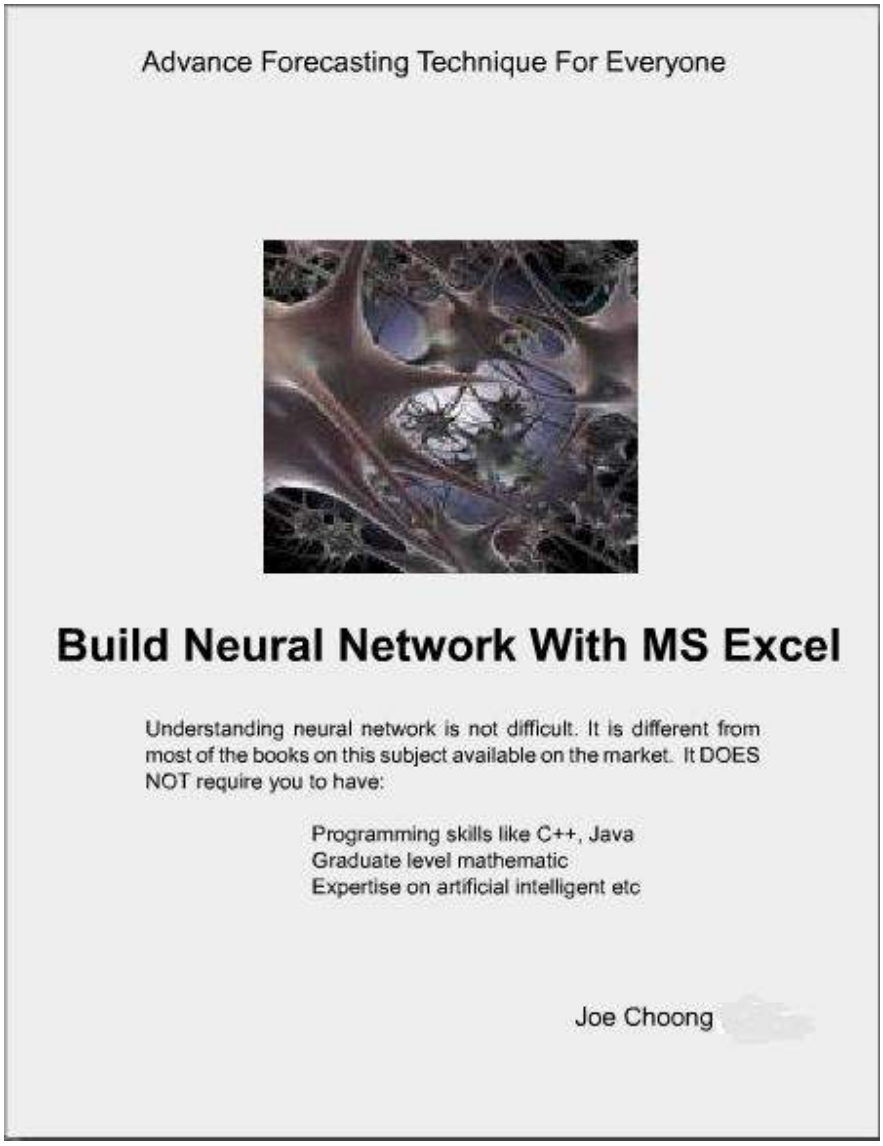


Build Neural Network With MS Excel ®
Published by XLPert Enterprise

Table Of Contents

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 millenium.

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 modelling 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).

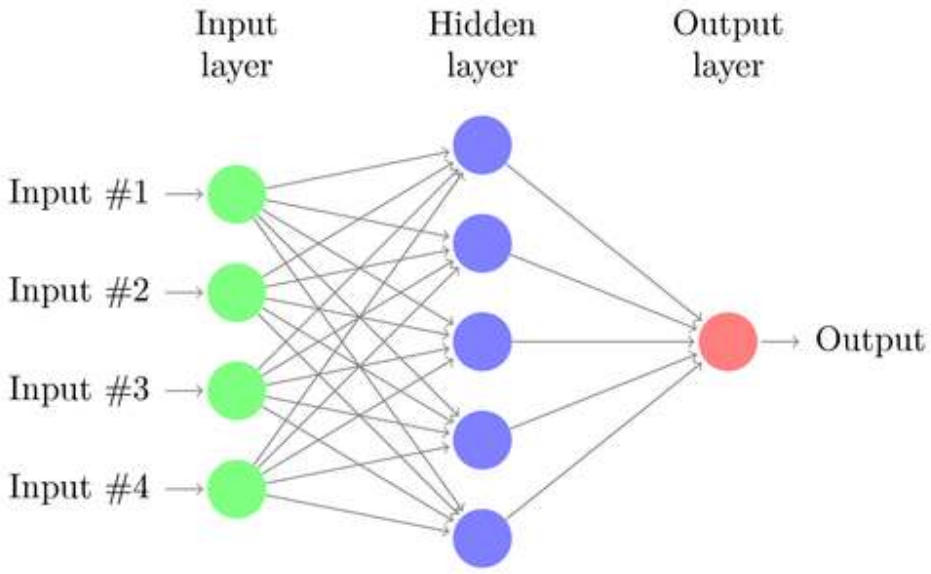


Figure 1.1

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:

p/g 6 not available for viewing

x

x

x

This part of the book is not available for viewing

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

x

x

x

x

x

To make a neural network performs some specific task, we must choose how the units are connected to one another (see Figure 1.1), 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

Neural Network Architecture

Feed-forward networks have the following characteristics:

1. Perceptrons are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers.
2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one layer to the next., and this explains why these networks are called feed-forward networks.
3. There is no connection among perceptrons in the same layer.
4. The number of nodes in the input, hidden and output layer follows a pyramidal rule

   -i.e, if you have 5 nodes in the input layer and 1 nodes in the output layer, then the hidden layer shall have 4 or 3 or 2 nodes. (5-4-1 or 5-3-1 or 5-2-1, see the hierarchy or pyramid structure here)

   However, this is just a general rule and need not be followed strictly.

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.

Open the file Credit_Approval.xls. 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)* 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 1.2

Now, we can see that Column A to column L in the *worksheet (Transform Data)* are all numerical. (see Figure 1.2 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 value 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 1.3)

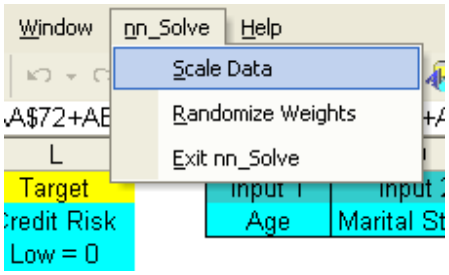


Figure 1.3

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 1.4 below)

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

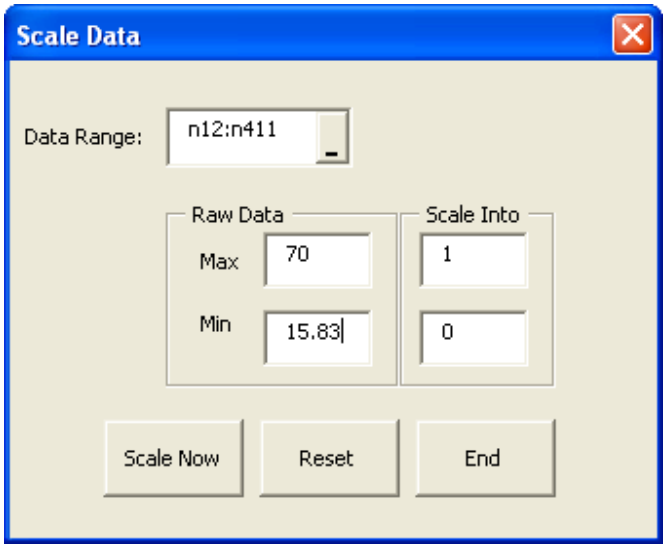


Figure 1.4

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 1.5 below)

	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 1.5

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 value 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 =	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 1.6

Figure 1.6 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:

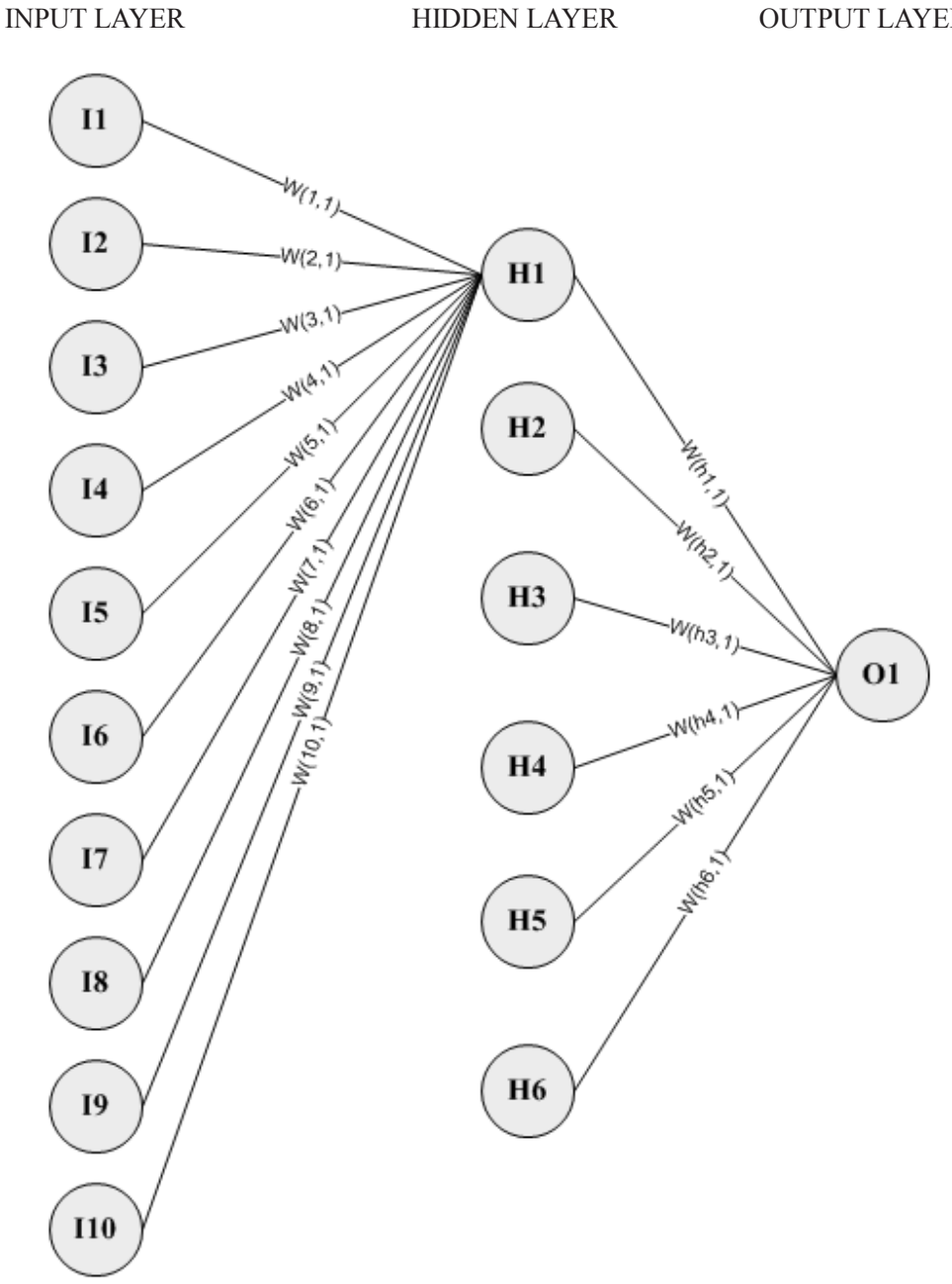


Figure 1.7

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 1.1 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 1.7 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 1.7, 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 x 6) + (6 x 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) = AA12 -> connecting I1 to H1

w(2,1) = AA13 -> connecting I2 to H1

w(3,1) = AA14 -> connecting I3 to H1

w(4,1) = AA15 -> connecting I4 to H1

w(5,1) = AA16 -> connecting I5 to H1

w(6,1) = AA17 -> connecting I6 to H1

w(7,1) = AA18 -> connecting I7 to H1

w(8,1) = AA19 -> connecting I8 to H1

w(9,1) = AA20 -> connecting I9 to H1

w(10,1) = AA21 -> connecting I10 to H1

w(1,2) = AA22 -> connecting I1 to H2

w(2,2) = AA23 -> connecting I2 to H2

w(3,2) = AA24 -> connecting I3 to H2

w(4,2) = AA25 -> connecting I4 to H2

w(5,2) = AA26 -> connecting I5 to H2

w(6,2) = AA27 -> connecting I6 to H2

w(7,2) = AA28 -> connecting I7 to H2

w(8,2) = AA29 -> connecting I8 to H2

w(9,2) = AA30 -> connecting I9 to H2

w(10,2) = AA31 -> connecting I10 to H2

" and

" so

" on

w(1,6) = AA62 -> connecting I1 to H6

w(2, 6) = AA63 -> connecting I2 to H6

w(3, 6) = AA64 -> connecting I3 to H6

w(4, 6) = AA65 -> connecting I4 to H6

w(5, 6) = AA66 -> connecting I5 to H6

w(6, 6) = AA67 -> connecting I6 to H6

w(7, 6) = AA68 -> connecting I7 to H6

w(8, 6) = AA69 -> connecting I8 to H6

w(9, 6) = AA70 -> connecting I9 to H6

w(10, 6) = AA71 -> connecting I10 to H6

From Hidden Layer to Output Layer

w(h1,1) = AA72 -> connecting H1 to O1

w(h2, 1) = AA73 -> connecting H2 to O1

w(h3, 1) = AA74 -> connecting H3 to O1

w(h4, 1) = AA75 -> connecting H4 to O1

w(h5, 1) = AA76 -> connecting H5 to O1

w(h6, 1) = AA77 -> 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.

p/g 15 is not available for viewing

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/nn_Solve.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

p/g 16 is not available for viewing

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/nn_Solve.htm for more information on the book

x

x

x

x

x

x

x

x

x

x

x

x

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*

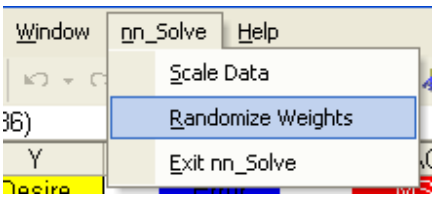


Figure 1.10

Enter AA12:AA77 and click on the *Randomize Weights* button. AA12:AA77 will be filled out with values between -1 to 1. (see Figure 1.11 below)

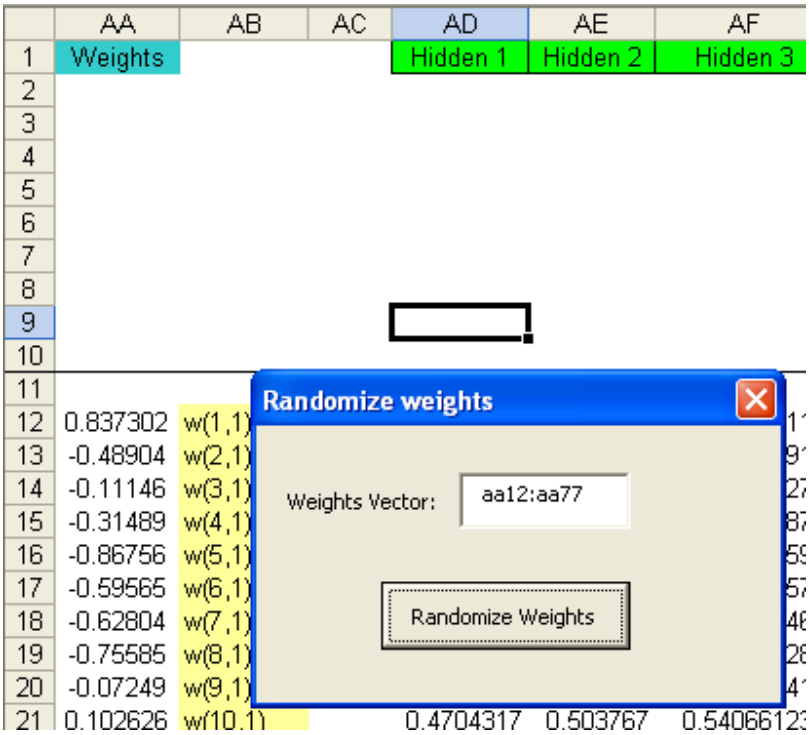


Figure 1.11

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

To use the Solver, click on the Tools heading on the menu bar and select the Solver . . . item.
(see Figure 1.12)

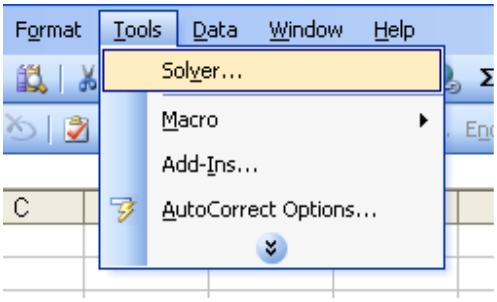


Figure 1.12

.

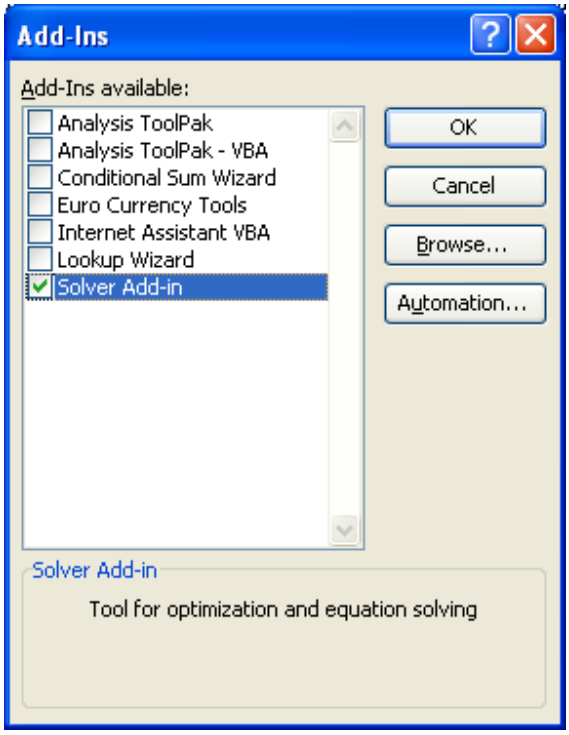


Figure 1.14

If Solver is not listed (see Figure 1.12) 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 1.14 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.

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

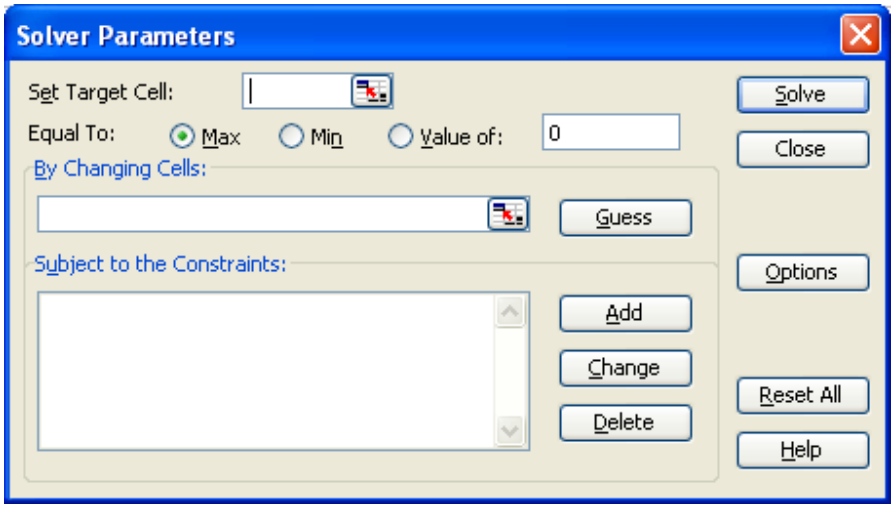


Figure 1.15

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 1.16 below

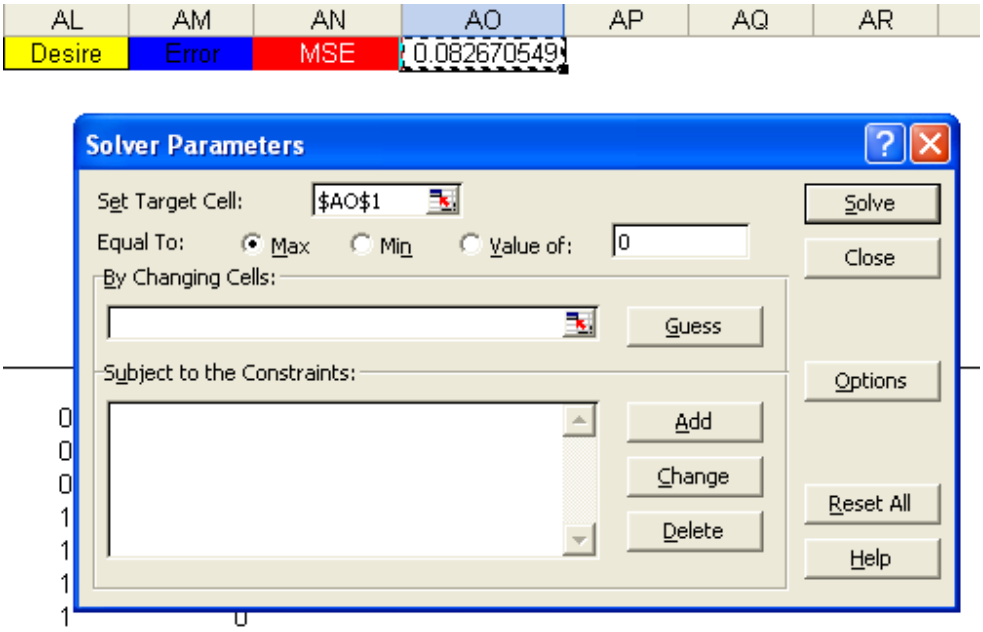


Figure 1.16

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.

p/g 21 is not available for viewing
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/nn_Solve.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

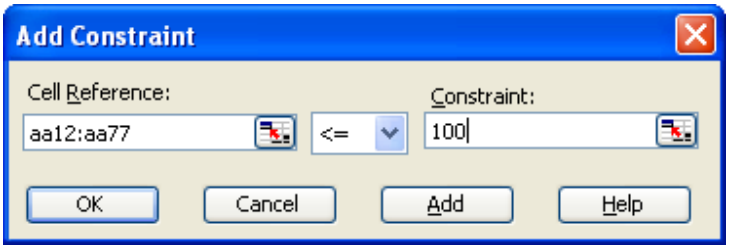


Figure 1.18

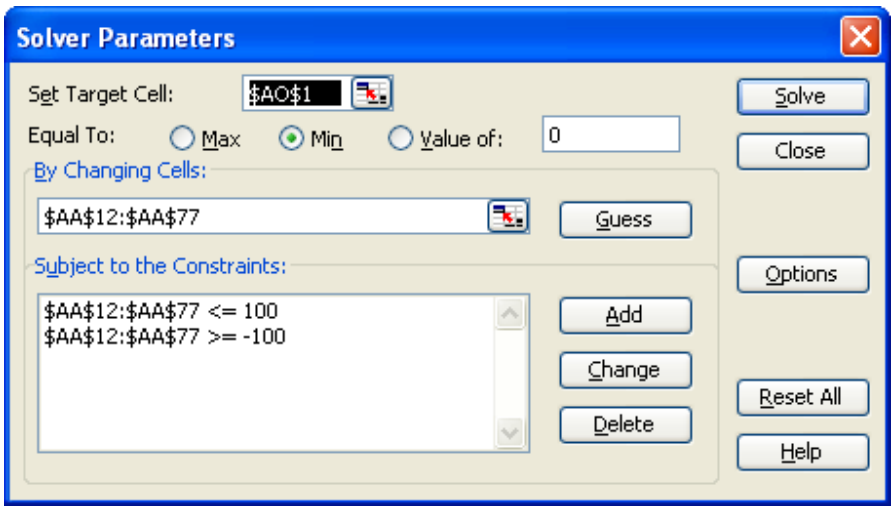


Figure 1.19

After that select the Options. This will allows you to adjust the way in which Solver approaches the solution.. (see Figure 1.20)

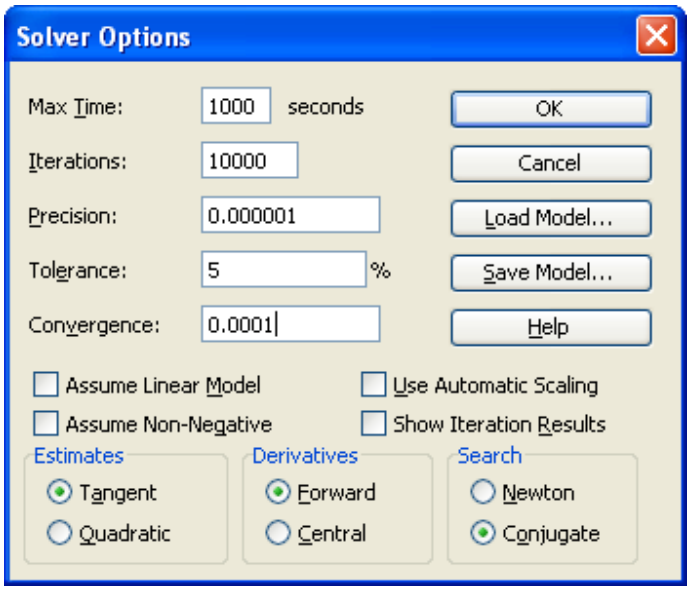


Figure 1.20

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.

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.

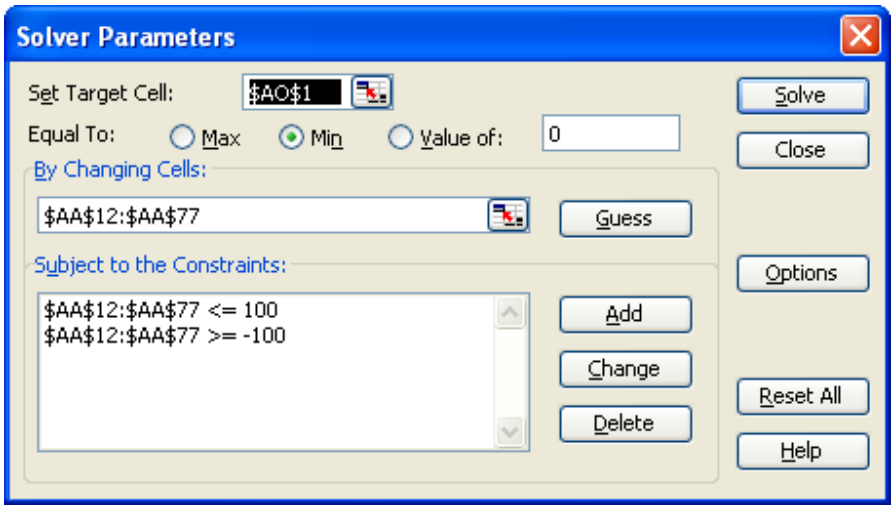


Figure 1.21

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.

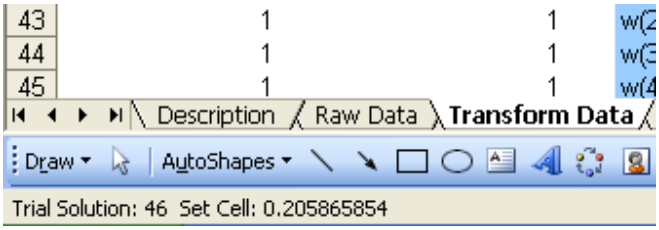


Figure 1.22

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

Figure 1.22 above.

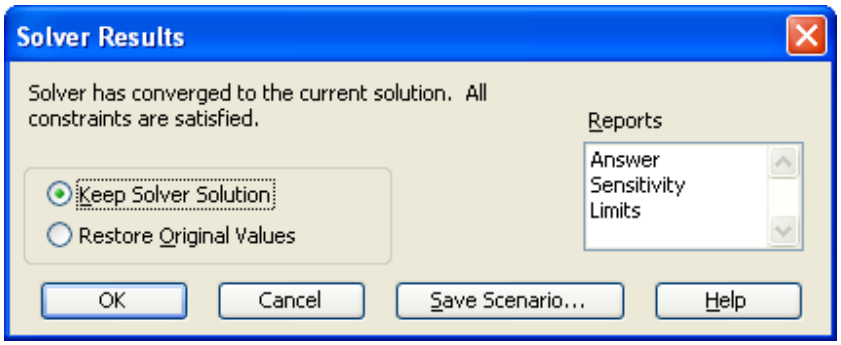


Figure 1.23

A message will appears after Solver has converged (see Figure 1.23)..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 1.24 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 1.24

Fill down the formula to row 410 to 411. (see Figure 1.25)

	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 1.25

On the Output cell i.e. AK410, the predicted value is 1 and AK411 is also 1 (see Figure 1.26 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 is. 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 1.26

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.
Open the file *Sales_Forecasting.xls*. 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)* and bring up *worksheet (Raw Data)*. There are 7 input factors and 2 desire 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 value 1 to 0. 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 2.1) 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 2.1

If you select the *worksheet(Transform Data),* it contains exactly what has been transform from *worksheet(Raw Data).* (see Figure 2.2)

	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 2.2

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 2.3)

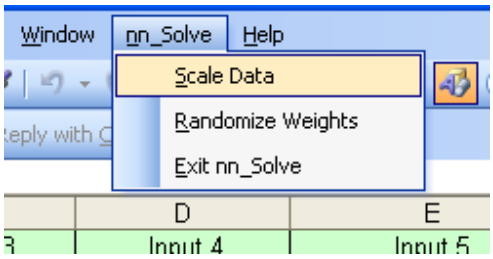


Figure 2.3

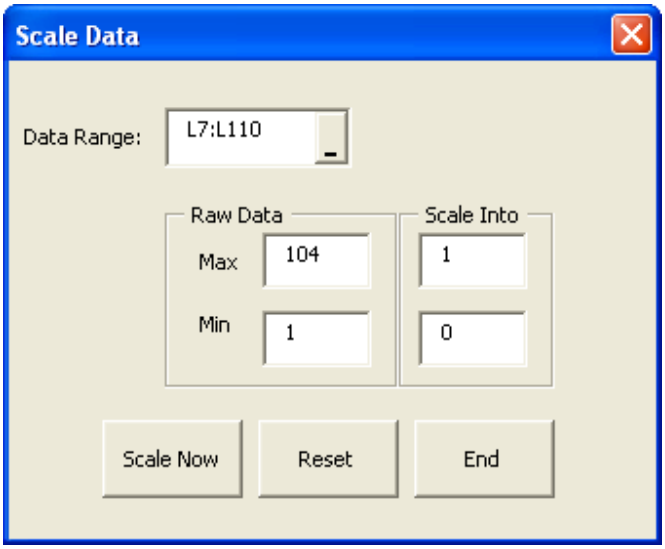


Figure 2.4

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 2.4)

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 2.5 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 2.5

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 2.6 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		Sales	Sales
3	Low = 0.25	Low = 0.25	Max = 10	Max = 5	Max =49800	Max = 27		(WeeklySales)	(Daily Sales)
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	

Figure 2.6

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:

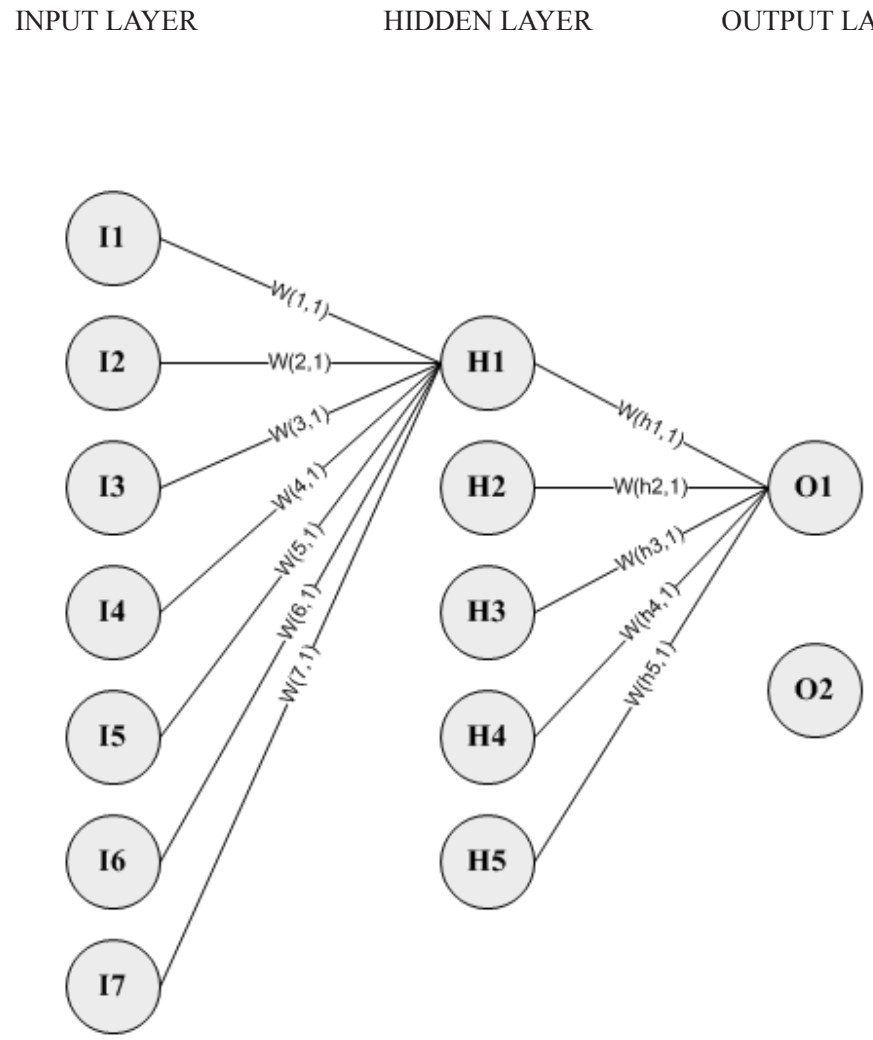


Figure 2.7

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 1.1

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 2.7 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 2.7 we have 7 inputs node and 5 hidden nodes and 2 output nodes. Here the number of weights are (7 x 5) + (5 x 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) = W7 -> connecting I1 to H1

w(2,1) = W8 -> connecting I2 to H1

w(3,1) = W9 -> connecting I3 to H1

w(4,1) = W10 -> connecting I4 to H1

w(5,1) = W11 -> connecting I5 to H1

w(6,1) = W12 -> connecting I6 to H1

w(7,1) = W13 -> connecting I7 to H1

w(1,2) = W14 -> connecting I1 to H2

w(2,2) = W15 -> connecting I2 to H2

w(3,2) = W16 -> connecting I3 to H2

w(4,2) = W17-> connecting I4 to H2

w(5,2) = W18 -> connecting I5 to H2

w(6,2) = W19 -> connecting I6 to H2

w(7,2) = W20 -> connecting I7 to H2

" and

" so

" on

"

w(1,5) = W35 -> connecting I1 to H5

w(2, 5) = W36 -> connecting I2 to H5

w(3, 5) = W37 -> connecting I3 to H5

w(4, 5) = W38 -> connecting I4 to H5

w(5, 5) = W39 -> connecting I5 to H5

w(6, 5) = W40 -> connecting I6 to H5

w(7, 5) = W41 -> connecting I7 to H5

From Hidden Layer to Output Layer

w(h1,1) = W42 -> 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) = W45 -> connecting H4 to O1

w(h5, 1) = W46 -> connecting H5 to O1

w(h1,2) = W47 -> 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) = W50 -> connecting H4 to O2

w(h5, 2) = W51 -> 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

p/g 34 is not available for viewing

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/nn_Solve.htm for more information on the book

x

x

x

x

x

x

x

x

x

x

x

x

x

x

p/g 35 is not available for viewing

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/nn_Solve.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

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 2.11 below)

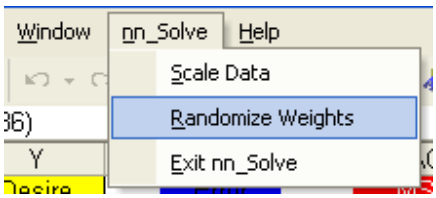


Figure 2.11

Enter W7:W51 and click on the *Randomize Weights* button. W7:W51 will be filled out with values between -1 to 1. (see Figure 2.12 below)

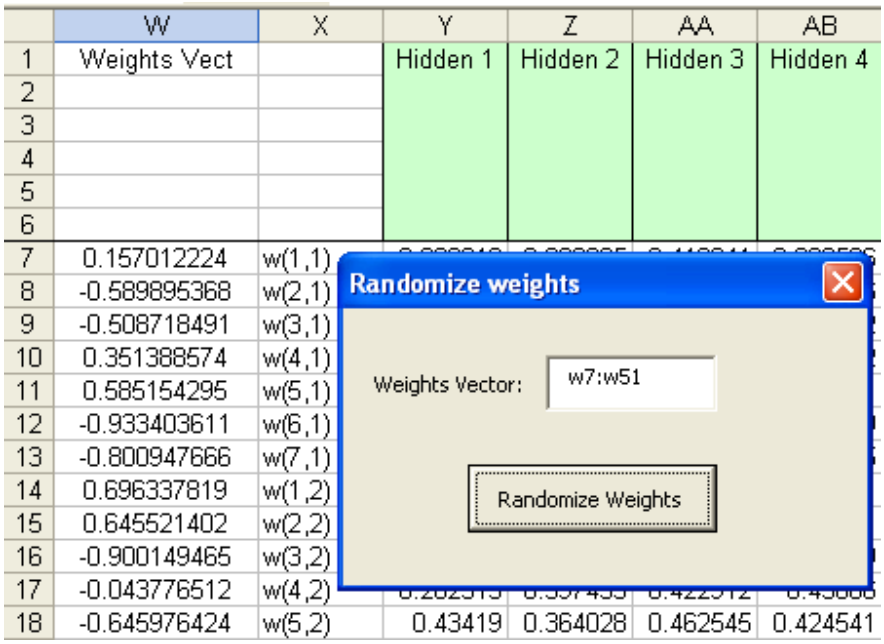


Figure 2.12

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 use the Solver, click on the Tools heading on the menu bar and select the Solver . . . item.
(see Figure 2.13)

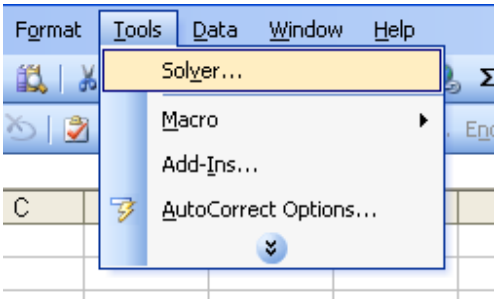


Figure 2.13

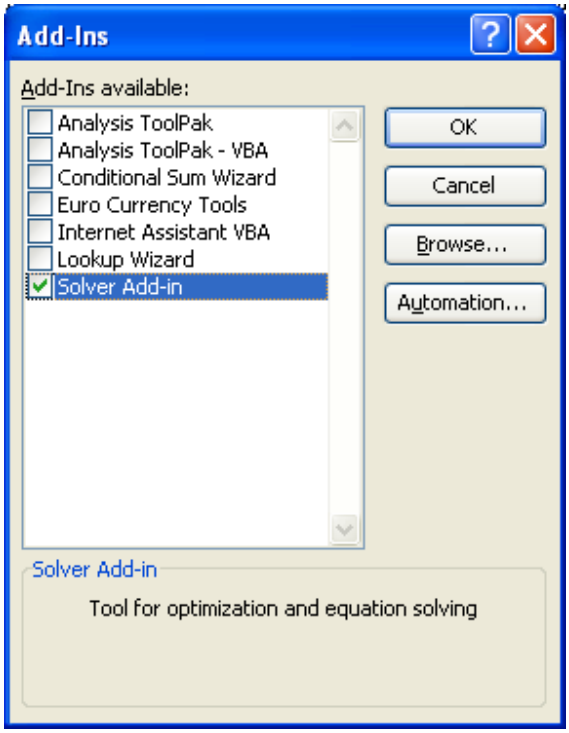


Figure 2.15

If Solver is not listed 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 by Figure 2.15 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.

It is imperative that you successfully load and install the Solver add-in because without it, neither Solver nor the Dummy Dependent Variable Analysis add-in will be available.

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

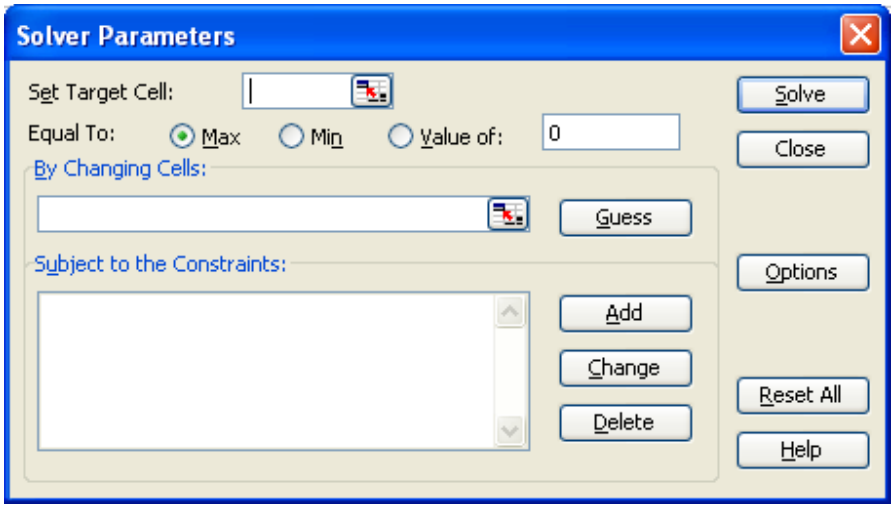


Figure 2.16

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 2.17 below

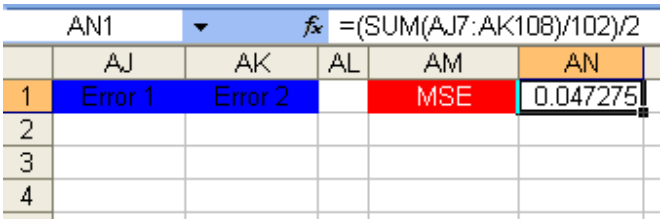


Figure 2.17

p/g 40 is not available for viewing

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/nn_Solve.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
p/g 41 is not available for viewing
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/nn_Solve.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
p/g 42 is not available for viewing
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/nn_Solve.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

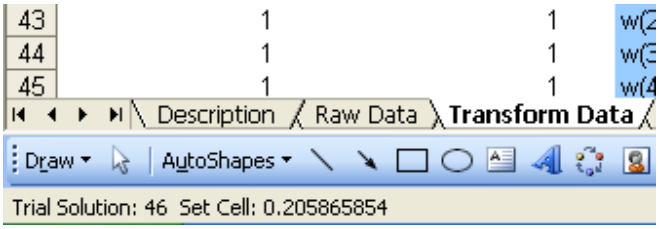


Figure 2.23

When Solver start optimizing, you will see the *Trial Solution* at the bottom left of your spreadsheet. See Figure 2.23 above.

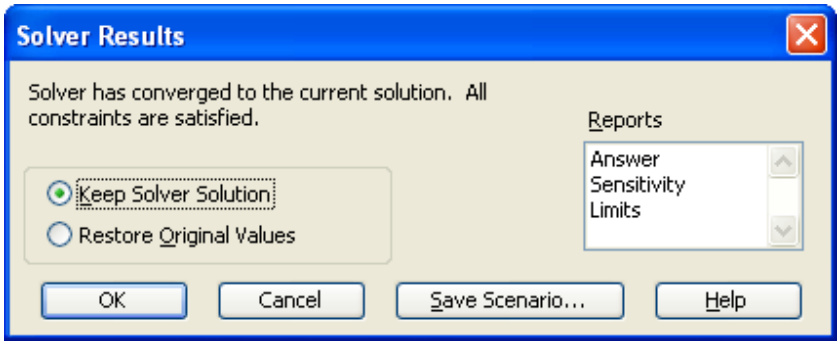


Figure 2.24

A message will appears after Solver has converged (see Figure 2.24). 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 2.25

Then goto Y108. Select Y108:AF108. (see Figue 2.26 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 Sales	Desire 2 Sales
2											
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 2.26

After that, you fill down until row 110. (see Figure 2.27)

	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 Sales	Desire 2 Sales
2											
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 2.27

	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 2.28

So, for row 109 we have 0.903441 for predicted Output 1 (AE109) and 0.102692 for predicted Output 2 (AF109). (see Figure 2.28).

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 2.29 below)

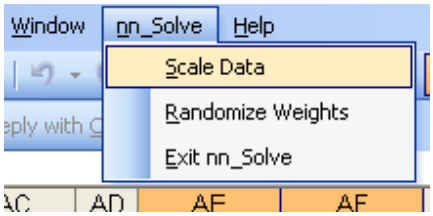


Figure 2.29

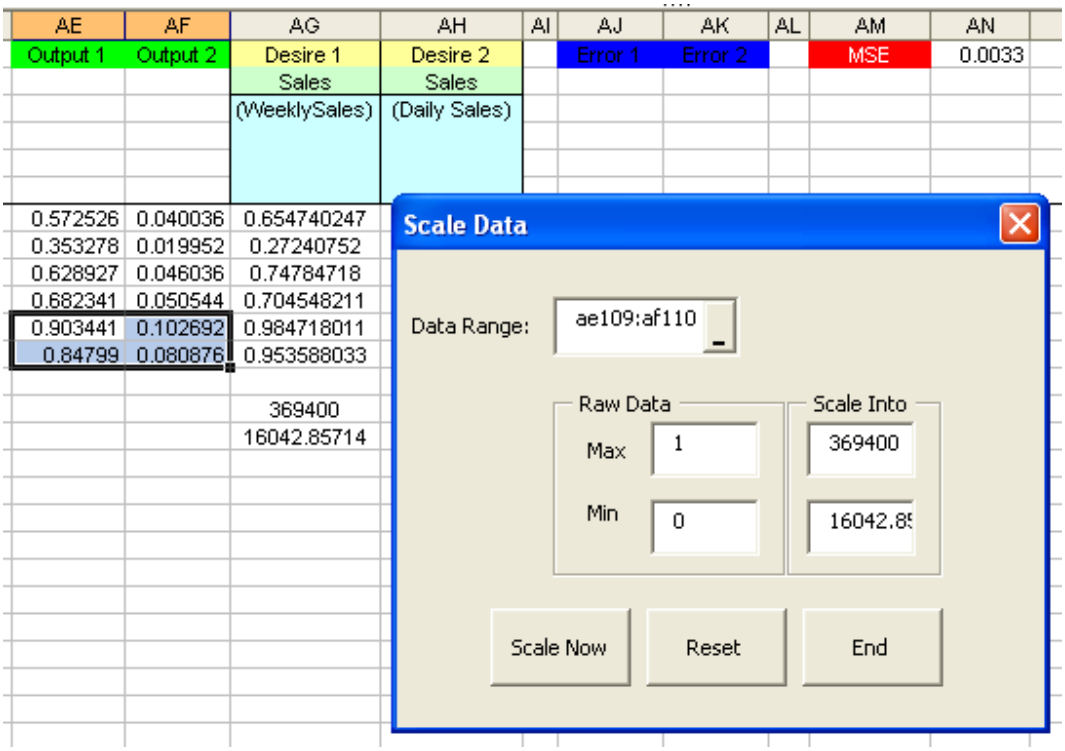


Figure 2.30

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 2.30).

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 Sales	Desire 2 Sales
2											
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 2.31

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 2.31 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.

Open the file Dow_Weekly.xls. In this example, I have build 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)* 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 3.1a)

	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 3.1a

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 3.1)

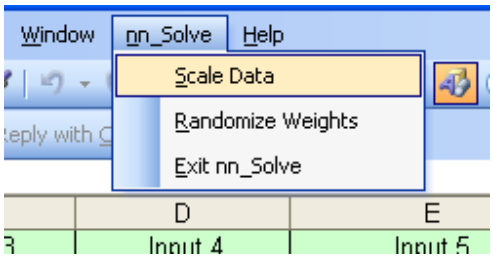


Figure 3.1

Thus we need to convert the data in the range J3:O289 (see Figure 3.2). 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 enter the value 0 as the minimum as it represent nothing. (see Figure 3.2 below)

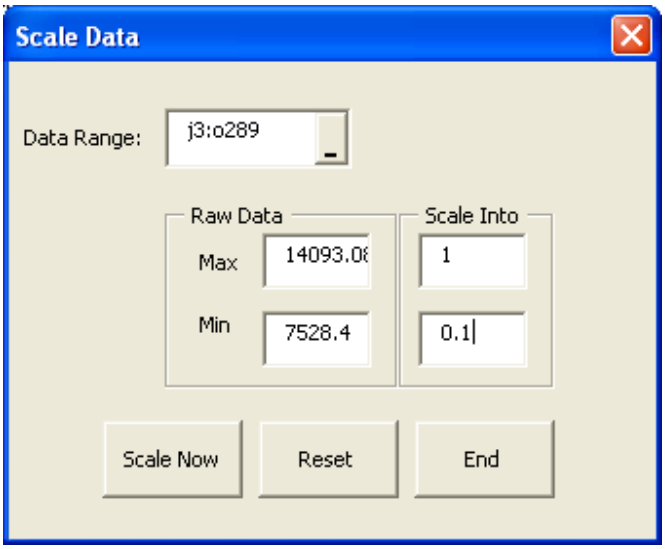


Figure 3.2

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 3.3 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 3.3

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:

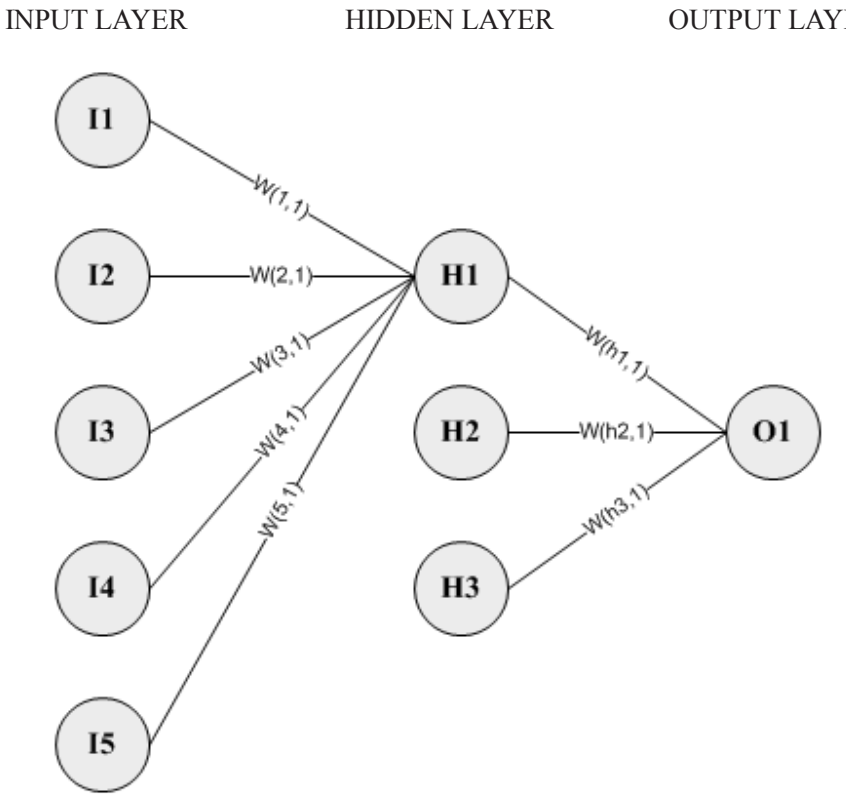


Figure 3.4

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 1.1

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 3.4 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 x 3) + (3 x 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) = Q3 -> connecting I1 to H1

w(2,1) = Q4 -> connecting I2 to H1

w(3,1) = Q5 -> connecting I3 to H1

w(4,1) = Q6 -> connecting I4 to H1

w(5,1) = Q7 -> connecting I5 to H1

w(1,2) = Q8 -> connecting I1 to H2

w(2,2) = Q9 -> connecting I2 to H2

w(3,2) = Q10 -> connecting I3 to H2

w(4,2) = Q11-> connecting I4 to H2

w(5,2) = Q12 -> connecting I5 to H2

“ and

“ so

“ on

“

w(1,5) = Q13 -> connecting I1 to H5

w(2, 5) = Q14 -> connecting I2 to H5

w(3, 5) = Q15 -> connecting I3 to H5

w(4, 5) = Q16 -> connecting I4 to H5

w(5, 5) = Q17 -> connecting I5 to H5

From Hidden Layer to Output Layer

w(h1,1) = Q18 -> connecting H1 to O1

w(h2, 1) = $ Q$19 -> connecting H2 to O1

w(h3, 1) = Q20 -> 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

p/g 52 is not available for viewing

x

x

x

x

x

x

x

x

x

This part of the book is not available for viewing

x

x

x

x

x

x

x

x

x

x

x

p/g 53 is not available for viewing

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/nn_Solve.htm for more information on the book

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

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 = (SUM(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 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 Q3:Q20. From the **nn_Solve** menu, select *Randomize Weights* (see Figure 3.6)

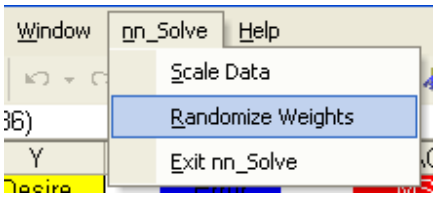
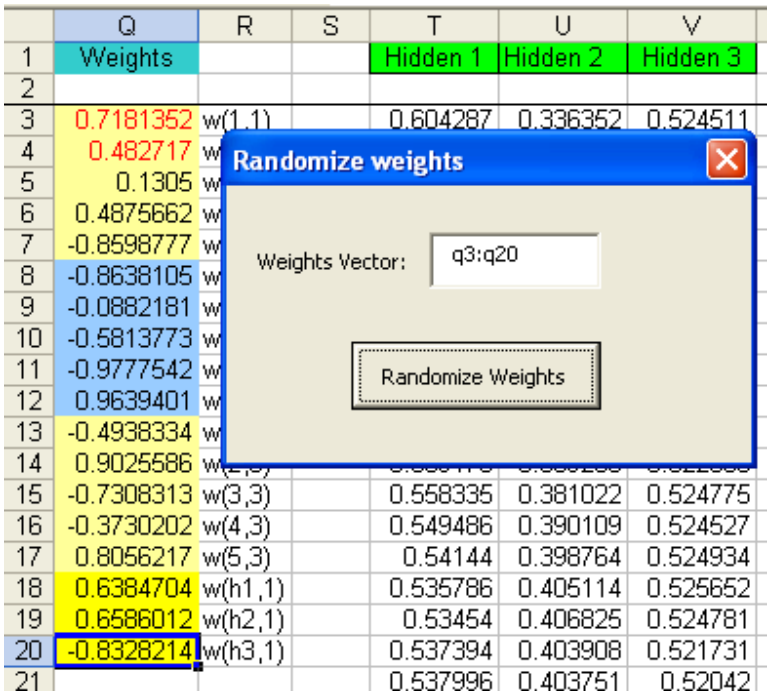


Figure 3.6

Enter Q3:Q20 and click on the *Randomize Weights* button. Q3:Q20 will be filled out with values between -1 to 1. (see Figure 3.7 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 3.7

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 use the Solver, click on the Tools heading on the menu bar and select the Solver . . . item. (see Figure 3.8)

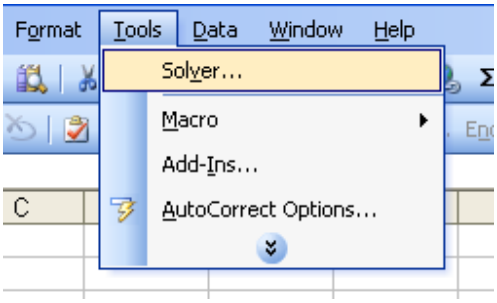


Figure 3.8

.

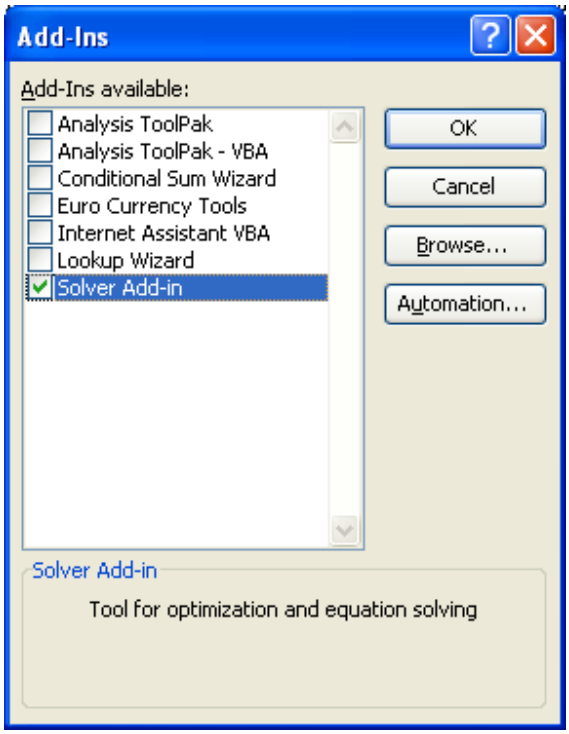


Figure 3.10

If Solver is not listed (see Figure 3.8 above), 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 by the Figure 3.10 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.

When you click on the Tools menu, it should be listed somewhere as shown above on the right.

If the Solver add-in is not listed in the Add-Ins dialog box, click on the Select or Browse button and navigate to the Solver add-in (called solver.xla in Windows and Solver on the MacOS) and open it. It should be in the Library directory in the folders where Microsoft Office is installed.

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.

.

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

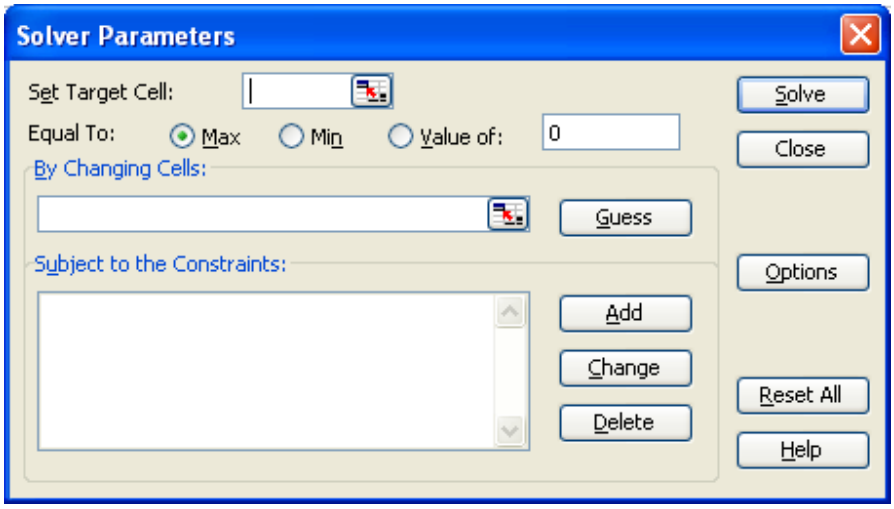
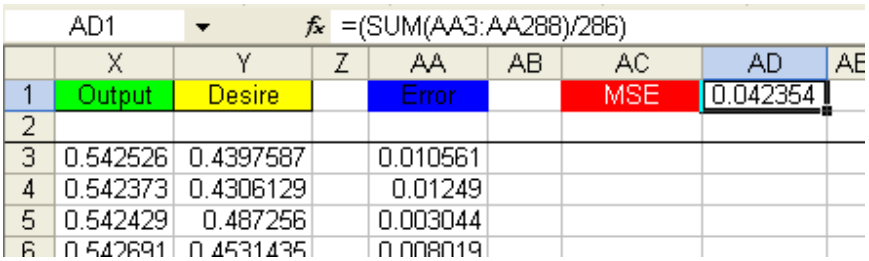


Figure 3.11

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 3.12 and 3.13 below



	X	Y	Z	AA	AB	AC	AD	AE
1	Output	Desire		Error		MSE	0.042354	
2								
3	0.542526	0.4397587		0.010561				
4	0.542373	0.4306129		0.01249				
5	0.542429	0.487256		0.003044				
6	0.542691	0.4531435		0.008019				

Figure 3.12

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

p/g 59 is not available for viewing
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/nn_Solve.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

p/g 60 is not available for viewing
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/nn_Solve.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

Select **Conjugate** as the *Search* method. This prove 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.

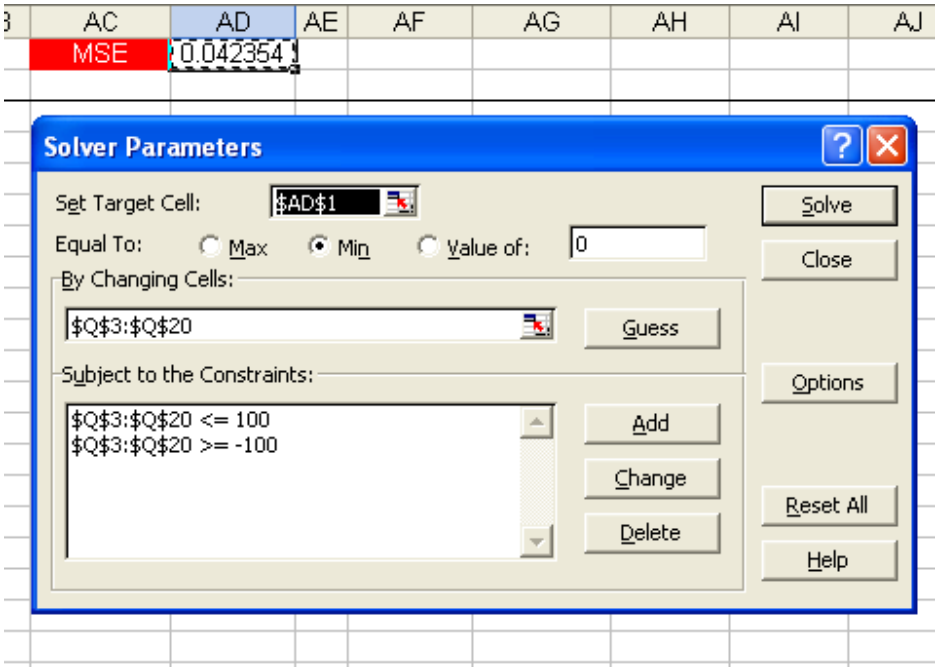


Figure 3.17

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.

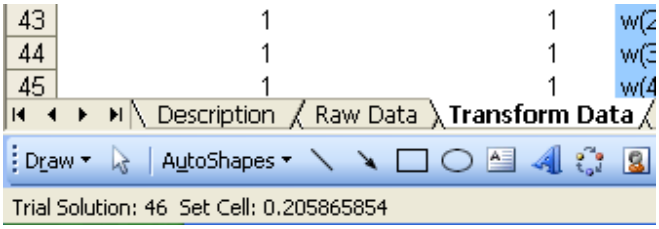


Figure 3.19

When Solver start optimizing, you will see the *Trial Solution* at the bottom left of your spreadsheet. See Figure 3.19 above.

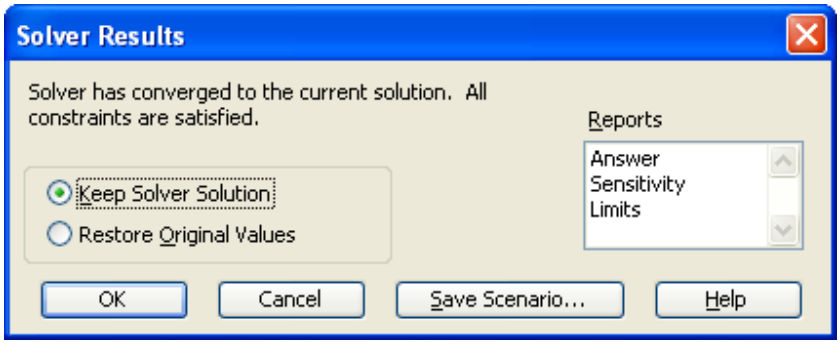


Figure 3.20

A message appears when Solver converged (see Figure 3.20). 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 3.21

Then goto T288. Select T288:X288. See Figure 3.22 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 3.22

After that, you fill down until row 289 (see Figure 3.23 below)

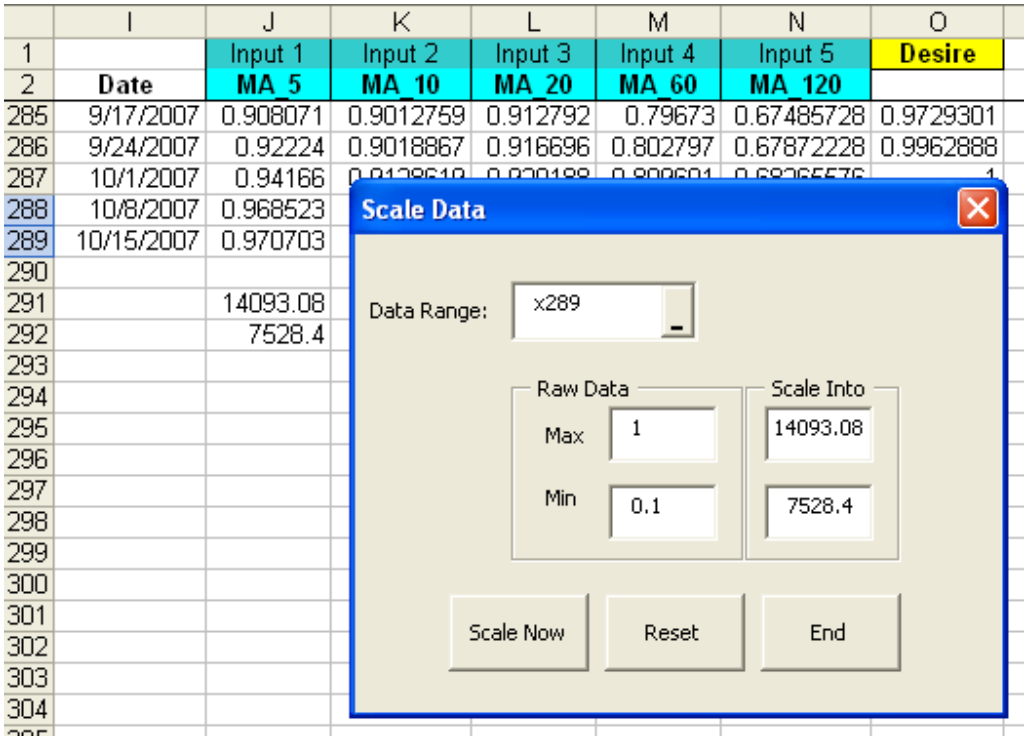
	S	T	U	V	W	X	Y
1		Hidden 1	Hidden 2	Hidden 3		Output	Desire
2							
285		0.992519	5.04E-09	0.790843		0.922078	0.9729301
286		0.99295	4.36E-09	0.79402		0.928817	0.9962888
287		0.993345	3.52E-09	0.797625		0.93595	1
288		0.993777	2.71E-09	0.80205		0.943863	0.9217092
289		0.993988	2.55E-09	0.803408		0.946003	
290							
291							

Figure 3.23

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.



	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	
285	9/17/2007	0.908071	0.9012759	0.912792	0.79673	0.67485728	0.9729301
286	9/24/2007	0.92224	0.9018867	0.916696	0.802797	0.67872228	0.9962888
287	10/1/2007	0.94166					1
288	10/8/2007	0.968523					
289	10/15/2007	0.970703					
290							
291		14093.08					
292		7528.4					
293							
294							
295							
296							
297							
298							
299							
300							
301							
302							
303							
304							

Figure 3.24

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 3.24)

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 3.25

So our predicted DJIA weekly price is 13860.05 as in X289 (see Figure 3.25). 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

Open the file *Real_Estate.xls*. 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)* 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

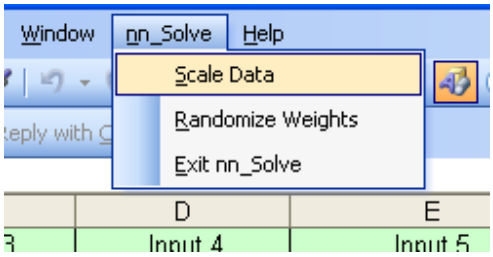


Figure 4.1

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 4.2)

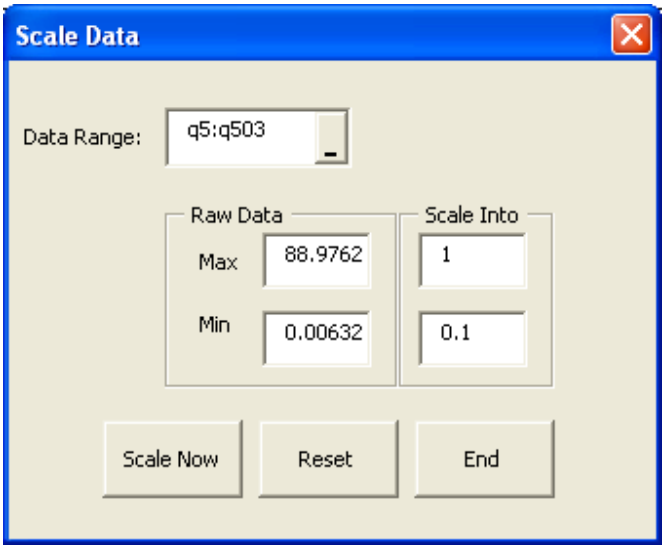
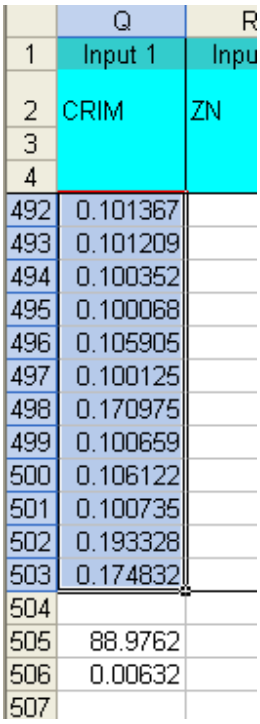


Figure 4.2

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. (see Figure 3. below)

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 4.3 below)



	Q	R
1	Input 1	Inpu
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 4.3

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 4.4 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 4.4

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:

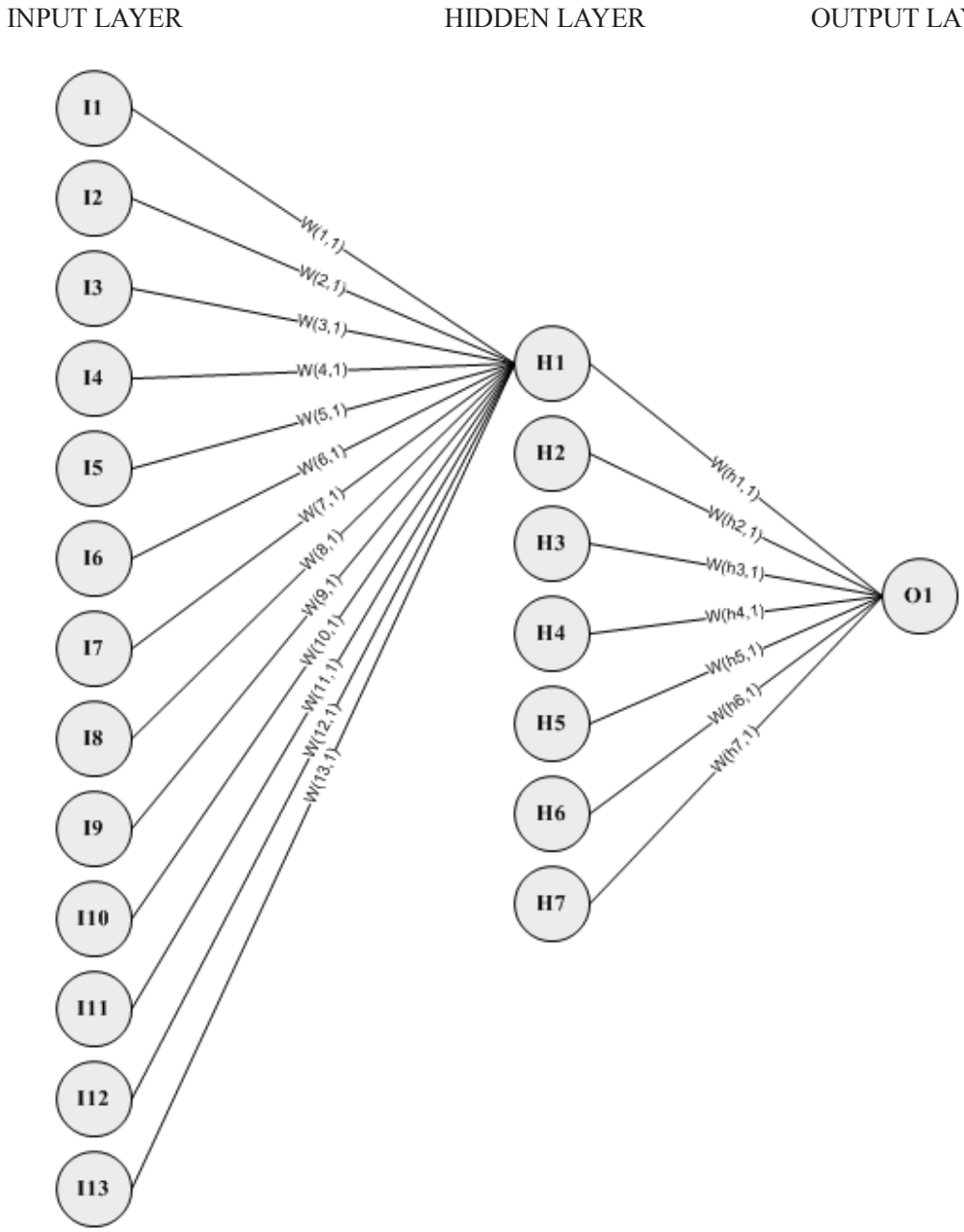


Figure 4.5

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 1.1

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 4.5 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:

i. The output of a neuron in a layer goes to all neurons in the following layer. See Figure 4.5

ii. we have 13 inputs node and 7 hidden nodes and 1 output node. Here the number of weights are (13 x 7) + (7 x 1) = 98

iii. Each neuron has its own input weights.

iv. 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) = AG5 -> connecting I1 to H1

w(2,1) = AG6 -> connecting I2 to H1

w(3,1) = AG7 -> connecting I3 to H1

w(4,1) = AG8 -> connecting I4 to H1

w(5,1) = AG9 -> connecting I5 to H1

w(6,1) = AG10 -> connecting I6 to H1

w(7,1) = AG11 -> connecting I7 to H1
w(8,1) = AG12 -> connecting I8 to H1
w(9,1) = AG13 -> connecting I9 to H1
w(10,1) = AG14 -> connecting I10 to H1
w(11,1) = AG15 -> connecting I11 to H1
w(12,1) = AG16 -> connecting I12 to H1
w(13,1) = AG17 -> connecting I13 to H1

w(1,2) = AG18 -> connecting I1 to H2
w(2,2) = AG19 -> connecting I2 to H2
w(3,2) = AG20 -> connecting I3 to H2
w(4,2) = AG21-> connecting I4 to H2
w(5,2) = AG22 -> connecting I5 to H2
w(6,2) = AG23 -> connecting I6 to H2
w(7,2) = AG24 -> connecting I7 to H2
w(8,2) = AG25 -> connecting I8 to H2
w(9,2) = AG26 -> connecting I9 to H2
w(10,2) = AG27 -> connecting I10 to H2
w(11,2) = AG28-> connecting I11 to H2
w(12,2) = AG29 -> connecting I12 to H2
w(13,2) = AG30 -> connecting I13 to H2

“

“ and

“ so

“ on

w(1,7) = AG83 -> connecting I1 to H7
w(2,7) = AG84 -> connecting I2 to H7
w(3,7) = AG85 -> connecting I3 to H7
w(4,7) = AG86-> connecting I4 to H7
w(5,7) = AG87 -> connecting I5 to H7
w(6,7) = AG88 -> connecting I6 to H7
w(7,7) = AG89 -> connecting I7 to H7
w(8,7) = AG90 -> connecting I8 to H7
w(9,7) = AG91 -> connecting I9 to H7

w(10,7) = AG92 -> connecting I10 to H7

w(11,7) = AG93-> connecting I11 to H7

w(12,7) = AG94 -> connecting I12 to H7

w(13,7) = AG95 -> connecting I13 to H7

From Hidden Layer to Output Layer

w(h1,1) = AG96 -> connecting H1 to O1

w(h2, 1) = AG97 -> connecting H2 to O1

w(h3, 1) = AG98 -> connecting H3 to O1

w(h4, 1) = AG99 -> connecting H4 to O1

w(h5, 1) = AG100 -> connecting H5 to O1

w(h6, 1) = AG101 -> connecting H6 to O1

w(h7, 1) = AG102 -> 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.

p/g 71 is not available for viewing

x

x

x

x

x

x

x

x

x

x

p/g 72 is not available for viewing

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/nn_Solve.htm for more information on the book

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

Remember 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. For example in Figure 4.7 above indicate that the desire Output is 0.191111 i.e. AS5. The actual output is 0.536547 i.e. AR5. Since 0.536547 is not what we want, we minus this with the desire output. After that we square the error to get a positive value

Error = (AR5 – AS5) ^ 2 ; we get 0.119326 (in AT5)

The closer the actual output of the network matches the desired output, the better. This is only one pattern error. Since we have using 498 rows of patterns, we fill down the formula again from row 5 until row 502 in this spreadsheet. Remember, we save the last 1 rows for testing our model later. Sum up all the error and take the average. (see Figure 4.8)

MSE = (SUM(AT5:AT502)/498)

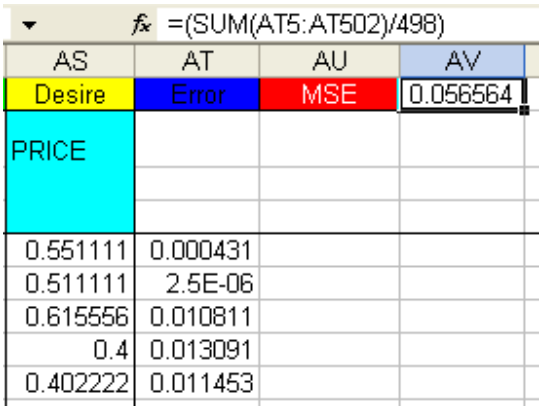


Figure 4.8

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 AG5:AG102. From the **nn_Solve** menu, select *Randomize Weights* (see Figure 4.9)

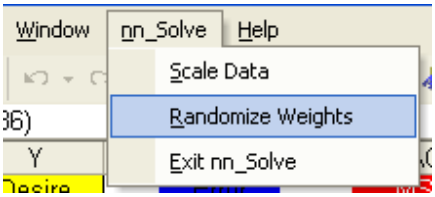


Figure 4.9

Enter AG5:AG102 and click on the *Randomize Weights* button. AG5:AG102 will be filled out with values between -1 to 1. (see Figure 4.10 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.

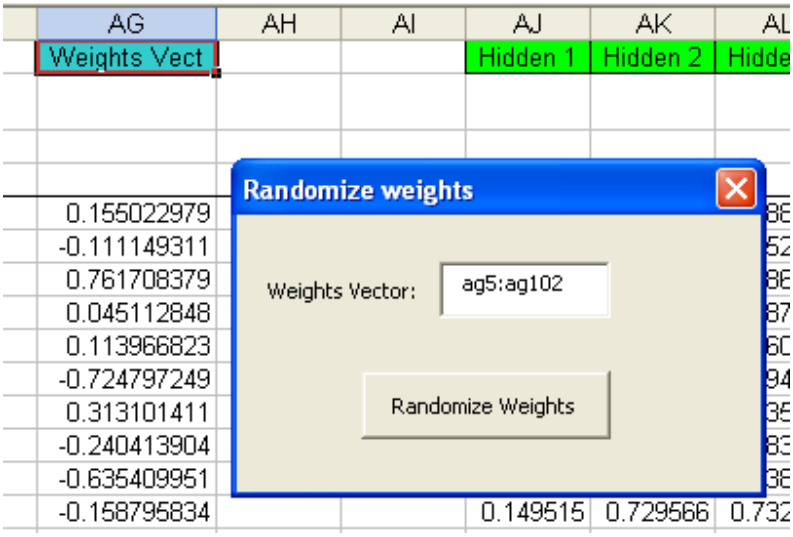


Figure 4.10

Accessing Excel's Solver

To use the Solver, click on the Tools heading on the menu bar and select the Solver . . . item.

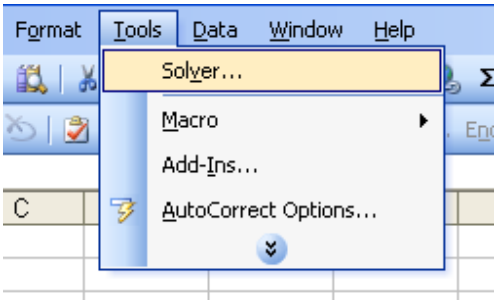


Figure 4.11

.

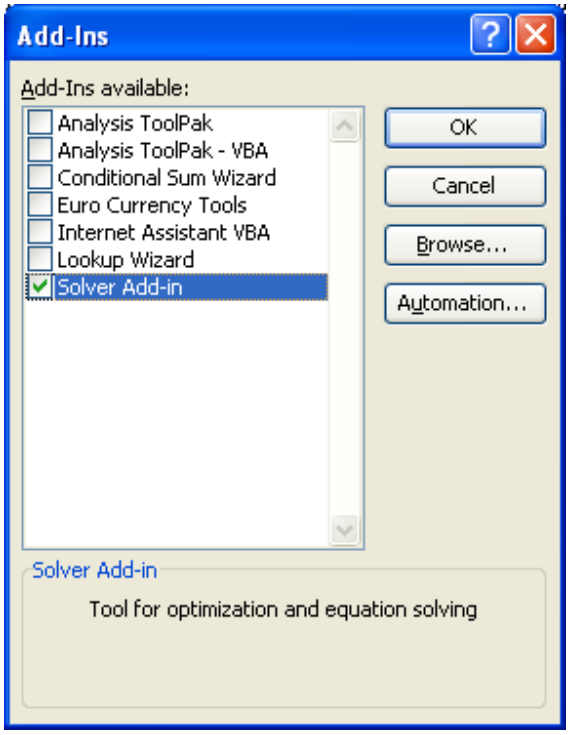


Figure 4.13

If Solver is not listed (like Figure 4.11), 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 by the Figure 4.13 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.

When you click on the Tools menu, it should be listed somewhere as shown above on the right.

If the Solver add-in is not listed in the Add-Ins dialog box, click on the Select or Browse button and navigate to the Solver add-in (called solver.xla in Windows and Solver on the MacOS) and open it. It should be in the Library directory in the folders where Microsoft Office is installed.

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.

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

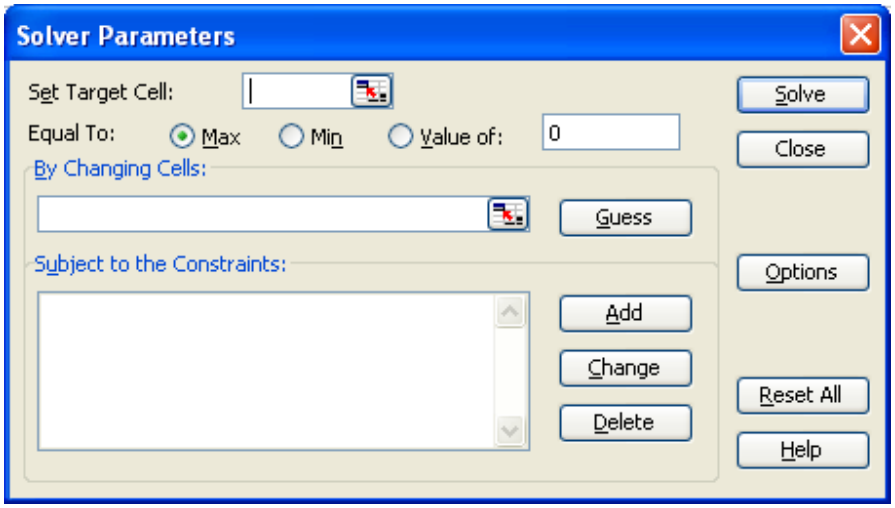


Figure 4.14

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 4.15 below

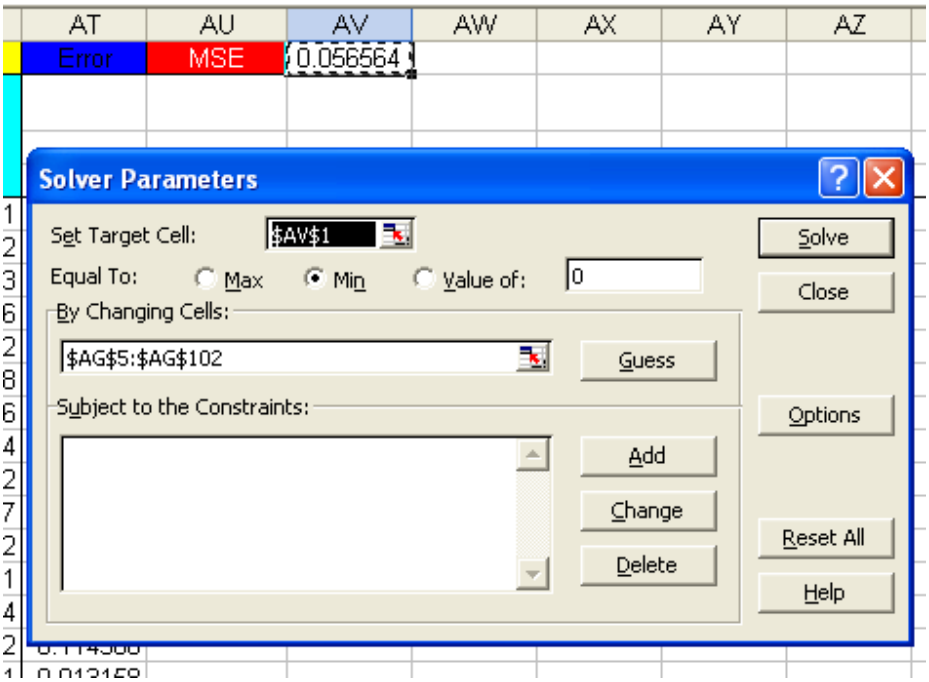


Figure 4.15

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

p/g 78 is not available for viewing
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/nn_Solve.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

p/g 79 is not available for viewing

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/nn_Solve.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
p/g 80 is not available for viewing
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/nn_Solve.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

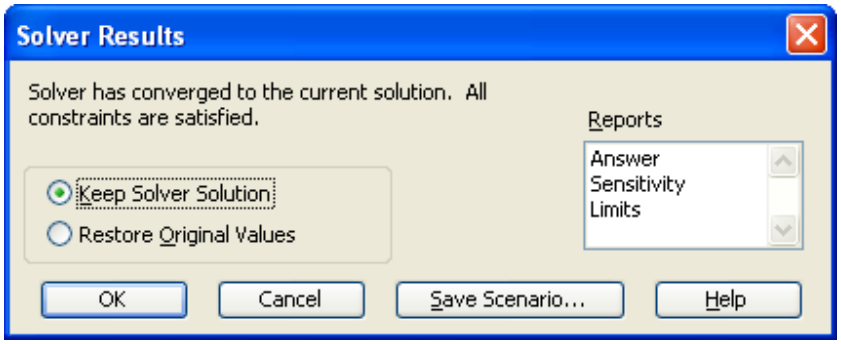


Figure 4.21

A message appears when Solver converged (see Figure 4.21). 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 4.22

Select AJ502:AR502. (See Figure 4.22 above)

After that, you fill down until row 503 (see Figure 4.23 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	

Figure 4.23

So, for row 503 we have 0.26748 for predicted Output 1 (AR503). (see Figure 4.23)

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.

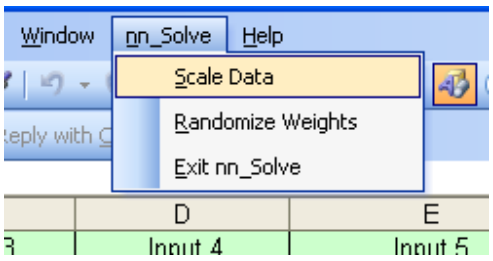


Figure 4.24

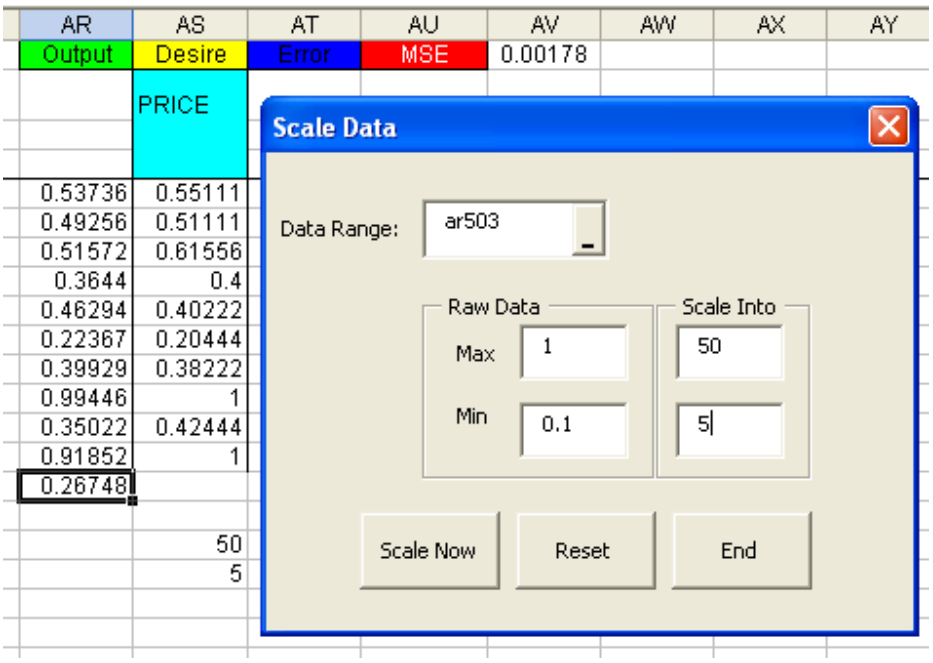


Figure 4.25

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 save 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 4.25 above). Click on *Scale Now.*

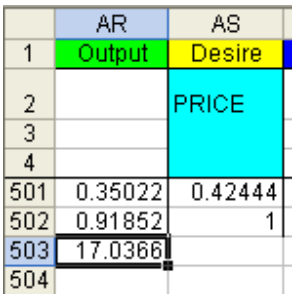


Figure 4.26

So our predicted real estate price is 17.0366 as in AR503 (see Figure 4.26 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)* 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 5.1)

	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 5.1

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 5.2)

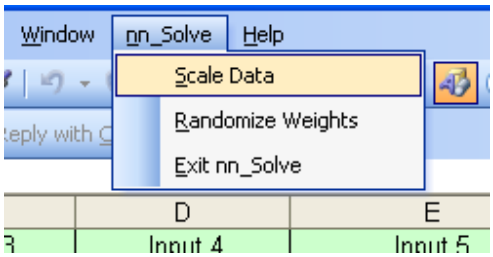


Figure 5.2

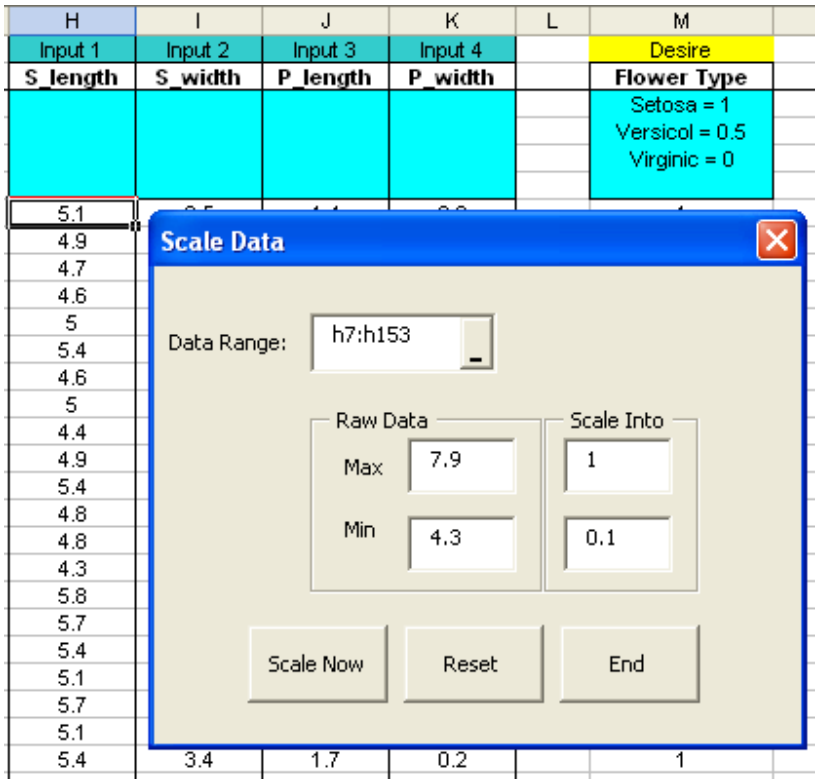


Figure 5.3

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 5.2)

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 5.4 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 5.4

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 5.5 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 5.5

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

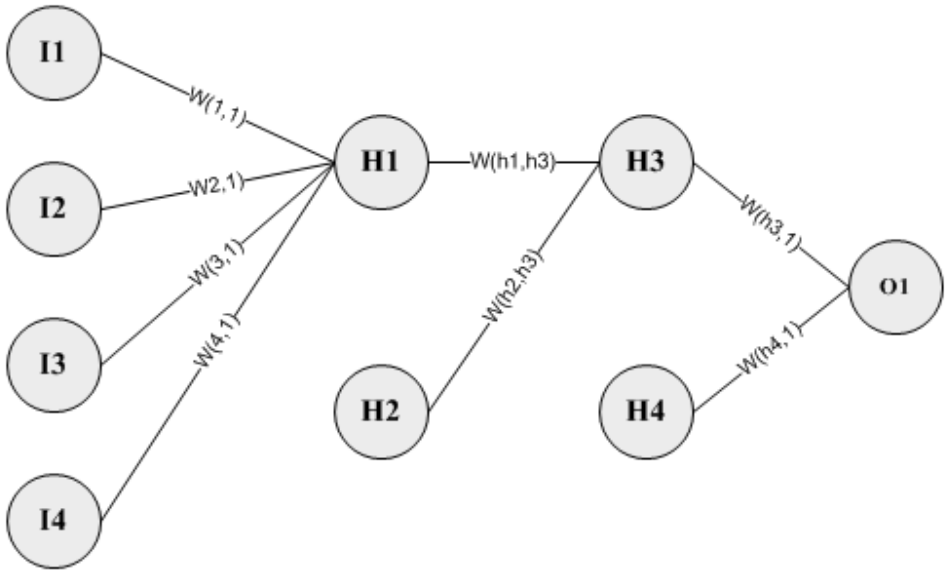


Figure 5.6

There are 4 (column H:K) nodes or neuron on the input layer. 2 neurons on the hidden layer 1 (columnR: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 1.1

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 1.2 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 5.6
- 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 x 2) + (2 x 2) + (2 x 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) = O7 -> connecting I1 to H1

w(2,1) = O8 -> connecting I2 to H1

w(3,1) = O9 -> connecting I3 to H1

w(4,1) = O10 -> connecting I4 to H1

w(1,2) = O11 -> connecting I1 to H2

w(2,2) = O12 -> connecting I2 to H2

w(3,2) = O13 -> connecting I3 to H2

w(4,2) = O14-> connecting I4 to H2

“ and

“ so

“ on

“

“

From Hidden Layer 1 to Hidden Layer 2

w(h1,h3) = O15 -> connecting H1 to H3

w(h2,h3) = O16 -> connecting H2 to H3

w(h1,h4) = O17 -> connecting H1 to H4

w(h2,h4) = O18-> connecting H2 to H4

From Hidden Layer 2 to Output Layer

w(h3,1) = O19 -> connecting H3 to O1

w(h4,1) = O20 -> 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.

p/g 89 is not available for viewing

x

x

x

x

x

x

x

x

x

x

x

x

p/g 90 is not available for viewing

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/nn_Solve.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

Remember 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. For example in Figure 5.7a above indicate that the desire Output is 1 i.e. Y7. The actual output is 0.394570671 i.e. X7. Since 0.394570671 is not what we want, we minus this with the desire output. After that we square the error to get a positive value

Error = (X7 – Y7) ^ 2 ; we get 0.366545 (in Z7)

The closer the actual output of the network matches the desired output, the better. This is only one pattern error. Since we have using 146 rows of patterns, we fill down the formula again from row 7 until row 152 in this spreadsheet. Remember, we save the last 1 rows for testing our model later. Sum up all the error and take the average. (see Figure 5.7a)

MSE = (SUM(Z7:Z152)/146)

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 O7:O20. From the **nn_Solve** menu, select *Randomize Weights* (see Figure 5.8)

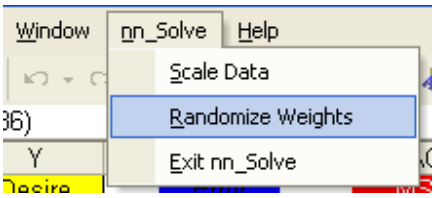
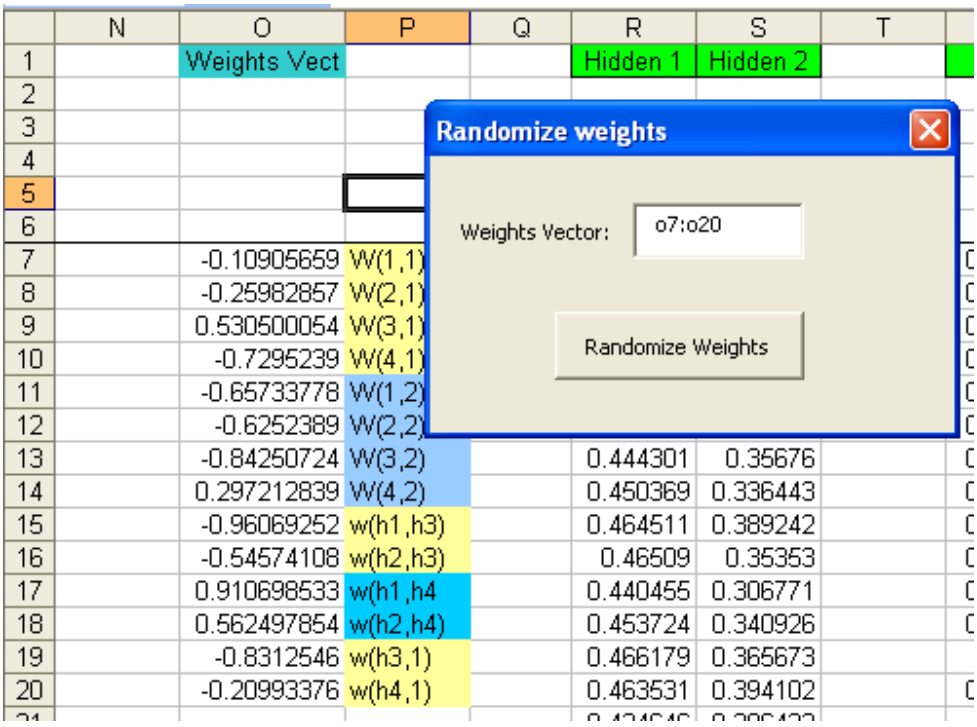


Figure 5.8

Enter O7:O20 and click on the *Randomize Weights* button. O7:O20 will be filled out with values between -1 to 1. (see Figure 5.9 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 5.9

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 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 use the Solver, click on the Tools heading on the menu bar and select the Solver . . . item.

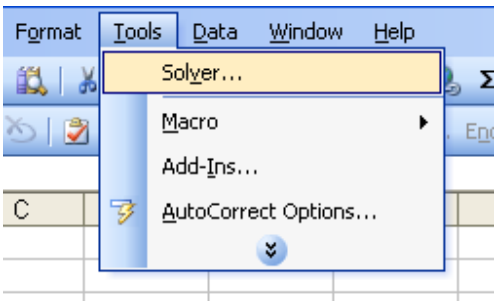


Figure 5.10

.

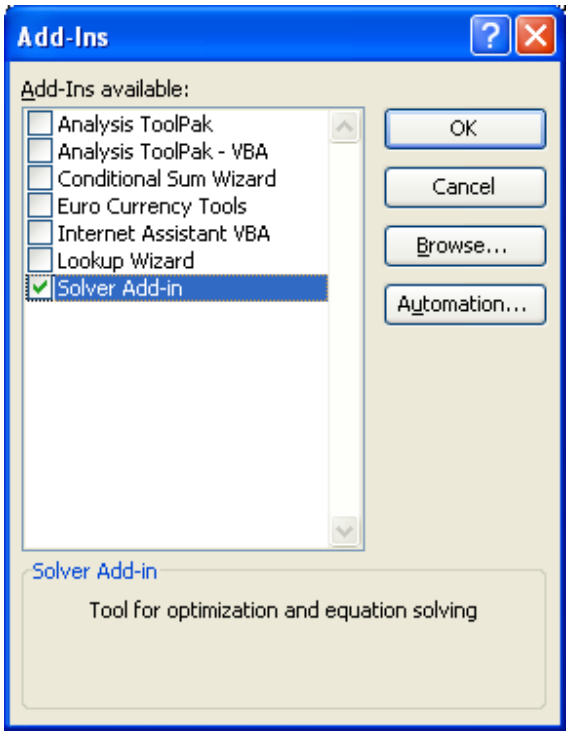


Figure 5.12

If Solver is not listed (like Figure 5.10), 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 by the Figure 5.12 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.

When you click on the Tools menu, it should be listed somewhere as shown above on the right.

If the Solver add-in is not listed in the Add-Ins dialog box, click on the Select or Browse button and navigate to the Solver add-in (called solver.xla in Windows and Solver on the MacOS) and open it. It should be in the Library directory in the folders where Microsoft Office is installed.

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.

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

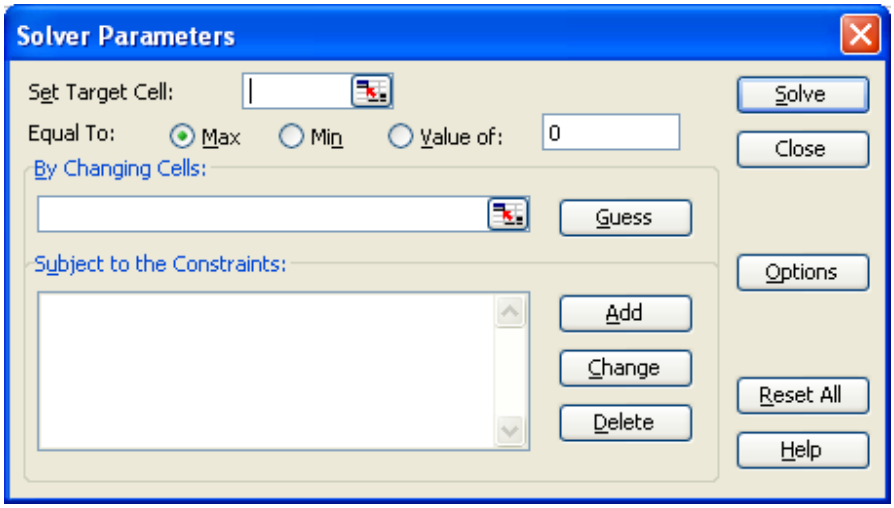


Figure 5.13

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 5.14 below

p/g 95 is not available for viewing

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/nn_Solve.htm for more information on the book

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

p/g 96 is not available for viewing

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/nn_Solve.htm for more information on the book

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

x

p/g 97 is not available for viewing

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

x

x

x

x

x

x

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.

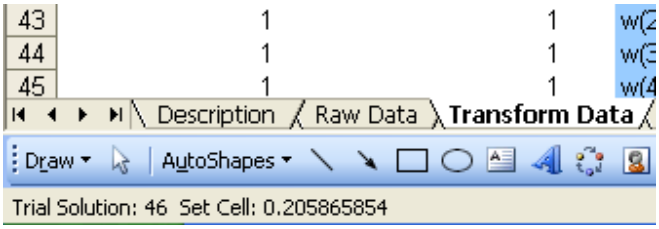


Figure 5.19

When Solver start optimizing, you will see the *Trial Solution* at the bottom left of your spreadsheet. See Figure 5.19 above.

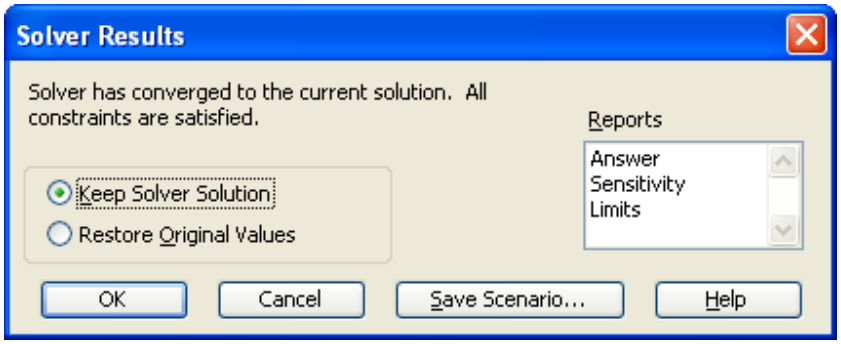


Figure 5.20

A message appears when Solver converged (see Figure 5.20). 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 5.21

Select R152:X152. (See Figure 5.21 above).After that, you fill down until row 153 (see Figure 5.22 below)

	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									
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 5.22

So, for row 153 we have 0.08839486 for predicted Output 1 (X153). (see Figure 5.23 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							FLOWER
3							Setosa = 1
4							Versicol = 0.5
5							Virginic = 0
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 5.23

So our predicted type of Irises price is Virginic which is represented by the value 0 as in X153 (see Figure 5.23 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 visualise 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.

This book comes with an Excel add in **nn_Solve** and 5 neural network models developed with Excel spreadsheets. The 5 workbooks that contain the neural network models are:

a. Credit_Approval

b. Sales_Forecasting

c. Dow_Weekly_Prices

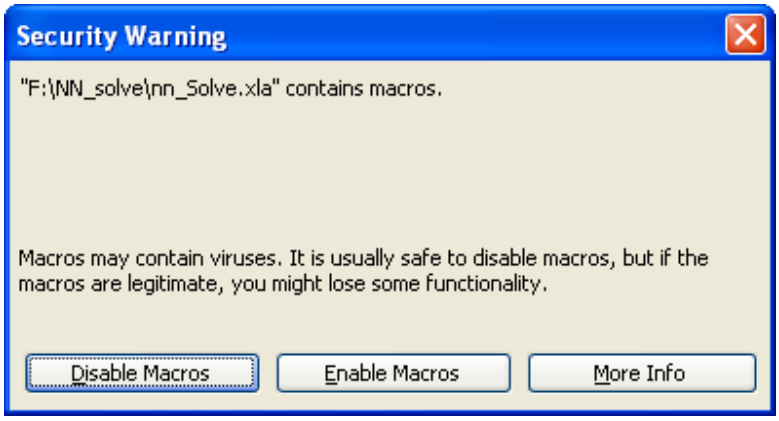
d. Real_Estate

e. Irises

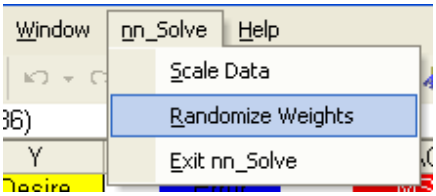
Installing the nn_Solve add-in :

1. Open the folder nn_Solve.zip. You will see the addin **nn_Solve**. Double click on it. This will launch **nn_Solve**.

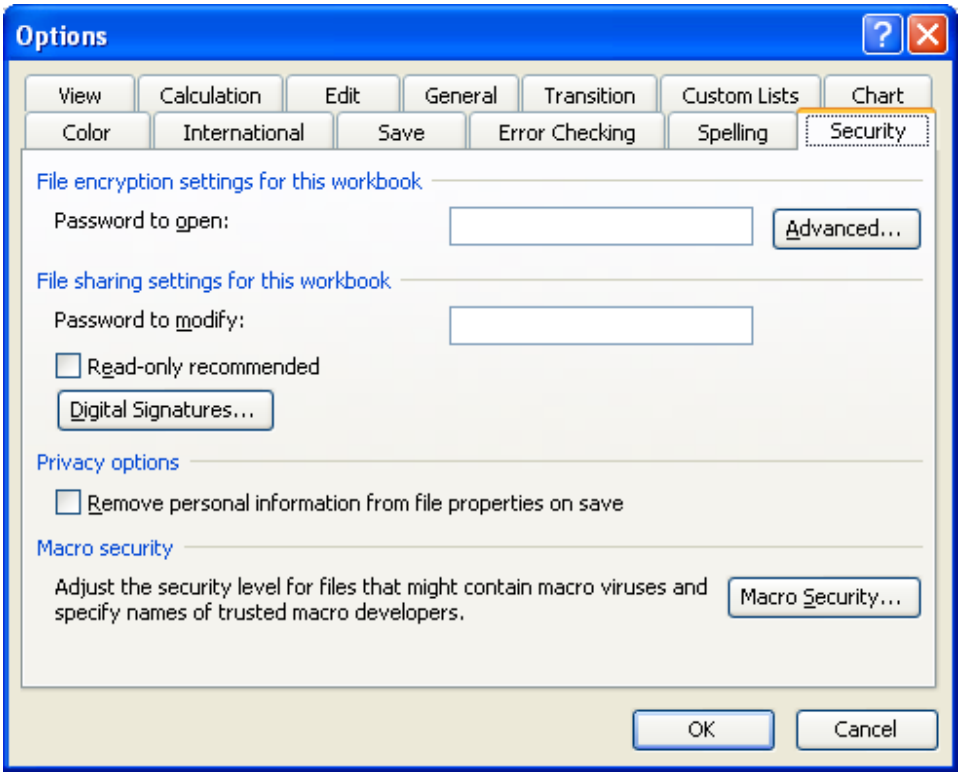
2. Select **Enable Macros** (see below)

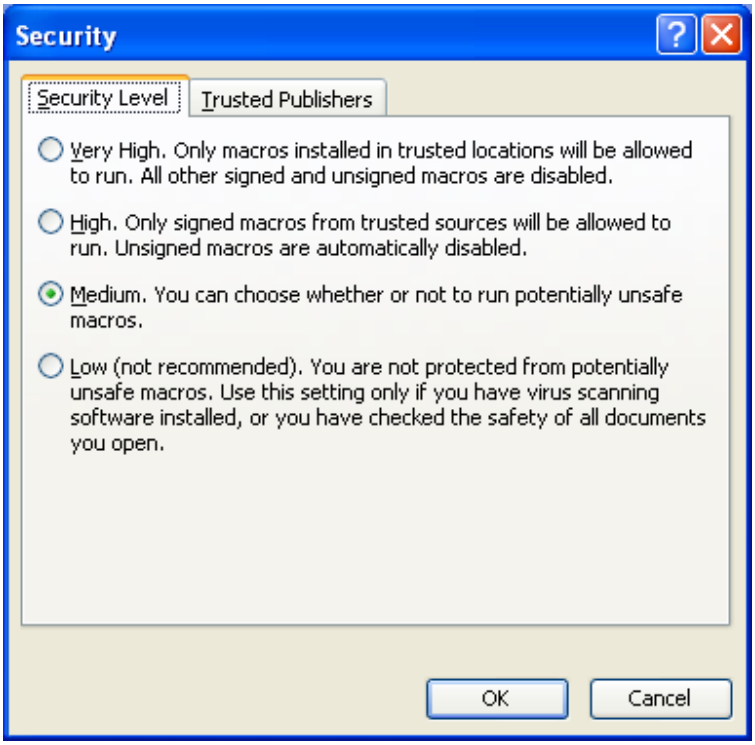


3. You will see **nn_Solve** on the Excel menu bar like below:



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**