% of our time on node 0 on the string, and 75% of our time on node 1.

Let's start to further understand Markov Chain with the following example.

### Example 1: The Courier Company

Suppose that a courier company has to deliver parcels from its customers in three locations in New York: Manhattan location (labelled A), Brooklyn end location (labelled B) and a Newark end location (labelled C). The company has a group of delivery drivers to serve all three locations. The company's scheduler has determined the following:

1. Of the calls to the Manhattan location, 30% are delivered in Manhattan area, 30% are delivered in the Brooklyn end, and 40% are delivered in the Newark end
2. Of the calls to the Brooklyn end location, 40% are delivered in Manhattan area, 40% are delivered in the Brooklyn end, and 20% are delivered in the Newark end
3. Of the calls to the Newark end location, 50% are delivered in Manhattan area, 30% are delivered in the Brooklyn end, and 20% are delivered in the Newark end.

After a delivery is made, a driver will go to the nearest location to make the next delivery. Therefore, the location of a specific driver is determined only by his or her previous location.

The delivery can be modelled with the following matrix:

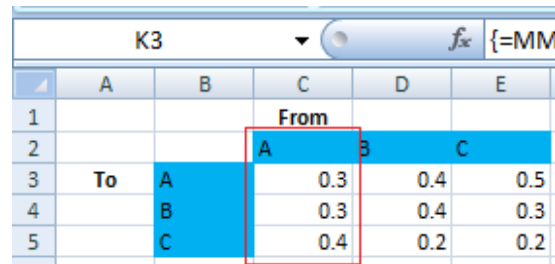
$$T = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0.3 & 0.4 & 0.5 \\ 0.3 & 0.4 & 0.3 \\ 0.4 & 0.2 & 0.2 \end{bmatrix} \end{matrix}$$

Figure 6.2

$$0.3 = 30\% ; 0.2 = 20\% ; 0.4 = 40\%$$

T is called the **transition matrix** of the above system. In our example, a **state** is the location of a particular driver in the system at a particular time. The entry  $s_{ji}$  in the above matrix represents the probability of transition from the state corresponding to i to the state corresponding to j. (e.g. the state corresponding to 2 is B)

To put this in a simple way, let's assume that it takes each delivery person the same amount of time (say 20 minutes) to make a delivery, and then to get to their next location. According to the statistician's data, after 20 minutes, of the drivers that began in A, 30% will again be in A, 30% will be in B, and 40% will be in C. (see Figure 6.3 below)



	A	B	C	D	E
1			From		
2			A	B	C
3	To	A	0.3	0.4	0.5
4		B	0.3	0.4	0.3
5		C	0.4	0.2	0.2

Figure 6.3

Since all drivers are in one of those three locations after their delivery, each column sums to 1; that is  $0.3 + 0.3 + 0.4 = 1$ . As this involves the theory of probabilities, each entry must be between 0 and 1, inclusive. Based on this important fact we can model this situation as a Markov chain where the next location for delivery depends only on the current location, not previous history. It is also true that our matrix of probabilities does not change during the time we are observing.

Now, let's elaborate this further with a simple question. If you start at location C, calculate the probability (say, P) that you will be in area B after 2 deliveries?

Accordingly we can get to B in two steps.

We can go from C to C, then from C to B, we can go from C to B, then from B to B,

or

we can go from C to A, then from A to B. To calculate P, let  $P(XY)$  represent the probability of going from X to Y in one delivery (where X,Y can be A,B or C).

Before we go further, let's refresh on how probabilities work. If two (or more) independent events must both (all) happen, to obtain the probability of them both (all) happening, we multiply their probabilities together. To get the probability of either (any) happening, we add the probabilities of those events together.

	A	B	C	D	E
1			From		
2			A	B	C
3	To	A	0.3	0.4	0.5
4		B	0.3	0.4	0.3
5		C	0.4	0.2	0.2

Figure 6.3a

This gives us

$P = P(CA)P(AB) + P(CB)P(BB) + P(CC)P(CB)$  for the probability that a delivery person goes from C to B in 2 deliveries.

$CA = 0.5, AB = 0.3, CB = 0.3, BB = 0.4, CC = 0.2, CB = 0.3$  (see Figure 6.3a above)

Substituting into our formula using the statistician's data above gives

$$P = (.5)(.3) + (.3)(.4) + (.2)(.3) = .33$$

Thus if we start at location C, we have a 33% chance of being in location B after 2 deliveries.

Let's look at another pair. If we start at location B, what is the probability of being at location B after 2 deliveries? Using the same steps shown above, the probability of going from location B to location B in two deliveries is

$$P(BA)P(AB) + P(BB)P(BB) + P(BC)P(CB) = (.4)(.3) + (.4)(.4) + (.2)(.3) = .34.$$

It's easy to calculate after 2 deliveries, but what if you need to know where you will be after 5, or 15 deliveries? It's going to take quite a long time. Fortunately there is an easier method. When you look carefully at these numbers you can see they are the result of matrix multiplication.. Going from C to B in 2 deliveries is the same as taking the inner product of row 2 and column 3 (row 2 multiply by row 3) Going from B to B in 2 deliveries is the same as taking the inner product of row 2 and column 2 (row 2 multiply by column 2). If you multiply T by T, the (2, 3) and (2,2) entries are respectively, the same answers that you got for these two questions above. The rest of  $T^2$  answers the same type of question for any other pair of locations X and Y.

$$T^2 = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0.41 & 0.38 & 0.37 \\ 0.33 & 0.34 & 0.33 \\ 0.26 & 0.28 & 0.3 \end{bmatrix} \end{matrix}$$

Figure 6.4

As you can see the elements on each column still add to 1 and each element is between 0 and 1, inclusive. This is a requirement since we are modelling our problem with a Markov chain. This matrix indicates the probabilities of going from location  $i$  to location  $j$  in exactly 2 deliveries.

With this matrix, it is much easier to find where we will be after 3 deliveries. We will let  $p(AB)$  represent the probability of going from A to B in 2 deliveries.

Calculate the probability of going from C to B in 3 deliveries: It is

$$p(CA)P(AB) + p(CB)P(BB) + p(CC)P(CB) = (.37)(.3) + (.33)(.4) + (.3)(.3) = .333$$

You will see that this probability is the inner product of row 2 of  $T^2$  and column 3 of  $T$ . Therefore, if we multiply  $T^2$  by  $T$ , we will get the probability matrix for 3 deliveries.

$$T^3 = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0.385 & 0.39 & 0.393 \\ 0.333 & 0.334 & 0.333 \\ 0.282 & 0.276 & 0.274 \end{bmatrix} \end{matrix}$$

With the examples shown above you should know how we find the matrix of probabilities for 4, 5 or more deliveries. Just keep on multiplying the  $T$  matrix. You will find out that the elements on each column still add to 1. It is vital that you keep as many decimal places as possible to retain accuracy. By any means do not round your answers.

$$T^4 = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0.3897 & 0.3886 & 0.3881 \\ 0.3333 & 0.3334 & 0.3333 \\ 0.2770 & 0.2780 & 0.2786 \end{bmatrix} \end{matrix}, \quad T^5 = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0.38873 & 0.38894 & 0.38905 \\ 0.33333 & 0.33334 & 0.33333 \\ 0.27794 & 0.27772 & 0.27762 \end{bmatrix} \end{matrix}, \quad T^6 = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0.388921 & 0.388878 & 0.388857 \\ 0.333333 & 0.333334 & 0.333333 \\ 0.277746 & 0.277788 & 0.277810 \end{bmatrix} \end{matrix},$$

$$T^7 = \begin{matrix} & \begin{matrix} A & B & C \end{matrix} \\ \begin{matrix} A \\ B \\ C \end{matrix} & \begin{bmatrix} 0.3888825 & 0.3888910 & 0.3888953 \\ 0.3333333 & 0.3333334 & 0.3333333 \\ 0.2777842 & 0.2777765 & 0.2777714 \end{bmatrix} \end{matrix}$$

As we calculate more and more deliveries you will see a pattern in these matrices. The numbers in each row seems to be converging to a particular number. It can be observed that the difference between the elements of the transition probabilities after a number of transitions tends to be zero. Think about what this tells us about our long-term probabilities. This tells us that after a large number of deliveries, it no longer matters which location we were in when we started. At the end of the week, we have (approximately) a 38.9% Chance of being at location A, a 33.3% chance of being at

location B, and a 27.8% chance of being in location C. This convergence will happen with most of the transition matrices that we consider.

In other words all elements of all rows tend towards a common limit as the number of transition increases. Thus the process is said to have reached a steady state

I know that it is quite cumbersome to calculate the T matrix. Luckily there is a built in formula in MS Excel that you can use for matrix multiplication. The formula is =MMULT(). Open the file Markov.xlsx in the folder Chapter 6. Select worksheet “Transport”

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
x

Please visit <http://www.xlpert.com/forecast.htm> for more information on the book

## Example 2: The Cola Wars

Assume there are three soft drink companies, King, Peps, and Royal, which have consistent customer preference from year to year represented by the following transition matrix T. (see Figure 6.7)

	A	B	C	D	E
			From		
			King	Peps	Royal
To	King		0.75	0.33	0.1
	Peps		0.1	0.55	0.02
	Royal		0.15	0.12	0.88

Figure 6.7

Given that a person's last cola purchase was King cola, there is a 75% chance that his next cola purchase will also be King cola, 10% will be Peps and 15% will be Royal. Given that a person's last cola purchase was Peps cola, there is a 55% chance that his next cola purchase will also be Peps cola, 33% will be King and 12% will be Royal. Given that a person's last cola purchase was **Royal cola**, there is a **88%** chance that his next cola purchase will also be **Royal cola**, 10% will be King and 2% will be Peps. Let's look at a few scenarios for further understanding of Markov Chain.

*Scenario 1:*

Given that a person is currently a King cola purchaser, what is the probability that he will purchase Peps cola two purchases from now?

Select the worksheet "Cola". The formulas are entered like the one explained above for worksheet "Transport".

Two purchase from now will be  $T \times T \times T$  ( $T^3$ )

The value is 0.13933 i.e. there is a 13.933% that the next two purchases from now will be a Peps cola (see Figure 6.8)

X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X  
X

P	Q	R	S	T
			T ^ 4	
		King	Peps	Royal
King		0.47487	0.45574	0.25485
Peps		0.13604	0.18136	0.06616
Royal		0.3891	0.36289	0.67898

Figure 6.9

### Scenario 3:

Assume each person makes one cola purchase per week. Suppose 50% of all people now drink King, 30% drink Peps and 20% drink Royal. What fraction of people will be drinking King cola three weeks from now?

So here we have the market shares of the three Colas which is also the initial states of 0.5, 0.3 and 0.2 respectively

Thus we will use the matrix  $T^4$  for our calculation. (3 weeks means next three purchases from now)

$$\Pr[X_3=\text{King}] = 0.5 \times 0.47487 + 0.3 \times 0.45574 + 0.2 \times 0.25485 = 0.42513$$

Thus the fraction of people will be drinking King cola three weeks from now is 42.513%

I hope the above examples gave you a good idea about the process of Markov chains. There is a rich mathematical theory of Markov chain (unless you have a mathematical bent, you will probably want to skip heavy stuff below.) The important objective of this book is to show readers on how to build a prediction model practically with MS Excel. However if you want to know the theoretical aspect of Markov Chain, please read on.

### Definitions

From a mathematical point of view, a Markov chain describes a process that can be considered to be in exactly one of a number of "states" at any given time. A Markov chain with  $n$  states, the **state vector** is a column vector whose  $i^{\text{th}}$  component represents the probability that the system is in the  $i^{\text{th}}$  state at that time.

The sum of the entries of a state vector must be 1. For example, vectors  $X_0$  and  $X_1$  in the above example are state vectors. If  $p_{ij}$  is the probability of movement (transition) from one **state j** to **state i**, then the matrix  $T=[p_{ij}]$  is called the **transition matrix of the Markov chain**.

The following Theorem gives the relation between two consecutive state vectors:

*If  $X_{n+1}$  and  $X_n$  are two consecutive state vectors of a Markov chain with transition matrix  $T$ , then*

$X_{n+1} = T \text{ multiply by } X_n$  Or

$$X_{n+1} = T X_n$$

We are usually interested in the long-term behavior of a general state vector  $X_n$  in a Markov Chain. That is we will find the limit of  $X_n$  as  $n \rightarrow \infty$  ( $n$  approaching infinity). For **regular Markov chains** this is always the case. But not so for cyclic chains or for chains with ergodic sets like the example below.

It may happen that this limit does not exist, for example let

$$T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

then

$$T^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, T^3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, T^4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, T^5 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

As you can see  $X_n$  cycles between the vectors  $(0, 1)$  and  $(1, 0)$  and therefore does not reach a fixed vector. Hence  $T$  is ergodic, but not regular.

Then what makes  $X_n$  approach a limiting vector as  $n \rightarrow \infty$ . This will be explained in next theorem but first we need a definition:

A transition matrix  $T$  is called **regular** if, for some integer  $r$ , all entries of  $T^r$  are strictly positive. (0 is not strictly positive).

For example, the matrix

$$T = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

is regular since

$$T^2 = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

A Markov chain process is called **regular** if its transition matrix is regular. The main theorem in Markov chain theory is explain below:



1. If  $T$  is a regular transition matrix, then as  $n$  approaches infinity,  $T^n \rightarrow S$  where  $S$  is a matrix of the form  $[v, v, \dots, v]$  with  $v$  being a constant vector.
2. If  $T$  is a regular transition matrix of a Markov chain process, and if  $X$  is any state vector, then as  $n$  approaches infinity,  $T^n X \rightarrow p$ , where  $p$  is a fixed probability vector (the sum of its entries is 1), all of whose entries are positive.

Consider a Markov chain with a regular transition matrix  $T$ , and let  $S$  denote the limit of  $T^n$  as  $n$  approaches infinity, then  $T^n X \rightarrow SX = p$ , and therefore the system approaches a fixed state vector  $p$  called the **steady-state vector of the system**.

Now since  $T^{n+1} = TT^n$  and that both  $T^{n+1}$  and  $T^n$  approach  $S$ , we have  $S = TS$ . Note that any column of this matrix equation gives  $TP = p$ . Therefore, the steady-state vector of a regular Markov chain with transition matrix  $T$  is the unique probability vector  $p$  satisfying  $TP = p$ .

Is there a way to compute the steady-state vector of a regular Markov chain without using the limit? Well, if we can solve  $TP = p$ , for  $p$ , then yes! You might have seen this sort of thing before (and certainly will in your first linear algebra course:  $Ax = b$ )

Let's me show you an example on how linear algebra can be use to solve  $p$ .

If you have lived in London for a while, you must have realized that the weather is a main concern of the population. An unofficial study of the weather in the city in early spring yields the following observations:

1. It is almost impossible to have two nice days in a row
2. If we have a nice day, we just as likely to have snow or rain the next day
3. If we have snow or rain, then we have an even chance to have the same the next day
4. If there is a change from snow or rain, only half of the time is this a change to a nice day.
  - a. Write the transition matrix to model this system.
  - b. If it is nice today, what is the probability of being nice after one week?
  - c. Find the long time behaviour of the weather.

**Answer:**

1) since the weather tomorrow depends only on today, this is a Markov chain process. The transition matrix of this system is

$$= \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.5 & 0.3 \\ 0.1 & 0.3 & 0.6 \end{bmatrix} \begin{bmatrix} N \\ R \\ S \end{bmatrix}$$

where the letters  $N$ ,  $R$ ,  $S$  represent Nice, Rain, Snow respectively.

2) If it is nice today, then the initial state-vector is

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

After seven days (one week), the state-vector would be

$$= \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.2 & 0.5 & 0.3 \\ 0.1 & 0.3 & 0.6 \end{bmatrix}^7 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.20 \\ 0.50 \\ 0.30 \end{bmatrix}$$

So, there is about 20% chance of being nice in one week.

- 3) Notice first that we are dealing with a regular Markov chain since the transition matrix is regular, so we are sure that the steady-state vector exists. To find it we solve the homogeneous system  $(T-I)X=0$  which has the following coefficient matrix:

$$= \begin{bmatrix} -0.3 & 0.2 & 0.1 \\ 0.2 & -0.5 & 0.3 \\ 0.1 & 0.3 & -0.4 \end{bmatrix}$$

Reducing to reduced echelon form gives

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The general solution of this system is

$$\begin{bmatrix} 0.5t \\ 0.4t \\ 0.1t \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

So what solution do we choose? Remember that a steady-state vector is in particular a probability vector; that is the sum of its components is 1:  $0.5t + 0.4t + 0.1t = 1$  gives  $t = 0.4$ . Thus, the steady-state vector is

$$\begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

In the long term, there is 20% chance of getting a nice day, 40% chance of having a rainy day and 40% chance of having a snowy day.

### Conclusion:

In our everyday lives, jobs and business, we make decisions that have consequences. By deciding to get up in the morning, we have created the possibility that events might occur in the day. Each of these events has associated probability. Which events occur during the day will influence what might occur next, and probability is still associated with it. Most naturally occurring events are not certain. There is always a series of possibilities and probabilities associated with them. By understanding these, we can understand how events unfold in our lives, jobs and business, how to optimize our choices, and how to travel the optimal path with the highest probability. Hopefully with the Markov Chain examples built with MS Excel above will help you on your day to day decisions making process.

# Chapter 7

## Bayes Theorem

### Introduction

#### Bayesian Probability

Bayesian probability's application in corporate and business world is highly dependent on the "degree of belief" rather than historical frequencies of identical or similar events.

You may think that you need to know a lot about probability theory in order to study Bayesian methods. In fact you don't have to. While any mathematically based topic can be taken to rather complex depths, the use of a basic Bayesian probability model in forecasting can help refine probability estimates using an intuitive process.

You can also use your historical beliefs based on frequency to use the model; it's a very versatile model. I'll show you with 2 examples build with MS Excel on how this is done.

Bayes' Theorem is a theorem of probability theory first proposed by the Reverend Thomas Bayes. It can be seen as a way of understanding how the probability that a theory is true is affected by a new piece of evidence.

It has been used in many fields, ranging from marine biology to the development of "Bayesian" spam blockers for email systems.

It has been used to try to clarify the relationship between theory and evidence in the philosophy of science. Many insights in the philosophy of science involving confirmation, falsification, the relation between science and pseudoscience, and other topics can be made more precise, and sometimes extended or corrected, by using Bayes' Theorem.

Before we learn more Bayes' Theorem, we need to understand the concept of 'mutually exclusive'

#### Mutually exclusive

In this section of the tutorial, you will learn the difference between statistically independent events and mutually exclusive events, sampling with and without replacement.

Two events are *statistically independent* if the first event do not affect outcome of the second event and vice versa. But, these two events may have something in commons called join events. If the two events do not have something in common, then the two events are called *mutually exclusive events*.

For example, you have a basket with 5 colourful marbles, 2 marbles has Red colour, 2 marbles has Green colour, and one Blue marble in short [R, R, G, G, B]. Let us do simple experiments called *sampling with replacement* and *sampling without replacement*.

Take two of these marbles without replacement, meaning that after you see the colour of the first marble you take, record it and immediately take the second marble. Do not return those marbles into the basket. Now, what is the probability that the first marble you take is Red and the second marble is Blue? Since you have 5 marbles in the beginning and 2 Red, then the probability that you get Red first marble is 2 out of 5 = 40%. After the first even, the total marbles in the basket become 4. The probability that you get Blue second marble is 1 out of 4 = 25%. Thus, the probability that you get Red marble then Blue marble is 40% times 25% = 10%. (i.e.  $0.4 \times 0.25 = 0.1$ )

Now return all marbles in the basket. This time you will take two of these marbles with replacement, meaning that after you see the colour of the first marble and record it, return that marble to the basket before you take the second marble. Notice that the total marbles of the two events are constant, equal to 5, because every time you take a marble, you return it to the basket before you do the next event. What is the probability that the first marble you take is Red and the second marble is Blue? You have 2 red marbles out of 5 = 40% and you have 1 Blue marble out 5 = 20%. Multiply these two percentages, we get 40% times 20% = 8%. (i.e.  $0.4 \times 0.2 = 0.08$ )

Now the statistical moral story is like this. The two types of sampling have something in common because they are taken from the same sample of 5 marbles in one basket. In *sampling without replacement*, the first event will affect the second event. We say that the two events are dependent to each other. In the marble experiment, this relation happens because the total sample change (i.e. reduce) after the first event. The probability of taking the next event is depending on the previous event. On the other hand, for *sampling with replacement*, the first event does not affect the outcome of the second event. We called the two events are *statistically independent*. In the marble experiment, the total sample never changes because you always return the marble that you examine. The probability of taking the next event is not depending on the previous event. (as oppose to Markov Chain where the probability of the next event depends only on the current event and not on the previous event. See Chapter 6)

Bayes' theorem can be stated as follows:

**Bayes' theorem.** Let  $A_1, A_2, \dots, A_n$  be a set of mutually exclusive events that together form the sample space  $S$ . Let  $B$  be any event from the same sample space, such that  $P(B) > 0$ . Then,

$$P(A_k | B) = \frac{P(A_k \cap B)}{P(A_1 \cap B) + P(A_2 \cap B) + \dots + P(A_n \cap B)}$$

Note: Invoking the fact that  $P(A_k \cap B) = P(A_k)P(B | A_k)$ , Baye's theorem can also be expressed as

$$P(A_k | B) = \frac{P(A_k) P(B | A_k)}{P(A_1) P(B | A_1) + P(A_2) P(B | A_2) + \dots + P(A_n) P(B | A_n)}$$

Figure 7.1

$P(A)$  is the probability of  $A$  occurring, and is called the prior probability.  $P(A|B)$  is the conditional probability of  $A$  given that  $B$  occurs. This is the posterior probability due to its variable dependency on  $B$ . This assumes that the  $A$  is not independent of  $B$ .

$P(B|A)$  is the conditional probability of  $B$  given that  $A$  occurs.

$P(B)$  is the probability of  $B$  occurring.

If we are interested in the probability of an event of which we have prior observations; we call this the prior probability. We'll deem this event event  $A$ , and its probability  $P(A)$ . If there is a second event that affects  $P(A)$ , which we'll call event  $B$ , then we want to know what the probability of  $A$  is given  $B$  has occurred. In probabilistic notation this is  $P(A|B)$ , and is known as posterior probability or revised probability. This is because it has occurred after original event, hence the post in posterior. This is how Bayes' theorem uniquely allows us to update our previous beliefs with new information.

Unless you are a world-class statistician, Bayes' theorem (as expressed above) can be scary. But in reality it is easy to use. The rest of this chapter covers material and examples that can help you understand when and how to apply Bayes' theorem effectively.

### When to Apply Bayes' Theorem

There are a few factors that we need to know when applying Bayes' theorem. You should consider Bayes' theorem when the following conditions exist.

- The sample space is partitioned into a set of mutually exclusive events  $\{ A_1, A_2, \dots, A_n \}$ .
- Within the sample space, there exists an event B, for which  $P(B) > 0$ .
- The analytical goal is to compute a conditional probability of the form:  $P( A_k | B )$ .
- You know at least one of the two sets of probabilities described below.
  - $P( A_k \cap B )$  for each  $A_k$
  - $P( A_k )$  and  $P( B | A_k )$  for each  $A_k$

Bayes' theorem can be best understood through an example. We will see 2 examples below on how to apply Bayes' theorem to solve statistical problems.

### Example 1: Weather Forecast (Rainy or Sunny)

Susan is celebrating her 30<sup>th</sup> birthday tomorrow. She has planned a huge barbeque party outdoor in the desert. In the past few years, it has rained only 5 days each year in the desert. Unfortunately, the weatherman has predicted rain for tomorrow. When it actually rains, the weatherman correctly forecasts rain 90% of the time. When it doesn't rain, he incorrectly forecasts rain 10% of the time. What is the probability that it will rain on the day of Susan's birthday celebration?

*Solution:* The sample space is defined by two mutually-exclusive events - it rains or it does not rain. Additionally, a third event occurs when the weatherman predicts rain. Notation for these events appears below.

- Event  $A_1$ . It rains on Susan's birthday celebration.
- Event  $A_2$ . It does not rain on Susan's birthday celebration
- Event B. The weatherman predicts rain.

In terms of probabilities, we know the following:

- $P( A_1 ) = 5/365 = 0.0136985$  [It rains 5 days out of the year.]
- $P( A_2 ) = 360/365 = 0.9863014$  [It does not rain 360 days out of the year.]
- $P( B | A_1 ) = 0.9$  [When it rains, the weatherman predicts rain 90% of the time.]
- $P( B | A_2 ) = 0.1$  [When it does not rain, the weatherman predicts rain 10% of the time.]

We want to know  $P( A_1 | B )$ , the probability it will rain on the day of Susan's birthday celebration, given a forecast for rain by the weatherman. The answer can be determined from Bayes' theorem, as shown below.

$$P( A_1 | B ) = \frac{P( A_1 ) P( B | A_1 )}{P( A_1 ) P( B | A_1 ) + P( A_2 ) P( B | A_2 )}$$

$$P( A_1 | B ) = (0.014)(0.9) / [ (0.014)(0.9) + (0.986)(0.1) ]$$

$$P( A_1 | B ) = 0.111$$

Figure 7.2

Note the somewhat unintuitive result. Even when the weatherman predicts rain, it only rains only about 11% of the time. Despite the weatherman's gloomy prediction, there is a good chance that Susan will not get rained tomorrow on her birthday celebration.

### Example 2: Stock Market Index

For this example, we will be using the rules and assertions of the school of thought that pertains to frequency rather than subjectivity within Bayesian probability.

This means that the measurement of knowledge that is being quantified is based on historical data. This view of the model is where it becomes particularly helpful in financial modelling. The application of how we can integrate this into our models is explained in the section to follow. Let's see how it works while incorporating Bayes' theorem within an equity market concept.

Let's say we want to know how a change in interest rates would affect the value of a stock market index. All major stock market indexes have a plethora of historical data available so you should have no problem finding the outcomes for these events with a little bit of research. For our example we will use the data below to find out how a stock market index will react to a rise in interest rates.

Here:

P(SI) = the probability of the stock index increasing  
P(SD) = the probability of the stock index decreasing  
P(ID) = the probability of interest rates decreasing  
P(II) = the probability of interest rates increasing

So the equation will be:

$$P(SD|II) = \frac{P(SD)P(II|SD)}{P(II)}$$

Thus with our example plugging in our number we get:

$$P(SD|II) = \frac{\left(\frac{1100}{2000}\right)\left(\frac{800}{1100}\right)}{\left(\frac{1000}{2000}\right)} = \left(\frac{(0.55)(0.727)}{0.5}\right) = \frac{0.3998}{0.5} = 0.799 \approx 80\%$$

In the table you can see that out of 2000 observations, 1100 instances showed the stock index decreased. This is the prior probability based on historical data, which in this example is 55% (1100/2000). This probability doesn't take into account any information about interest rates, and is the one we wish to update. After updating this prior probability



with information that interest rates have risen leads us to update the probability of the stock market decreasing from 55% to 80%. 80% is the posterior probability.

Open the file Bayes.xls in the Chapter 7 folder. Select worksheet "Stock Price."

	A	B	C	D	E	F	G	H	I	J
1		Interest Rates								
2			Decline	Increase	Unit Frequency					
3	Stock Price	Decline	300	800	1100					
4		Increase	700	200	900					
5			1000	1000	2000					
6										
7	E3/E5 >>	P(S/D) = 300 + 800 / 2000 = 1100/2000 = 0.55								
8	D3/E3 >>	P(I/SD) = 800/1100 = 0.727				>>	P(SD/I) =	(0.55)x(0.727)/(0.5) = 0.799 ≈ 80%		
9		P(I) = 1000/2000 = 0.5								
10										

Figure 7.3

### Modelling with Bayes' Theorem

As seen above we can use the outcomes of historical data to base our beliefs on from which we can derive new updated probabilities. This example can be extrapolated to individual companies given changes within their own balance sheets, bonds given changes in credit rating, and many other examples.

So what if one does not know the exact probabilities but has only estimates? This is where the subjectivists' view comes strongly into play. Many people put a lot of faith into the estimates and simplified probabilities given by experts in their field; this also gives us the great ability to confidently produce new estimates for new and more complicated questions introduced by those inevitable roadblocks in financial forecasting. Instead of guessing or using simple probability trees to overcome these road blocks, we can now use Bayes' Theorem if we possess the right information with which to start.

Now that we have learned how to correctly compute Bayes' Theorem, we can now learn just where it can be applied in financial modelling. Other, and much more inherently complicated business specific, full-scale examples will not be provided, but situations of where and how to use Bayes' Theorem will.

Changing interest rates can heavily affect the value of particular assets. The changing value of assets can therefore greatly affect the value of particular profitability and efficiency ratios used to proxy a company's performance. Estimated probabilities are widely found relating to systematic changes in interest rates and can therefore be used effectively in Bayes' Theorem.

Another avenue where we can apply our newfound process is in a company's net income stream. Lawsuits, changes in the prices of raw materials, and many other things can heavily influence the value of a company's net income. By using probability estimates relating to these factors, we can apply Bayes' Theorem to figure out what is important to us.

Once we find the deduced probabilities that we are looking for it is only a simple application of mathematical expectancy and result forecasting in order to monetarily quantify our probabilities.

## **Conclusion**

And that's Bayes' Theorem. Rational inference on the left end, physical causality on the right end; an equation with mind on one side and reality on the other. Remember how the scientific method turned out to be a special case of Bayes' Theorem? If you wanted to put it poetically, you could say that Bayes' Theorem binds reasoning into the physical universe.

To conclude, we found that by using a myriad of related probabilities we can deduce the answer to rather complex questions with one simple formula.

## General Conclusion

After reading the book, you can start to create accurate forecasts quickly and easily using **proven statistical forecasting methods**. Research has shown that no single method works best for all data, which is why I've provides a comprehensive range of popular forecasting approaches to address all types of business and individual needs.

I have also given an overview of the types of forecasting methods available. The key in forecasting nowadays is to understand the different forecasting methods and their relative merits and so be able to choose which method to apply in a particular situation. All forecasting methods involve tedious repetitive calculations and so are ideally suited to be done by a computer. Using MS Excel as a powerful forecasting tool is a good starting point before you delve into expensive forecasting software. The user's application and preference will decide the selection of the appropriate technique. It is beyond the realm and intention of the author of this book to cover all these methods. Only popular and effective methods are presented.

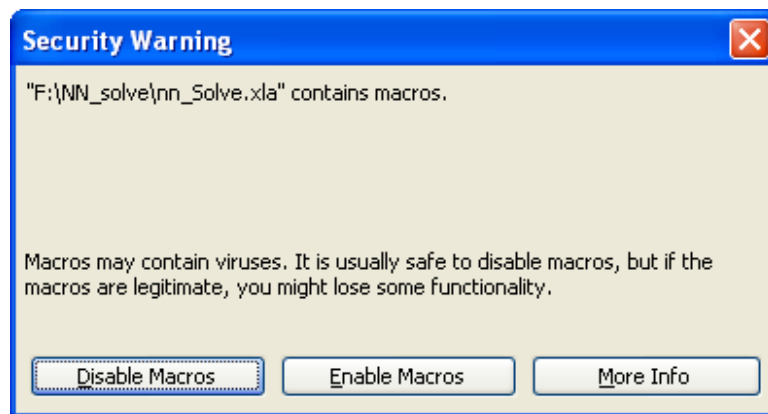
You can use this book spreadsheets models as reference to build statistically-based forecasts that is powerful yet remarkably easy to use. A full range of custom modeling spreadsheets and detailed diagnostic tools are included to support even the most sophisticated analysis. And you can do much more after reading this book, including adding your business knowledge, creating dazzling presentations and working with your existing data.

## Appendix A

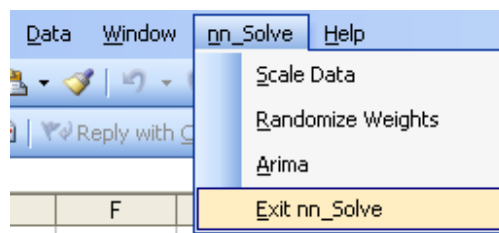
This book comes with an Excel add in **nn\_Solve** and many mathematical models developed with Excel spreadsheets.

### Installing the nn\_Solve add-in :

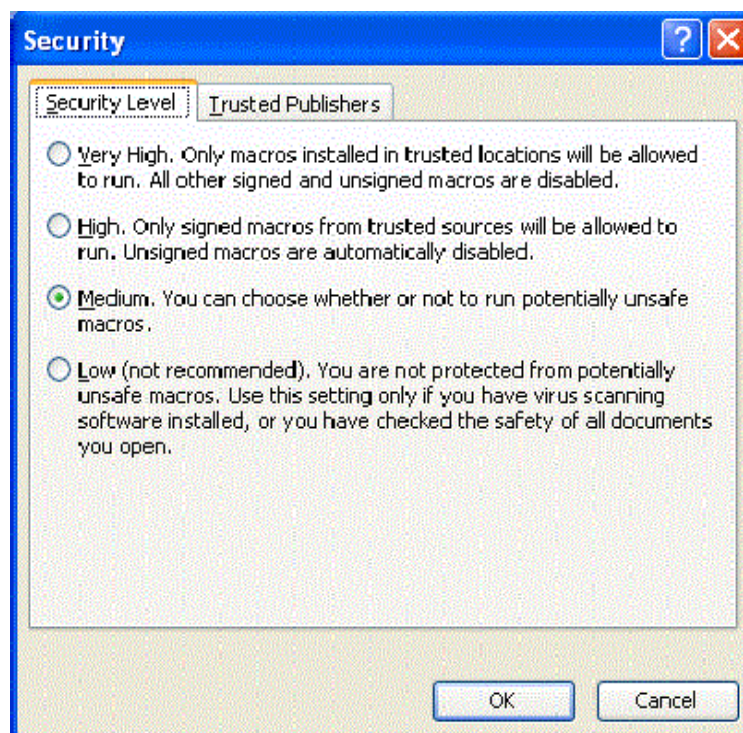
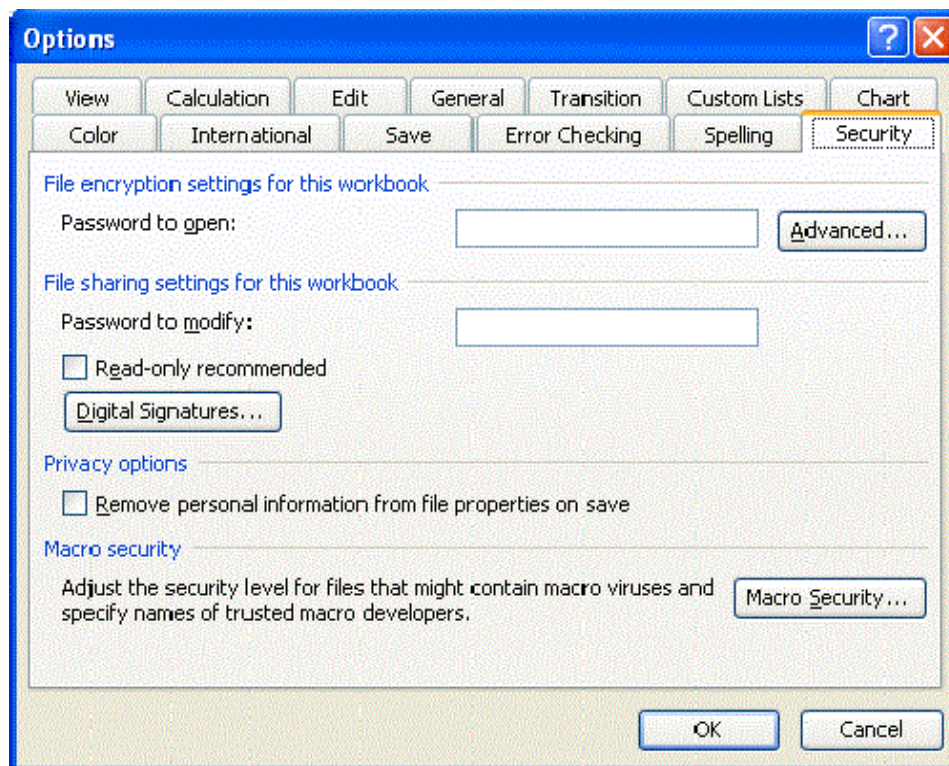
1. Open the folder nn\_Solve.zip. You will see the addin **nn\_Solve**. Double click on it. This will launch nn\_Solve.
2. Select **Enable Macros** (see below)



3. You will see **nn\_Solve** on the Excel menu bar like below:



4. If you can't open the addin **nn\_Solve**, then from the Excel menu bar, select Tools -> Options -> Security tab. Select *Macro Security* on the bottom right and Select *Medium*. (see below)



5. Double click on **nn\_Solve** icon again. This will open **nn\_Solve**

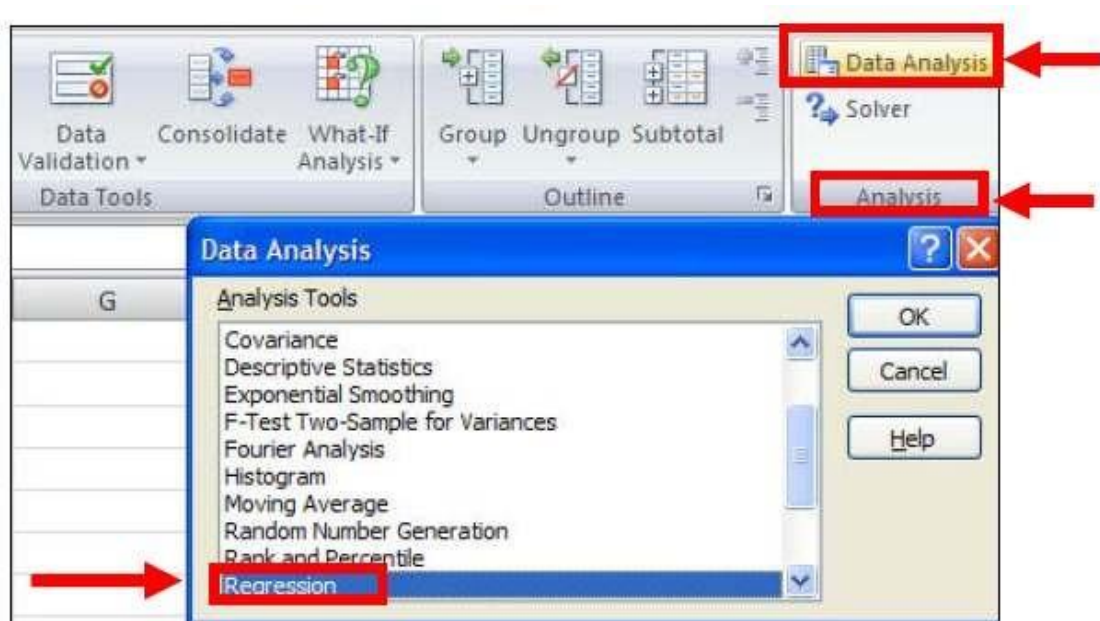
## Appendix B

### Simple Linear Regression For Forecasting

Excel 2007 has a built in regression analysis tool, packaged as part of its Analysis Toolpak. The Analysis ToolPak has a comprehensive approach to simple linear regression and returns much useful statistical in regards to the subjected data.

You can add Analysis Toolpack with these steps:

Excel options > Add-Ins > Go > Select Data Analysis Toolpack & Toolpack VBA. Data Analysis now available under Excel's Data tab



In linear regression, you are seeking to describe the association between two data sets in terms of the equation for a straight line:  $y = a + bx$ .

In this part of the book, we are going to

- How to carry out a Model I simple linear regression: drawing a regression line
- How to carry out a Model I simple linear regression: testing the significance of the association

#### Step 1

Open a new worksheet in Excel 2007 and enter the data shown in Figure A1.2

Enter the data sets in columns using appropriate labels in the first row.

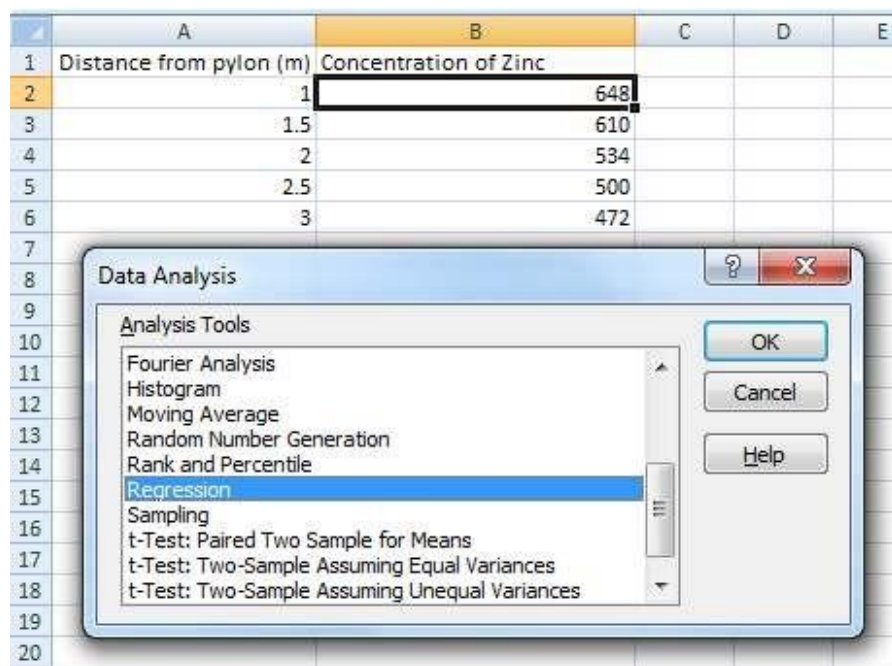


	B2	$f_x$	648
	A	B	
1	Distance from pylon (m)	Concentration of Zinc	
2	1	648	
3	1.5	610	
4	2	534	
5	2.5	500	
6	3	472	
7			
8			

*Step 2.*

*Select and click on the 'Data' tab.*

Ensure that 'Analysis ToolPak' has been added to the 'Data Analysis' tool using 'Add-ins' as described above. Click on 'Data Analysis' to open the 'Data Analysis' box. From the 'Analysis Tools' list, select 'Regression'. Click on 'OK'.



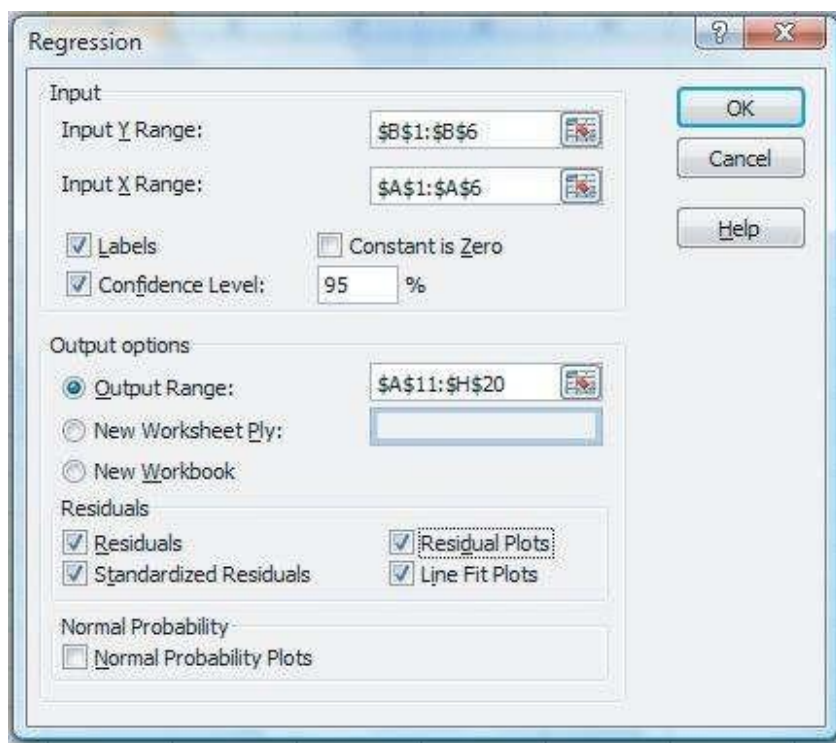
*Step 3. In the 'Regression' dialogue box that opens, enter the input variables.*

You must determine which set of data is to be plotted on the x-axis (the independent variable) and which on the y-axis (the dependent variable). In this example, 'distance from the pylon' is the independent variable and so the cell locations for these data, including the data label, must be entered into the box alongside 'Input X Range'. Click in this box to ensure that the cursor is flashing there and then click on cell A1 and, holding

down the left mouse button, drag down to cell A6 and let go. The cell locations will now be entered into the box. Repeat this procedure to enter the dependent data set into the 'Input Y Range' box. As the data labels have been included in the data range, click on the 'Labels' box so the tick appears. (If this is not done, Excel will try to read the first row as numerical data and an error will occur.)

There is an option to select 'Constant is Zero'. This will assign the value of zero to the constant term  $a$  in the equation  $y = a + bx$  and force the regression line through the origin (point 0,0). For some data sets, this property is very useful, but it is not appropriate in this example, as we anticipate a negative association between these data sets.

In the 'Input' box, you can also select the confidence level and this defaults to 95% so normally would not need to be changed.



#### *Step 4.*

#### *Outputting the data.*

There are a number of output options. First, select where your results will be returned. If you select 'Output Range', remember to click on the radio button and then also to click in the adjacent box to bring the cursor to that location (otherwise the input data will be changed). Click on a square or drag across an area to select where the results will be returned. Excel will determine the number of cells it requires for the output so you do not have to. Alternatively, you can select to have the output on a 'New worksheet' or a 'New workbook' by clicking the appropriate radio buttons.

Under 'Residuals', you can select all items by clicking in each box:



Click on 'Residuals' to include residual values in the residuals output table. Click on 'Standardized Residuals' to include standardized residual values in the residuals output table. Click on 'Residual Plots' for a chart (automatically generated) showing each independent variable versus the residual value. Click on 'Line Fit Plots' for a chart (automatically generated) showing predicted values versus the observed values. Having selected all the desired options, click on 'OK'.

11	SUMMARY OUTPUT									
12										
13	Regression Statistics									
14	Multiple R	0.984879874								
15	R Square	0.969988366								
16	Adjusted R Square	0.959984488								
17	Standard Error	14.83689096								
18	Observations	5								
19										
20	ANOVA									
21		df	SS	MS	F	Significance F				
22	Regression	1	21344.4	21344.4	96.96124	0.002226791				
23	Residual	3	660.4	220.1333333						
24	Total	4	22004.8							
25										
26		Coefficients	Standard Error	t Stat	P-value	Lower 95%	Upper 95%	Lower 95.0%	Upper 95.0%	
27	Intercept	737.6	19.90577806	37.05456767	4.32E-05	674.2509302	800.94907	674.2509302	800.9490698	
28	Distance from pylon (m)	-92.4	9.383673765	-9.846889642	0.002227	-122.2630379	-62.5369621	-122.263038	-62.5369621	
29										
30										
31										
32	RESIDUAL OUTPUT									
33										
34	Observation	Predicted Concentration of zinc (mg Zn g-1 soil)	Residuals	Standard Residuals						
35	1	645.2	2.8	0.21791368						
36	2	599	11	0.856089457						
37	3	552.8	-18.8	-1.463134708						
38	4	506.6	-6.6	-0.513653674						
39	5	460.4	11.6	0.902785245						

Step 5.

*Interpreting the data.*

The first box gives a summary output and returns values for  $R$  (the product moment correlation coefficient) and  $R^2$  (the coefficient of determination). Note that both these values are approaching 1.0 and so there is an extremely strong correlation between the two variables. Adjusted  $R^2$  gives an adjusted value in relation to the number of observations.

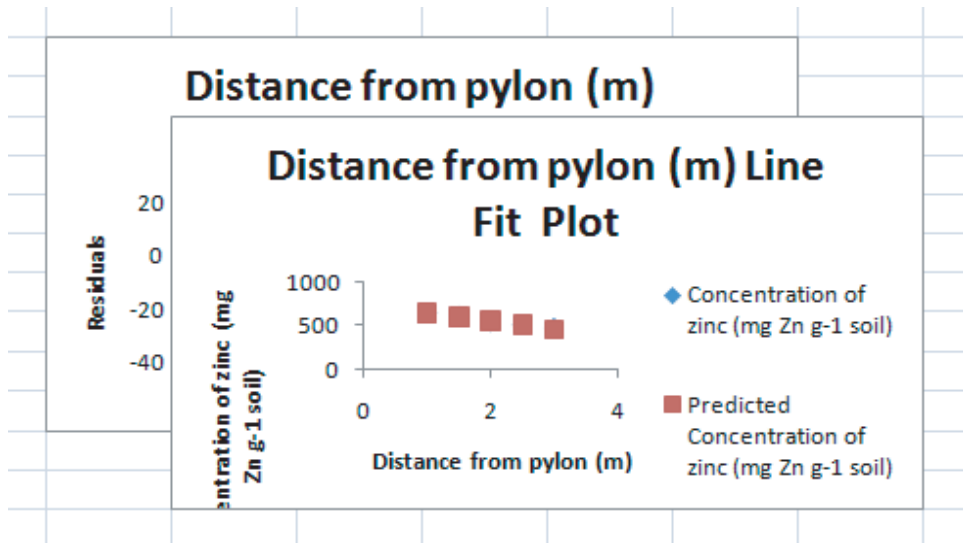
An ANOVA table is given. The ANOVA tests the significance of the regression line. A value for  $F$  is given (96.96) and its significance ( $p = 0.002$ ) and therefore the regression line is statistically significant.

In the following table, 'Intercept' gives the value for  $a$  in the equation:  $y = a + bx$ . Here  $a = 737.6$ . 'Distance from the pylon' gives the value of ' $b$ ' in the equation and is the gradient of the regression line (i.e. the gradient of the slope). Note that the value is negative, as one of the variables increases as the other decreases ( $b = -92.4$ ).

For each coefficient, values are assigned for the standard error.  $t$  statistics are also generated, which test the levels of significance of the coefficients. The probabilities for

these statistics are given; note that both are significant ( $p < 0.005$  for the gradient (actual value  $p = 0.002$ ) and  $p < 0.00005$  for the intercept value,  $p = 4.32 \times 10^{-5}$ ).

Residual values are given: these indicate the distance of the actual data points from the regression line. These values can be plotted above and below a horizontal line representing the regression line in the residual plot chart, which facilitates a visualization of the spread of values around the line. The charts are returned 'stacked'. It is necessary to drag them apart and edit the charts as required – particularly expanding the plot area (see below for line fit plot).

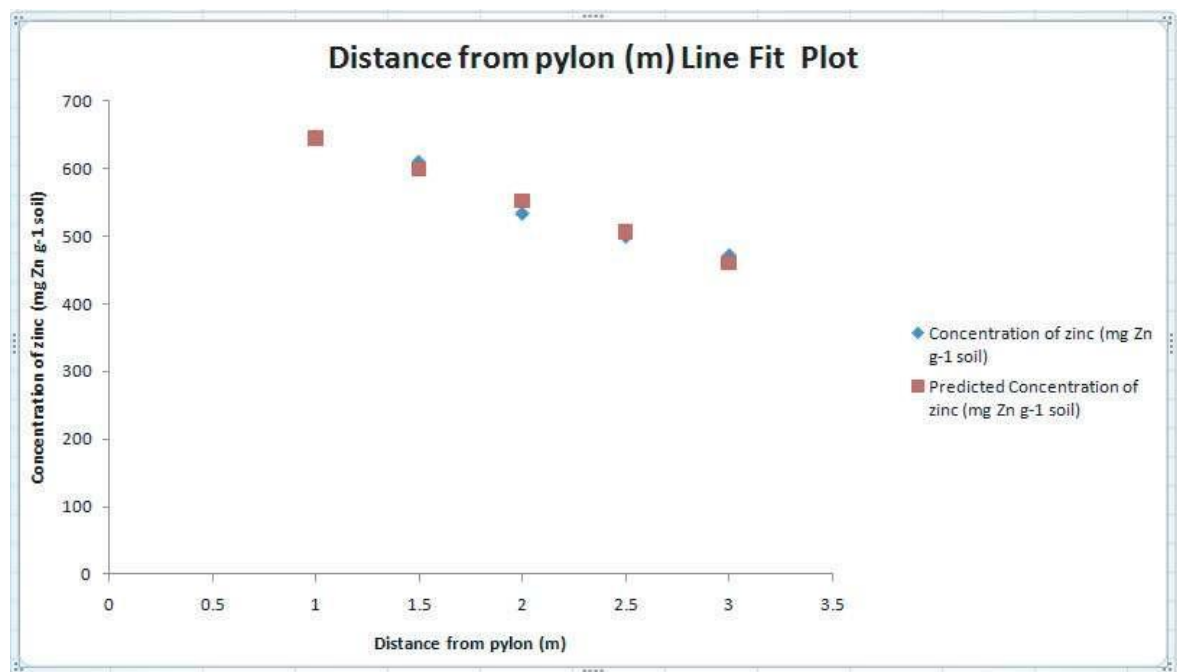


Step 6.

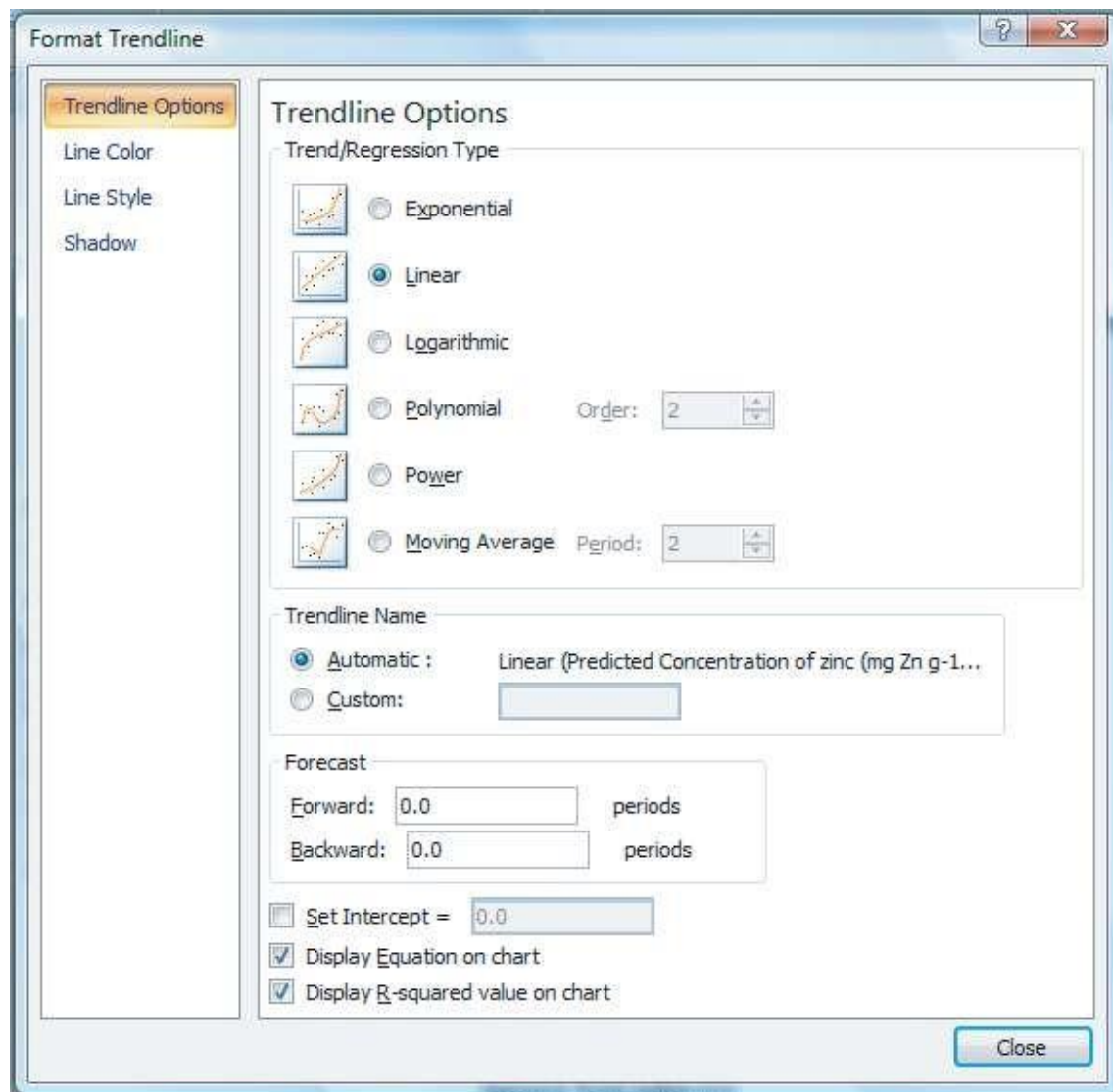
*Presenting data.*

Click on the top chart to select it and drag it to an empty part of the worksheet. Do the same to the second chart. Now expand and edit the charts to improve their legibility and appearance. Having expanded the line fit plot, you can see the actual values plotted as blue diamonds and the 'predicted' values as red squares. The predicted values give the position of the regression line calculated from the data. In this chart, the two sets of values almost coincide.

In order to add the regression line, move the cursor over one of the red squares and click with the right mouse button. This highlights all the red squares and they appear to have a blue cross behind them. Simultaneously, a pop-up menu appears.



From the menu, click on 'Add Trendline' and a new box will open. Here you can select the type of trend line required and also format it.



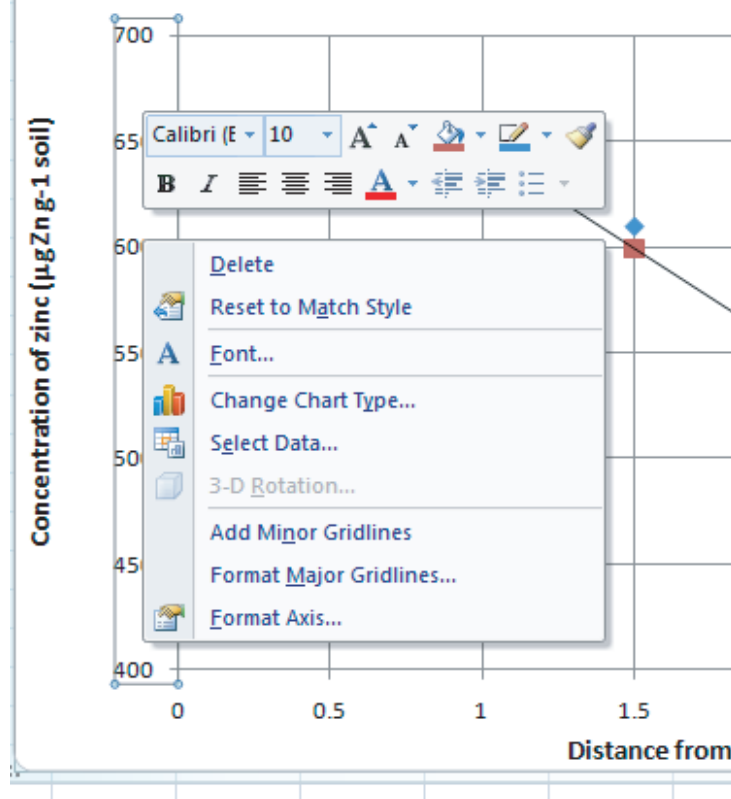
Here we will choose the 'linear' trendline and also tick the boxes for 'Display Equation on chart' and 'Display R-squared on chart'. You can also select trendline options for colour, style, and shadow on the trendline.

The 'Forecast' box enables the fitted regression line to be extended either side of the regression line. This is a useful facility in some cases, but note that, on most occasions with biological and environmental data, this must not be done, as it would be erroneous to assume values beyond the data points. The next option is to set the intercept to zero or another chosen value. This is not required in this example.

To remove the redundant lower values on the y-axis, click on the y-axis to highlight it and make sure the formatting boxes are open. Click on the attributes you want to change and select appropriate changes from the dialogue boxes.

To add grid lines, ensure that the chart is selected by clicking on it. The 'Chart Tools' should now become available on the tool bar. Click on the 'Layout' tab and then 'Gridlines' from the 'Axes' section.

## Variation of Zinc concentration in soil



Format Axis

**Axis Options**

Number

Fill

Line Color

Line Style

Shadow

3-D Format

Alignment

Minimum: ☐ Auto ☒ Fixed 400.0

Maximum: ☒ Auto ☐ Fixed 700.0

Major unit: ☒ Auto ☐ Fixed 50.0

Minor unit: ☒ Auto ☐ Fixed 10.0

☐ Values in reverse order

☐ Logarithmic scale Base: 10

Display units: None

☐ Show display units label on chart

Major tick mark type: Outside

Minor tick mark type: None

Axis labels: Next to Axis

Horizontal axis crosses:

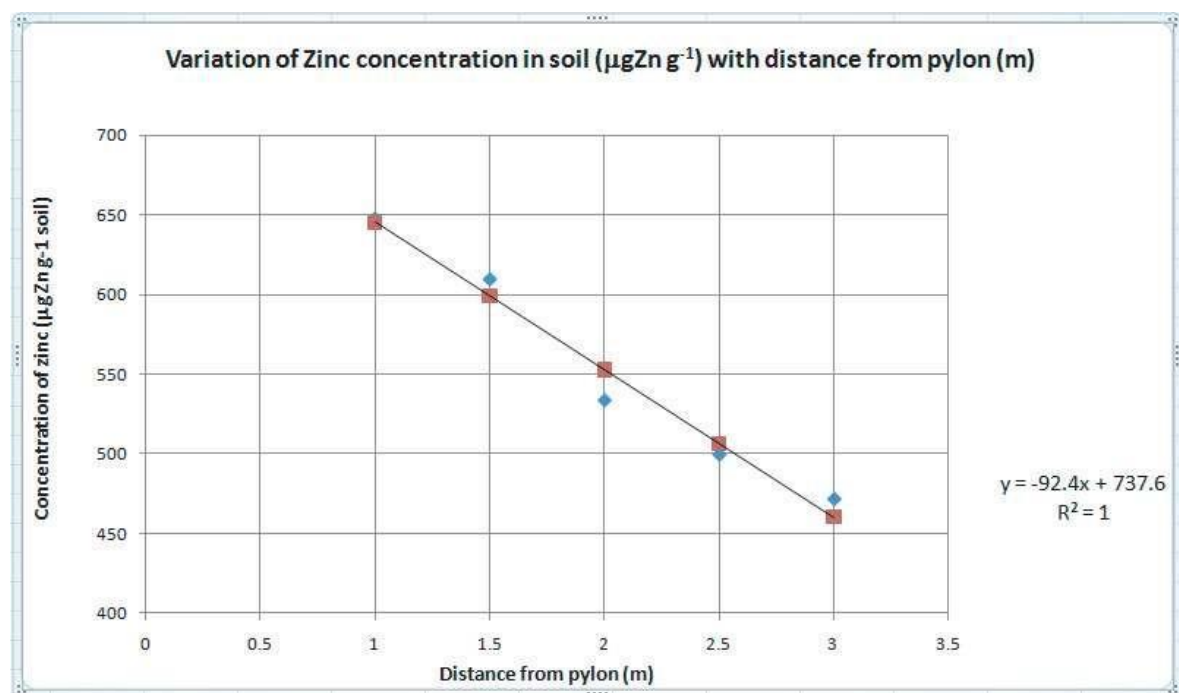
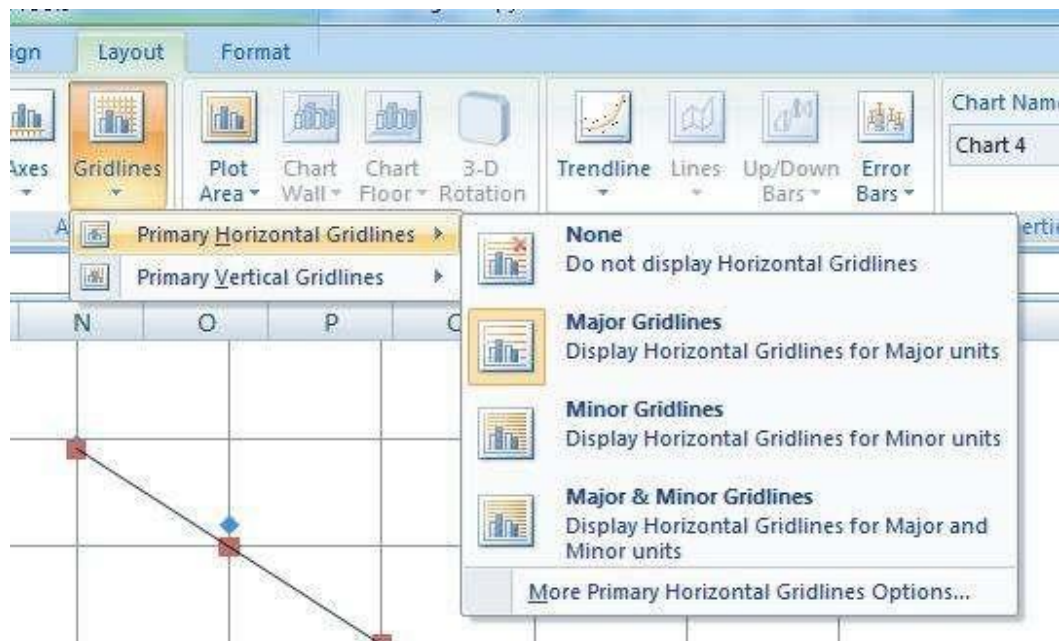
☒ Automatic

☐ Axis value: 400.0

☐ Maximum axis value

Close

Regression line Equation and  $R^2$  values



The 'Residuals plot' can also be expanded in the same way as the line fit plot. Here the relationship of the actual data points to the regression line is shown. The regression line is shown horizontally and the data points lie above and below this line where the values are greater or less than the regression line, respectively. The greater the deviation of the points from the regression line, the smaller would be the value for  $R^2$ . The chart above shows the residual plot of the association between zinc concentration in the soil and distance from the pylon (m).

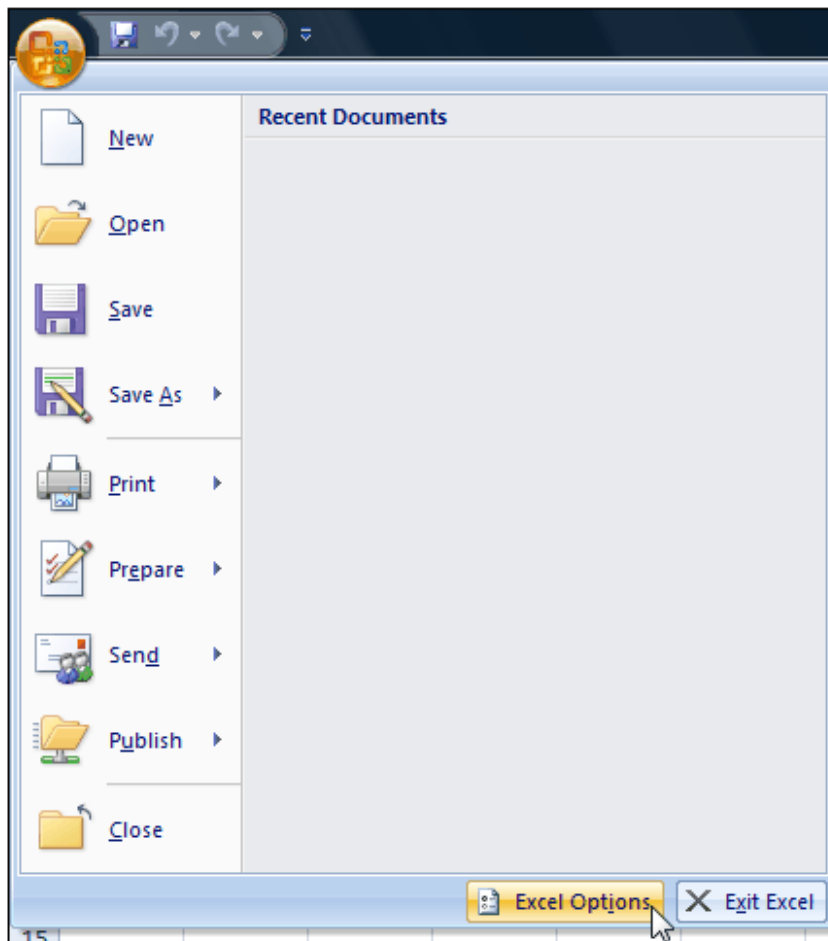
## Appendix C:

### Accessing Excel Solver 2007/2010/2016

The Solver Add-in is a Microsoft Office Excel add-in program that is accessible when you install Microsoft Office or Excel. To use it in Excel, however, you need to load it first.

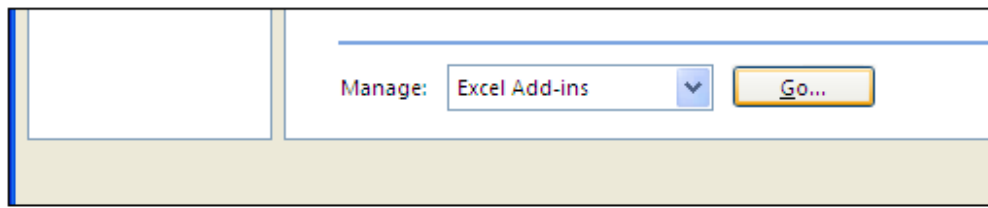
**Follow the instructions below to load the Solver Add-in:**

1. Click the **Microsoft Office Button**, and then select **Excel Options** from the bottom right-hand corner of the box.



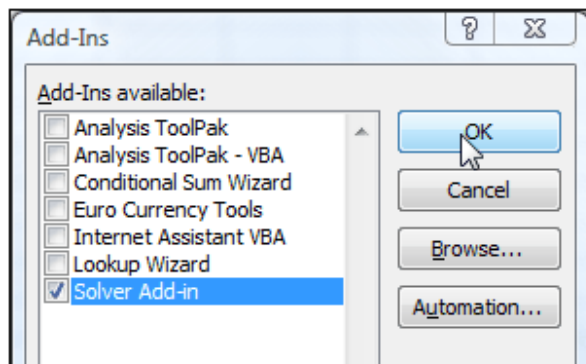
2. Switch to the **Add-Ins** tab, and then in the **Manage** box, select **Excel Add-ins**. Then click **Go**.





3. In the **Add-Ins available** box, select the **Solver Add-in** check box, and then click **OK**.

**NOTE:** If Solver Add-in is not listed in the Add-Ins available box, click **Browse** to locate the add-in. If you get prompted that the Solver Add-in is not currently installed on your computer, click **Yes** to install it.



4. After you load the Solver Add-in, the **Solver** command is available in the **Analysis** group on the **Data** tab. (right hand side)

