Applying Neural Networks Method to Define the Attractiveness of the Price of a Hotel Room

Joshua Beelis

Brookfield High School (CT, USA)

ABSTRACT

The author describes a Neural Net method and its application for finding the binary attractiveness of a hotel's room price. It is assumed that we have a history of the observed hotel's room prices and today's hotel's room price. The algorithm based on the Neural Net method is realized in the MATLAB package.

INTRODUCTION

Recently we see increasing interest in artificial neural networks (ANN's). They are successfully applied to the different fields of human activity – business, medicine. ANN's can be used in situations where we have some relation between the predictors (inputs) and predicted variables (outputs) even if this relation has a complex nature [1]. They are able to reproduce sophisticated dependence after training according to the special algorithm using a representative sample [2].

In this article we will try to simulate neural networks with help of MATLAB Neural Network Toolbox [3] to solve one special problem of pattern recognition.

ARTIFICIAL NEURAL NETWORK OVERVIEW

Initial model of artificial neuron was based on computational representation as a binary threshold unit [4].

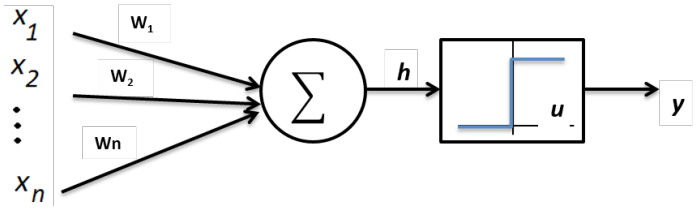

Figure 1. Model of a neuron

This mathematical neuron computes a weighted sum of its n input signals, x_j *(j=1, ...,n)* and generates an output of 1 if this sum is above a certain threshold u (see fig.1). Otherwise, an output of 0 results. Algebraically,

$$y = \theta\left(\sum_{j=1}^{n} w_j x_j - u\right)$$

where θ(·) is unit step function at 0 and wj is the synapse weight associated with the *j*-th input. For simplicity of notation we often consider the threshold *u* as another weight *w0=-u* attached to the neuron with a constant input *x0=1*. Positive weights correspond to *excitatory* synapses, while negative weights model *inhibitory* ones. It is known, that suitably chosen weights let a synchronous arrangement of such neurons perform universal computations. There is a crude analogy here to a biological neuron: wires and interconnections model axons and dendrites, connection weights represent synapses, and the threshold function approximates the activity in a soma. Discussed model, however, contains a number of simplifying assumptions that do not reflect the true behavior of biological neurons.

The McCulloch-Pitts neuron, depicted on fig.1, has been generalized in many ways. An obvious one is to use activation functions other than the threshold function, such as piecewise linear, sigmoid, or Gaussian [5]. The sigmoid function is the most frequently used in artificial neural networks. It is a strictly increasing function that exhibits smoothness and has the desired asymptotic properties. The standard sigmoid function is the logistic function, defined by g(x)=1/(1+e^(-βx)) where β is the slope parameter.

NETWORK ARCHITECTURES

ANN's can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs.

Based on the connection pattern (architecture), ANN's can be grouped into two categories

- *feed-forward* networks, in which graphs have no loops;
- *recurrent* (or *feedback*) networks, in which loops occur because

of feedback connections.

In the most common family of feed-forward networks, called *multilayer perceptron*, neurons are organized into layers that have unidirectional connections between them (see fig.2).

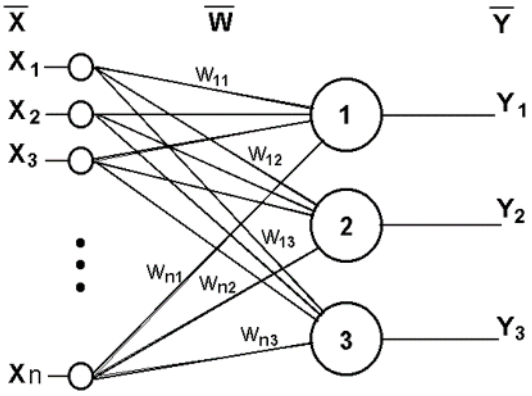

Figure 2. Scheme of two-layer perceptron

Generally speaking, feed-forward networks are *static*, that is, they produce only one set of output values rather than a sequence of values from a given input.

LEARNING

The ability to learn is a fundamental trait of intelligence. Although a precise definition of learning is difficult to formulate, a learning process in the ANN context can be viewed as the problem of updating network architecture and connection weights so that a network can efficiently perform a specific task. The network usually must learn the connection weights from available training patterns.

Performance is improved over time by iteratively updating the weights in the network. ANNs' ability to automatically *learn from examples* makes them attractive and exciting. Instead of following a set of *rules* specified by human experts, ANNs appear to learn underlying rules (like input-output relationships) from the given collection of representative examples. This is one of the major advantages of neural networks over traditional expert systems.

To understand or design a learning process, you need to:

1. Have a model of the environment in which a neural network operates, that is, you must to know what information is available to the network.
2. Realize, how network weights are updated, that is, which *learning rules* govern the updating process.

A *learning algorithm* refers to a procedure in which learning rules are used for adjusting the weights.

ATTRACTIVENESS OF THE HOTEL'S ROOM PRICE

It is well-known that prices of the hotel's room (HR-price) in cities - touristic centers behave like shares on the stock market (see example on fig. 3).

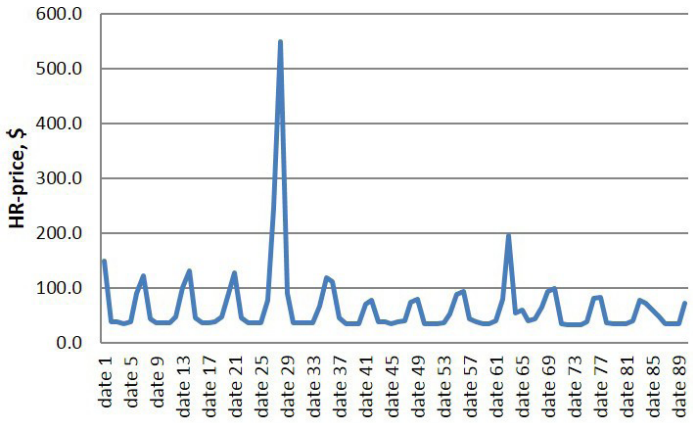

Figure 3. Dynamics of HR-price's changes for the quarter.

The specific is that one can book a hotel room for the year ahead, so we have a deal with the future prices (like futures), but they are not the subject of speculation and could not be traded. The consumer wants to book a hotel room in the city C on a particular date D, having a fixed amount of money M (date of booking B is less than D). It is known, that several hotels (H_1, H_2, ..., H_n) offer their rooms with the prices $P_i \leq M$ on the date D. What is the best choice for the consumer? In other words, which hotel gives a fair price on the date D?

To answer the question we need to define market price A - average price of hotel's room in the city - and know history of HR-price's observations. The idea is to compare HR-price P_i with the market price A by the special formula, which computes attractiveness index AI_j for hotels H_j on the date D:

$$AI_j = \frac{P_B^j}{A_B} - \frac{P_H^j}{A_H} \qquad (1)$$

Here P_B^j is HR-price of the hotel H_j on the date D, obtained at the date B; P_H^j is HR-price of the hotel H_j on the same date D, obtained as an average over a row of the previous observations (cf. [6]). Further, A_B is an average of the HR-prices P_B^j (obtained at the date B) of all N hotels in city C on the date D and A_H is an average of the HR-prices P_H^j for all K previous observations.

If $AI_k > 0$ then we say that HR-price of the hotel H_k is attractive (this hotel lowered HR-price with respect

to the average market price).

Let us remark, that real data with HR-price observations has

omissions, so computing of attractiveness index should be preceded by a special imputation procedure [7].

Applying –Neural Networks Method to Define HR-price Attractiveness

MATLAB package has the most suitable realization of the method [2]. First we need to prepare our data to be used and processed in MATLAB (see fig.4). In this table we have the data, referring to the Hotels 1-4 and including 234 observations. Namely, columns *titled* ***Hotel #j avg*** concerns HR-prices P^j, while lines titled ***Hotel #j checkin*** concerns HR-prices P^j for half-year observation. Lines, named ***City avg*** and ***City avg checkin*** concerns averages prices A_H and A_B respectively for the same period. Binary Attractive Index (BAI_j) for the H_j was defined by the following way:

$$BAI_j = \begin{cases} 1, \text{if} AI_j > 0 \\ 0, if AI_j > 0 \end{cases} \quad (2)$$

The main idea is to create and to train a neural network, which will be able to define BAI without formulas (2)-(3), using only the pattern.

Let us turn to MATLAB workspace (see fig. 5). First line corresponds to loading file with prepared data and saving it as array M (the original data was stored in a file named ***Hotels_NN_t.xls***).

Next step – selection some blocks of M, including 4 rows as inputs (e.g., columns B-E on fig.4) and unit rows of M, followed after mentioned blocks, as a targets (e.g. column F on fig.4). Corresponding MATLAB commands one can see on the fig.6.

To use matrix variable MI and binary vector variable M5 for neural network model we should transposed them by the following command:

x = MI';t = M5';

A	B	C	D	E	F	G	H	I	J	K
	Hotel #1 avg	Hotel #1 chekin	City avg	City avg checkin	Binar Attractive Index BA1j	Hotel #2 avg	Hotel #2 checkin	City avg	City avg checkin	Binar Attractive Index BA2
date 1	24,8	26	89,98	85,85	0	25,17	26	89,98	85,85	0
date 2	25	26	90,81	90,06	0	25,4	26	90,81	90,06	0
date 3	24,5	23	86,61	82,44	1	25,4	26	86,61	82,44	0
date 4	24,29	23	96,65	97,02	1	30,29	26	96,65	97,02	1
date 5	76,89	76	117,54	110,99	0	75,22	64	117,54	110,99	1
date 6	83,14	89	125,11	121,06	0	76	64	125,11	121,06	1
date 7	24,5	23	90,6	86,76	1	35,86	30	90,6	86,76	1
date 8	24,8	26	90,69	86,27	0	30,75	32	90,69	86,27	0
date 9	24,67	23	92,42	86,83	1	33,25	30	92,42	86,83	1
date 10	24,33	26	90,12	86,26	0	31,44	32	90,12	86,26	0
date 11	24,5	26	87,41	81,1	0	32,25	32	87,41	81,1	0
date 12	56	59	116,92	108,98	0	73,2	84	116,92	108,98	0
date 13	59	71	121,78	112,01	0	81,56	80	121,78	112,01	0
date 14	25,3	31	87,44	81,3	0	35,83	34	87,44	81,3	0
date 15	25,25	29	83,87	78,53	0	34,36	38	83,87	78,53	0
date 16	24,91	29	88,29	80,87	0	39	38	88,29	80,87	0
date 17	24,91	29	92,84	85,84	0	42,2	34	92,84	85,84	1
date 18	31,33	35	102,95	95,57	0	52,75	44	102,95	95,57	1
date 19	100,63	166	161,01	153,87	0	89,17	160	161,01	153,87	0
date 20	88	125	160,03	153,34	0	86,36	160	160,03	153,34	0
date 21	39,92	76	112,5	106,1	0	41	72	112,5	106,1	0
date 22	25,94	31	89,37	83,95	0	25,7	32	89,37	83,95	0
date 23	24,5	26	88,31	82,91	0	22,75	26	88,31	82,91	0
date 24	24,59	23	80,72	75,19	0	23,59	28	80,72	75,19	0
date 25	24,65	27	80,07	74,75	0	27,84	31	80,07	74,75	0
date 26	58,18	53	117,04	100,08	0	96,13	80	117,04	100,08	1
date 27	66,59	89	137,83	119,05	0	111,67	100	137,83	119,05	0

Figure 4. Fragment of prepared data

```
Command Window
>> M = xlsread('Hotels_NN_t.xls')

M =

  Columns 1 through 9

   24.8000   26.0000   89.9800   85.8500         0   25.1700   26.0000   89.9800   85.8500
   25.0000   26.0000   90.8100   90.0600         0   25.4000   26.0000   90.8100   90.0600
   24.5000   23.0000   86.6100   82.4400    1.0000   25.4000   26.0000   86.6100   82.4400
   24.2900   23.0000   96.6500   97.0200    1.0000   30.2900   26.0000   96.6500   97.0200
   76.8900   76.0000  117.5400  110.9900         0   75.2200   64.0000  117.5400  110.9900
```

Figure 5. First fragment of MATLAB code

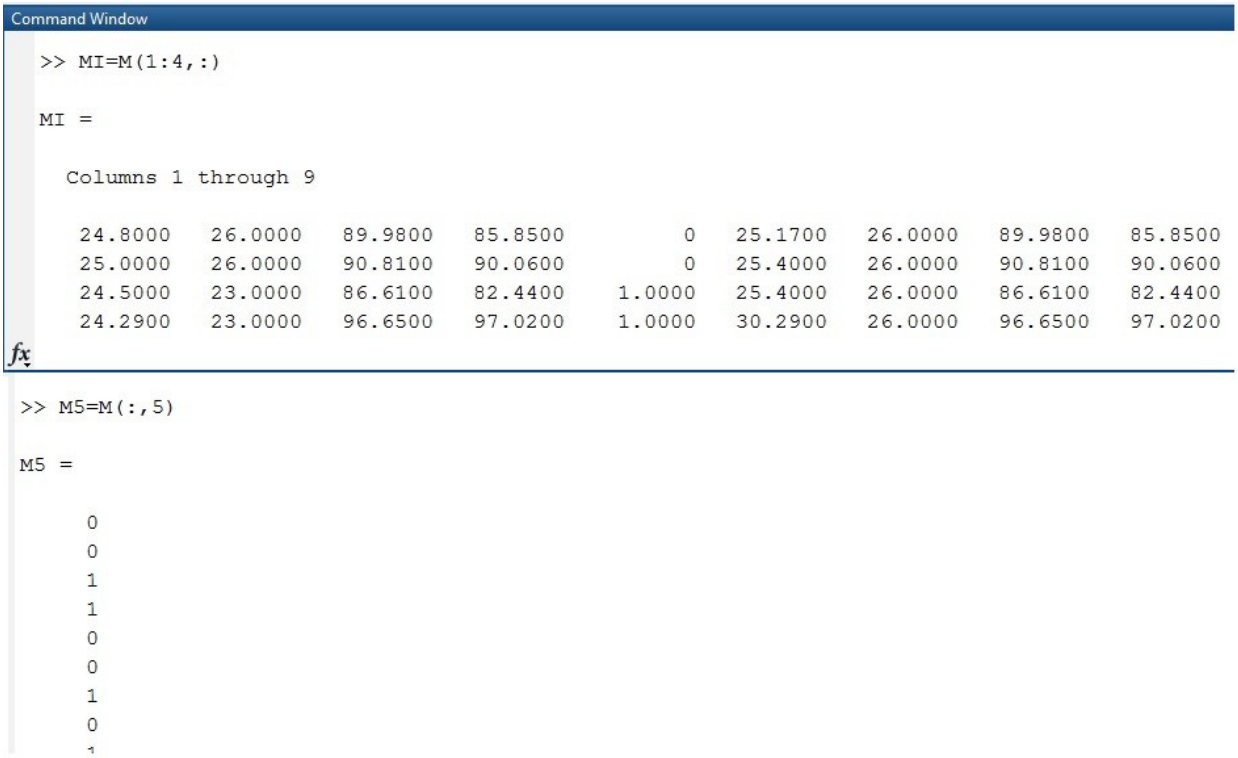

Figure 6. Second fragment of MATLAB code

The next step before training a network is creation the network object. The function ***feedforwardnet*** generates a two-layer network with 10 neurons in the hidden layer. It can be realized as

net = feedforwardnet;

During the configuration step, the number of neurons in the output layer is set to one, which is the number

of elements in each vector of targets:

net = configure(net,x,t);

The configure command also initializes the weights and biases of the network; therefore the network is now ready for training.

Also we need to choose input and output pre/post-processing functions. Our choice is *removing matrix rows with constant values and map matrix row means and deviations to standard values:*

net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

The next step is setup division of data for training, validation and testing:

net.divideFcn = 'dividerand'; % Divide data randomly
net.divideMode = 'sample'; % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

Here a dataset is prepared for dividing into three parts - 70% for designing network, 15% for testing and 15% for validating it.

We are not far from the training of our net. It is necessary to choose training and performance functions now:

net.trainFcn = 'trainbfg'; % quasi-Newton back propagation training function
net.performFcn = 'mse'; % Mean squared error performance function

Also we need to choose plot function to represent results of the net training:

net.plotFcns = {'plotperform'};

Let us start to train our network:

[net,tr] = train(net,x,t);

For testing the network one can use the next (standard) block of commands

y = net(x);
e = gsubtract(t,y);
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);
performance = perform(net,t,y)

If we you run the script, you can see the next windows (fig.7).

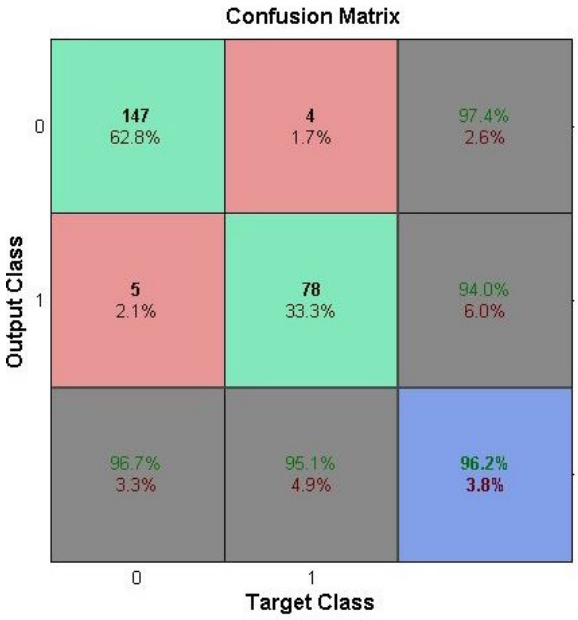

Figure 7. Results of network training

This figure shows the confusion matrices for training, testing, and validation, and the three kinds of data combined. The network outputs are very accurate, as you can see by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrate the overall accuracies.

As we satisfied with the network performance, one can turn to calculate the network response to another input. Let us check the result of BAI prediction for Hotel 3:

MI3=M(:,11:14);x3=MI3';a=net(x3)

The implementation of this line will create a new vector variable *a*, containing 234 numbers, which should be close to 0 or 1 (fig.8).

a

1x234 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	-0.0146	0.0998	-0.0382	0.2114	-0.1867	-0.1682	-0.0030	-0.0166	-0.0311	-0.0069	-0.0736	-0.1897	-0.1825	-0.0707	-0.0781	-0.0852	-0.0546	-0.003
2																		

Figure 8. Results of network using (fragment)

To see the difference between the real and predicted values let us copy transposed line *a* into MS Excel sheet and apply function ROUND (fig.9).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1		date 1	date 2	date 3	date 4	date 5	date 6	date 7	date 8	date 9	date 10	date 11	date 12	date 13	date 14	date 15	date 16	date 17	date 18	date 19	date 20	date 21	date 22	Var23
2	Real Binar Attractive Index BA3j	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Predicted Binar Attractive Index BA3j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 9. Checking the quality of prediction (fragment)

If we continue to compare real ***BAI*** with predicted one, we can be sure that there is only one mistake (date 4). So, it is high quality algorithm, predicting error is less than 0.5%!

CONCLUSION

Neural Network method provides very effective rules for calculation of the hotel's room price binary attractiveness. The result was obtained for two-layer network with 10 neurons in the hidden layer by selection of pre/post-processing functions as

1. removing matrix rows with constant values;
2. map matrix row means and deviations to standard values.

Significant features of the network creation are also choice ***quasi-Newton back propagation function*** as a training function and ***Mean squared error*** function as a performance one.

REFERENCES

1. *Activation Function.* (n.d.). Retrieved from http://en.wikipedia.org/wiki/Activation/function
2. *Artificial Neural Network.* (n.d.). Retrieved from http://en.wikipedia.org/wiki/Artificial_neural_network
3. Bogatov, E., & Bogatov, V. (2013). On the definition of attractive prices degree for hotel rooms. *Bulletin of Belgorod University of Economics, and Law of Cooperatives*, 45, 251-255.
4. Buuren, V. (2012). Flexible imputation of missing data. *CRC Press*, 342.
5. *Create, Train, and Simulate Neural Networks.* (n.d.). Retrieved from Neural Network Toolbox: http://www.mathworks.com/products/neural-network/
6. Jain, A., Mao, J., & Mohiudden, K. (1996). Artificial Neural Networks: A tutorial. 31-44.
7. Stergiou, C., & Siganos, D. (n.d.). *Neural Networks.* Retrieved from http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html