

COPYRIGHT NOTICE:

**David A. Kendrick, P. Ruben Mercado, and Hans M. Amman:  
Computational Economics**

is published by Princeton University Press and copyrighted, © 2006, by Princeton University Press. All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher, except for reading and browsing via the World Wide Web. Users are not permitted to mount this file on any network servers.

Follow links Class Use and other Permissions. For more information, send email to: [permissions@pupress.princeton.edu](mailto:permissions@pupress.princeton.edu)

## Neural Nets in Excel

---

---

Much of economics is about finding optimal variables given parameters that describe human behavior. For example, in the optimal growth model that we solved with Excel in Chapter 1 the goal was to find the optimal levels of the consumption and capital stock variables given the parameters of the production and utility functions.

In this chapter we invert this duality. We begin with the observed behavior and attempt to find the parameters that permit the specified relationships to fit the data most closely. Such is the subject matter of econometrics and estimation. However, we are looking at a type of estimation that until recently had not been in the mainstream of econometrics but rather developed in other fields and is now increasingly being used to fit economic relationships—namely neural nets.

Neural network models are suitable for dealing with problems in which the relationships among the variables are not well known. Examples are problems in which information is incomplete or output results are only approximations, as compared to more structured problems handled, for example, with equation-based models. Neural networks are particularly useful for dealing with data sets whose underlying nonlinearities are not known in advance.<sup>1</sup> Among the many possible applications are forecasting and identification of clusters of data attributes.

The example we use here is typical of the applications of neural nets to economics and finance—how best to predict the future prices of a stock.<sup>2</sup> The stock we use is that of the Ford Motor Company, and we attempt to predict its future share price by using the share price of a group of related companies—companies that provide inputs to automobile production and companies that produce competing vehicles.

The central notion of neural net analysis is that we can use a set of observations from the past to predict future relationships. Thus we use the closing price of Ford

1. One of the strengths of neural net methods is that they can approximate any functional shape.

2. Neural nets are not necessarily a better way to predict stock prices than standard econometric methods; however, stock prices offer a clear and motivating example for many students, so we use that example here.

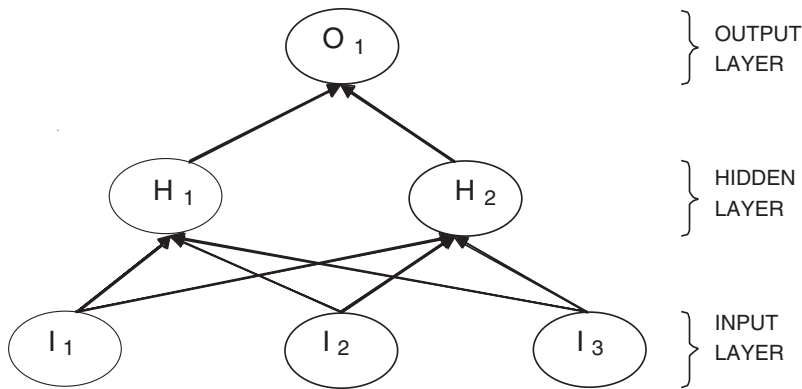


Figure 2.1. Neural net layers.

stock each week over a 14-week period to “train” the model and then use the parameters that emerge from the training to predict the Ford stock price in the 15th through 20th weeks. This is done in an Excel spreadsheet using the Solver that we first used in the growth model.

The chapter begins with an introduction to neural nets followed by the specification of an automobile stock price model. We then introduce the data that are used in the model, the representation of the model in Excel, and the use of the Excel Solver to find the best parameter values.

## 2.1. NEURAL NET MODELS

Neural networks (or, more properly, artificial neural networks) are inspired by, or up to a point are analogous to, natural neural networks. They have three basic components: processing elements (called nodes or neurons), an interconnection topology, and a learning scheme. From a computational point of view, a neural network is a parallel distributed processing system. It processes input data through multiple parallel processing elements, which do not store any data or decision results as is done in standard computing. As successive sets of input data are processed, the network processing functions “learn” or “adapt,” assuming specific patterns that reflect the nature of those inputs.

There are many alternative network architectures. Let us look now in more detail at the elements, architecture, and workings of a neural network as shown in Figure 2.1. This is known as back-propagation or as a feed-forward model, which is the type most commonly used. This is a simple network with one input layer with three neurons, one intermediate layer with two neurons (usually named the “hidden layer”), and one output layer with just one neuron. A key component of the network is the neuron, an elementary processing unit that, given inputs, generates output. It

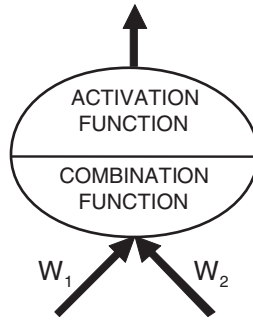


Figure 2.2. Activation and combination functions.

is composed of two main parts: a combination function and an activation function (Fig. 2.2). The combination function computes the net input to the neuron, usually as a weighted sum of the inputs. The activation function generates output given the net input.

It is standard procedure to constrain the output of a neuron to be within an interval (0,1). To do so, different functional forms can be used for the activation function, such as logistic functions, sigmoid functions, and so on. Furthermore, a threshold may be used to determine when the neuron will “fire” an output as the activation function yields a value above that threshold. Input layer neurons receive data (“signals”) from outside and in general transmit them to the next layer without processing them. Output layer neurons return data to the outside and are sometimes set to apply their combination functions only.

The learning process of the network consists of choosing values of the weights so as to achieve a desired mapping from inputs to outputs. This is done by feeding the network with a set of inputs, comparing the output (or outputs, in case of having more than one network output) to a known target, computing the corresponding error, and sometimes applying an error function. Then weights are modified to improve the performance. To do this, a variety of methods can be employed, such as Newton or conjugate gradient in Excel, which are discussed later in this chapter.

## 2.2. THE AUTOMOBILE STOCK MARKET MODEL

We begin with the specification of the combination function for the output layer as

$$y_t = \theta_0 + \sum_{j=1}^q \theta_j a_{tj} \quad (1)$$

where  $y_t$  is the output in period  $t$ ,  $a_{tj}$  is the hidden node value in period  $t$  for node  $j$  and the  $\theta_j$ 's are parameters. There are  $q$  hidden nodes. In our model the  $y_t$  variables

are the share price of the Ford Motor Company stock in each of the 14 weeks in 1997.

The  $\theta$ 's are among the parameters that we are seeking. The  $a_{tj}$ , which are the values in time period  $t$  at hidden node  $j$ , are given by the expression

$$a_{tj} = S\left(\sum_{i=1}^{q_j} w_{ji} x_{it}\right) \quad (2)$$

where the  $x_{it}$  are the inputs at node  $i$  in period  $t$ . There are  $q_j$  inputs at hidden node  $j$ .

The  $x_{it}$  are the share prices of the other companies in our example. The  $w_{ji}$  are the parameters at the  $j$ th hidden node for the  $i$ th input and are the second set of parameters that we want to choose. Thus, in summary, we are given the share prices of the other companies  $x_{it}$  and the share price of the Ford stock  $y_t$  and are seeking the parameters  $\theta$  and  $w$  that permit our functions to fit the data most closely.

What functions are being used? The first function, in Eq. (1), is linear and the second function, in Eq. (2),  $S$ , is a sigmoid function, with the mathematical form

$$S(z) = \frac{1}{1 + e^{-z}} \quad (3)$$

One can quickly see by examination that this function evaluated at  $z = 0$  is

$$S(0) = \frac{1}{1 + e^{-0}} = \frac{1}{1 + 1} = \frac{1}{2} \quad (4)$$

and that large negative values of  $z$  map to near zero, that is,

$$S(-5) = \frac{1}{1 + e^5} = 0.007 \quad (5)$$

and that large positive values of  $z$  map to approximately one, that is,

$$S(5) = \frac{1}{1 + e^{-5}} = 0.993 \quad (6)$$

Thus the function has the shape shown in Figure 2.3. It is sometimes called the “squasher” and it is quickly apparent why. Given any data set of numbers that range from very large negative numbers to very large positive numbers this function maps those numbers to the zero-to-one interval while maintaining their relative sizes.

The example we present here was developed by Joe Breedlove. This example contains share prices from the automotive suppliers of Ford in 1997, that is,

Bethlehem Steel  
 Owens Glass  
 Goodyear Tire and Rubber

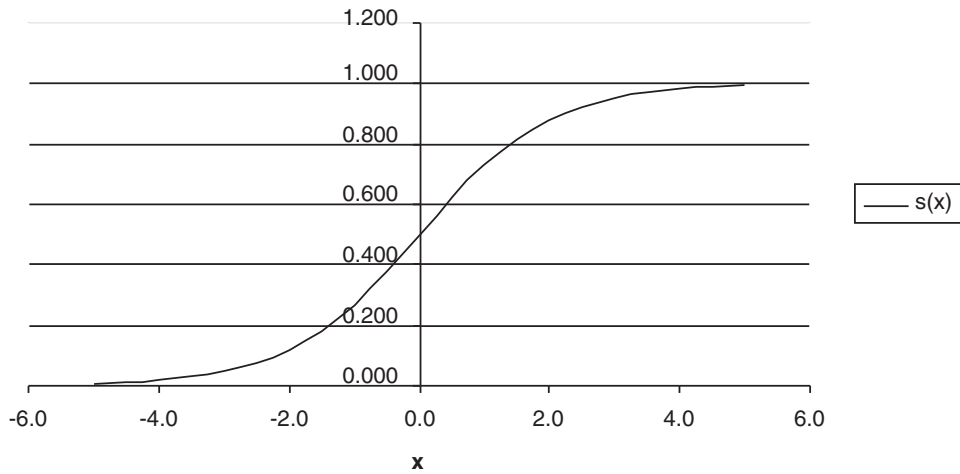


Figure 2.3. The sigmoid function.

and the competing auto makers to Ford, that is,

Chrysler  
General Motors

to predict the share price of the

Ford Motor Company

At that time stock prices were quoted as fractions rather than as decimals and the data in the spreadsheet reflect this fact. Ford's suppliers and competitors have changed since 1997, but the example is useful as a starting place for learning about neural nets.

The effect from the suppliers is aggregated into one hidden node and that from the competitors is aggregated into the second hidden node, as shown in Figure 2.4. So for the example at hand,

$$z_1 = w_{11} * x_1 + w_{12} * x_2 + w_{13} * x_3 \quad (7)$$

and

$$a_{t1} = \frac{1}{1 + e^{-(w_{11} * x_1 + w_{12} * x_2 + w_{13} * x_3)}} \quad (8)$$

$$z_2 = w_{21} * x_4 + w_{22} * x_5 \quad (9)$$

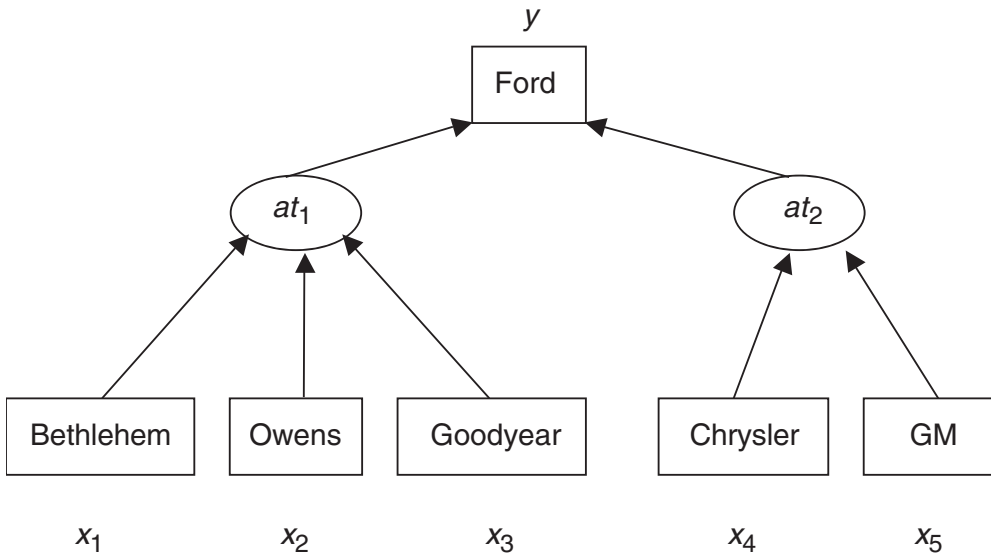


Figure 2.4. A neural net for Ford Motor Company share prices.

and

$$a_{t2} = \frac{1}{1 + e^{-(w_{21}x_4 + w_{22}x_5)}} \quad (10)$$

In addition,

$$\hat{y}_t = \theta_0 + \theta_1 a_{t1} + \theta_2 a_{t2} \quad (11)$$

Thus the optimization problem in Excel is to find the values of

$$w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, \theta_0, \theta_1, \theta_2 \quad (12)$$

that minimize the square of the separation between the predicted and actual values of the  $y$ 's, that is,

$$Norm = \sum_{t=1}^n (y_t - \hat{y}_t)^2 \quad (13)$$

where  $n$  is the number of observations, which for this example is fourteen.

### 2.3. THE DATA

Closing stock prices for Ford and for the three suppliers (Bethlehem, Owens, and Goodyear) and the two competitors (Chrysler and General Motors) for each week in January, February, and March 1997 were used as shown in Table 2.1. As mentioned earlier, at that time stock prices were listed as fractional numbers, rather than as decimal numbers, as is now the case.

Table 2.1. Share Prices of Ford and Related Companies

Week	Ford	Bethlehem	Owens	Goodyear	Chrysler	General Motors
Closing	y	x1	x2	x3	x4	x5
Jan 3	32½	9¼	42½	52¾	34%	57%
Jan 10	33¾	8%	49	54½	35%	61%
Jan 17	33	9	48%	55	34%	60%
Jan 24	33%	8%	45%	54¼	35%	62½
Jan 31	32%	8%	46%	54½	34%	59
Feb 7	32¼	8¼	45½	52½	34%	56¼
Feb 14	32%	7%	44¼	53%	34½	58%
Feb 21	33%	7%	43%	53%	35%	58½
Feb 28	32%	8¼	42%	52¼	34	57%
Mar 7	32¼	8%	42%	53%	31%	56%
Mar 14	32%	8½	42½	53%	30%	58
Mar 21	31¼	8¼	40%	54½	30%	57
Mar 27	30%	8½	40%	54¼	30%	56¼
Mar 31	31%	8¼	40%	52%	30	55%

## 2.4. THE MODEL REPRESENTATION IN EXCEL

Here we follow the representation of a neural net in Excel developed by Hans Amman and combine this with Joe Breedlove’s model of Ford share prices. The input file for Excel for this example can be obtained from the book web site. Once you have downloaded the file you can begin by opening it in Excel as is shown in Figure 2.5.

Skip down to the section on the data set beginning in line 17 and note that there are fourteen observations consisting of the weekly closing share price y for Ford shares and the five inputs x1 through x5 for the other stocks. These observations are aggregated using the sigmoid function into the hidden layers at1 and at2 using a formula like

$$at1 = 1 / (1 + \text{Exp}(-(D20*D5 + E20*D6 + F20*D7)))$$

where the D5, D6, and D7 are weights that are to be solved for and the D20, E20, and F20 are the observations x1, x2, and x3. You can see this formula in the spreadsheet by selecting the I20 cell and then looking at the expression in the formula bar at the top of the spreadsheet. Alternatively, you can see all of the formulas in the spreadsheet by selecting Tools:Options:Views and then checking the Formula box.

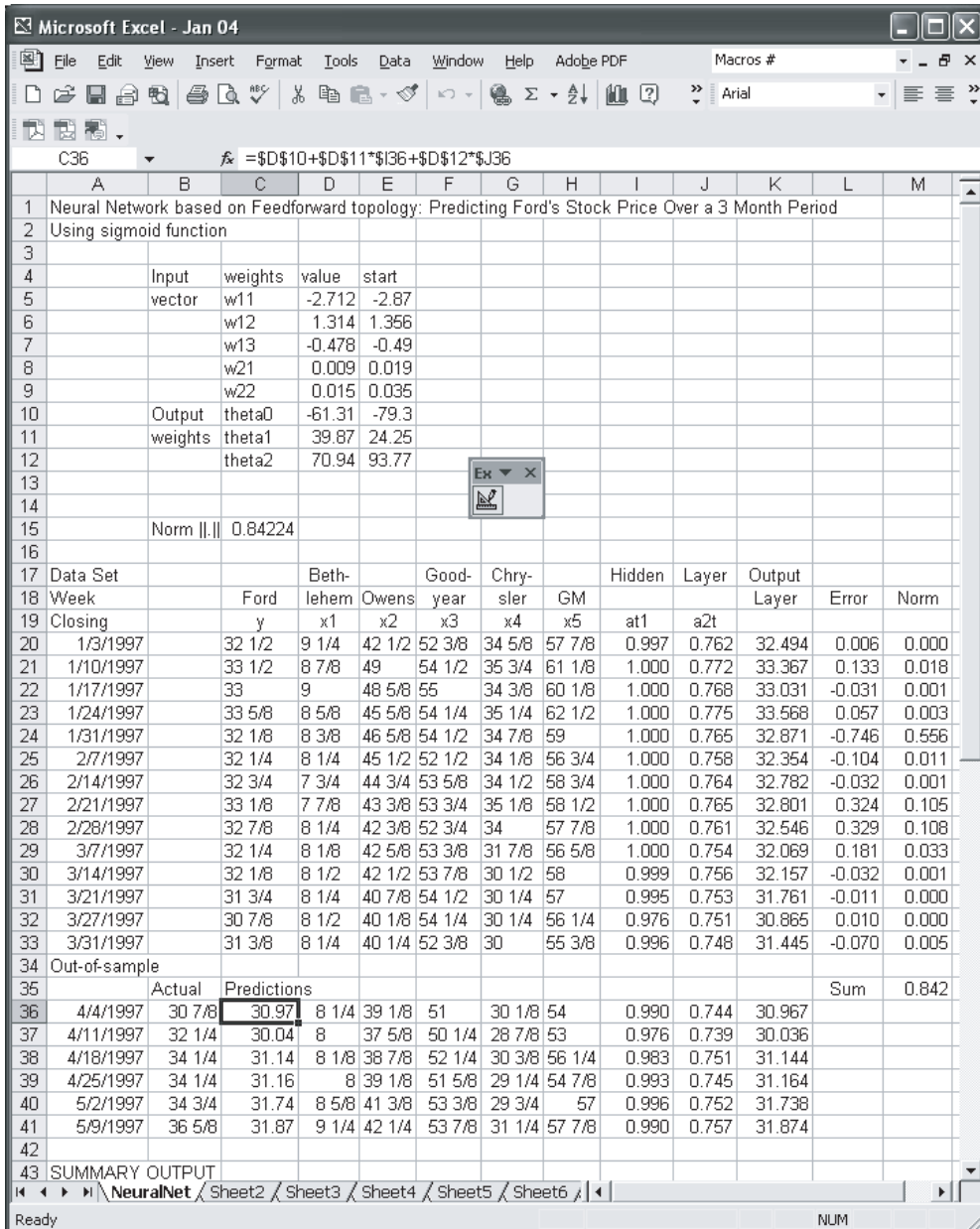


Figure 2.5. Spreadsheet for neural nets with stock prices.

Now back to the Data Set section of the spreadsheet. Check the column at2 and you will find that it is similar to the column at1 except that it uses input data for x4 and x5 to compute the second of the two hidden layer values.

Consider next the Output Layer column, which is computed using an expression of the form

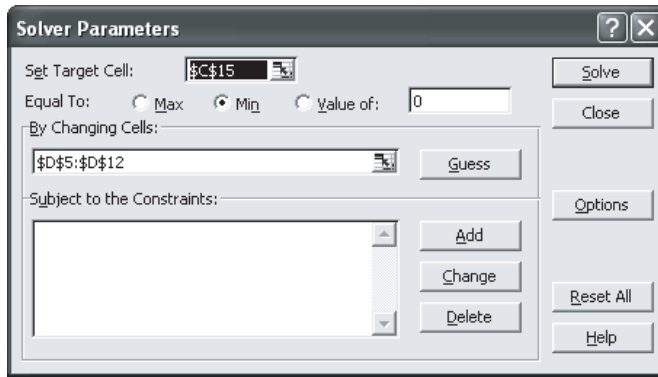


Figure 2.6. The Solver dialog box.

$$\text{Output} = \text{theta0} + \text{thetal} * \text{at1} + \text{theta2} * \text{at2}$$

where the *thetas* are weights that are computed in the optimization and are shown in the section on *Output weights* near the top of the spreadsheet.

Next look at the *Error* column in the *Data Set* section of the spreadsheet. This column is simply the difference

$$\text{Error} = y - \text{Output}$$

and the *Norm* column is the square of the elements in the *Error* column. The elements in the *Norm* column are summed up in cell *M35* at the bottom of the column.

Now we are ready for the optimization problem. It is seen by selecting *Tools:Solver* at which point the dialog box shown in Figure 2.6 should appear.<sup>3</sup> This dialog box indicates that the optimization problem is to minimize the value in cell *C15* (which on inspection is set equal to *M35*, which in turn is the sum of the elements in the *Norm* column).

As was discussed earlier, the Excel Solver uses nonlinear optimization methods (Newton method or conjugate gradient method—see Appendix F). The optimization is done by changing the elements in cells *D5* through *D12* until the minimum of the function is obtained. These cells are shown in Table 2.2 beginning with the number *-2.712* and going down the value column to the element *70.94*. The column to the right labeled *start* shows the numbers that were originally used when searching for the optimal parameters. They are not used in the present calculations but are stored there only to indicate which starting values were used. In fact each time the model is solved the numbers in the *value* column are used as the starting point and an effort is made to find values that decrease the norm. For a first experiment you

3. In case the dialog box does not appear, see Appendix C.

Table 2.2. Parameters

Input	weights	value	start
vector	w11	-2.712	-2.87
	w12	1.314	1.356
	w13	-0.478	-0.49
	w21	0.009	0.019
	w22	0.015	0.035
output	theta0	-61.31	-79.3
weights	theta1	39.87	24.25
	theta2	70.94	93.77

might try changing some of the elements in the `value` column, selecting `Tools:Solver` and then clicking on the `Solve` button to solve the optimization problem and see if the parameters return to the original values or converge to some others that have either a smaller or a larger norm.

A point of caution: at times the solution procedure converges to a result with a higher norm because neural net estimation problems are sometimes characterized by nonconvexities and may have local optimal solutions that are not the same as the global optimal solution. Sometimes the number of local solutions may be very large. Thus in Excel it may be advisable to use a number of different starting values in order to check for global convergence. When there are many local optima, global optimization algorithms such as genetic algorithms can be used to perform global exploration of the solution space [see Chapters 11 and 12 or Goldberg (1989)].

In addition, you can experiment by changing some data elements in the `y` and `x` columns either in an arbitrary manner or by looking up the share prices for these companies in another time period and seeing whether the parameter values have remained the same.

Finally the spreadsheet contains some forecast in the section called `Predictions`. These predictions are made for 6 weeks after the last week for which data were collected to “fit” or “train” the model. Look at the formulas for cells B36 and C36 that are shown in Table 2.3.

If you select the cell just beneath the `Predictions` label you see that the predictions use an expression such as

$$= D10 + D11*I36 + D12*J36$$

that translates to

$$\text{Output} = \text{theta0} + \text{theta1} * \text{at1} + \text{theta2} * \text{at2}$$

Table 2.3. Predictions

Out-of-sample		
	Actual	Predictions
4/4/1997	30%	30.97
4/11/1997	32%	30.04
4/18/1997	34%	31.14
4/25/1997	34%	31.16
5/2/1997	34%	31.74
5/9/1997	36%	31.87

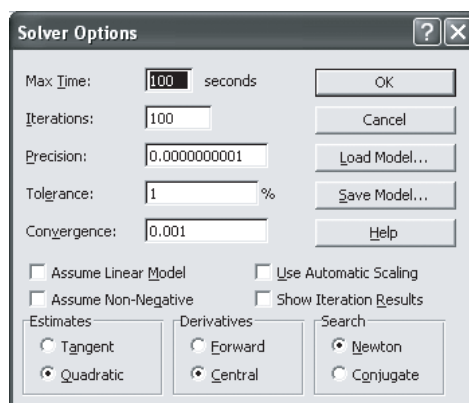


Figure 2.7. The Solver options dialog box.

Note in particular that these predictions are done from “out of sample” data, that is, the data that are used to fit the model are not used to make the predictions. Rather some elements of the sample are reserved to test the model after it is fitted to a subset of the data.

There is one other topic that needs to be mentioned about the Excel Solver. Select `Tools:Solver:Options` and the dialog box shown in Figure 2.7 appears. You can use this dialog box to control the number of iterations that the Solver will use in trying to achieve convergence. Keep the number of iterations low when you are first working with a new data set and then if convergence is not being achieved raise this number as necessary. Moreover, a convergence value of 0.001 is probably close enough for most of the work you do, but you may require a looser convergence by lowering this setting to 0.01 in order to obtain convergence in 100 iterations. On the other hand, you may want to keep the convergence value at 0.001 and increase the number of iterations.

Probably the most important element in the Solver options dialog box is `Use Automatic Scaling`. In many neural net data sets the various series may be of very different magnitudes. For example, you might have an unemployment series with numbers of the size of 0.04 and a consumption series with numbers like 625. In such a case it is wise to check the automatic scaling option. If you do this, the Solver will automatically scale all of your series so that they are of roughly the same magnitude and thereby increase the probability that the Solver will be able to find an optimal set of parameter estimates.

## 2.5. EXPERIMENTS

There are two kinds of experiments that come to mind with this spreadsheet. As discussed earlier, at the simplest level you can change the data in the  $y$  and  $x$  columns and see how the weights and predictions change. You could even use some kind of data of your own for doing this. Some students who have a greater interest in professional sports than in the stock market have used offensive and defensive statistics from basketball teams to predict the point spread in playoffs.

Moreover, you can change the number of input series  $x_1$  through  $x_5$  by adding series such as  $x_6$  and  $x_7$  for other automotive companies such as Toyota and Honda. However, this is somewhat harder to do than the experiments discussed previously as it involves making changes in the formulas in the spreadsheet. On the other hand, it is a very good way to really learn how a neural net is represented and solved in a spreadsheet.

## 2.6. FURTHER READING

Sargent (1993) provides an introduction to neural nets. Garson (1998) presents an introduction to the use of neural networks in the social sciences and a systematic coverage of the subject. Beltratti, Margarita, and Terna (1996) also present an introduction to neural networks and develop a variety of models for economic and financial modeling.