APPLICATION OF CLOUD-BASED SPREADSHEETS TO ARTIFICIAL NEURAL NETWORK MODELLING

Oksana Markova
PhD, Senior Lecturer
Department of Computer Systems and Networks
Kryvyi Rih National University
11 V. Matusevycha str., Kryvyi Rih, Ukraine, 50027
markova@mathinfo.ccjournals.eu

Serhiy Semerikov
Doctor of Pedagogical Sciences, Professor
Department of Computer Science and Applied Mathematics
Kryvyi Rih State Pedagogical University
54 Gagarina ave., Kryvyi Rih, Ukraine, 50086
semerikov@gmail.com

Abstract: The article substantiates the necessity to develop methods of computer simulation of neural networks in the spreadsheet environment. The systematic review of their application to simulating artificial neural networks is performed. The authors distinguish basic approaches to solving the problem of network computer simulation training in the spreadsheet environment, joint application of spreadsheets and tools of neural network simulation, application of third-party add-ins to spreadsheets, development of macros using the embedded languages of spreadsheets; use of standard spreadsheet add-ins for non-linear optimization, creation of neural networks in the spreadsheet environment without add-ins and macros. It is shown that to acquire neural simulation competences in the spreadsheet environment, one should master the models based on the historical and genetic approach. The article considers ways of building neural network models in cloud-based spreadsheets, Google Sheets. The model is based on the problem of classifying multidimensional data provided in "The Use of Multiple Measurements in Taxonomic Problems" by R. A. Fisher. Edgar Anderson's role in collecting and preparing the data in the 1920–1930s is discussed as well as some peculiarities of data selection.

Keywords: Anderson's Iris, computer simulation, neural networks, cloud-based spreadsheets.

1. Introduction

For the past years, the authors have been developing the concept of systematic computer simulation training at universities. Spreadsheets are chosen to be the leading environment for computer simulation training, their application discussed in articles [1, 2]. Using spreadsheet processors (autonomous, integrated and cloud-based) as examples, the authors demonstrate components of technology of computer simulation of determined and stochastic objects and processes of various nature. Extensive application of artificial intelligence in everyday life calls for students' early acquaintance with its models and methods including neural network-based while teaching machine learning [3]. It conditions the need for developing training methods of computer simulation of neural networks in the general-purpose simulation environment, i.e. spreadsheets.

The first description of spreadsheet application to artificial neural network simulation of visual phenomena belongs to Thomas T. Hewett, Professor of the Department of Psychology of Drexel University [4]. His approach implies simultaneous studying a neural network and understanding its functioning as psychology students conclude the laws of the neural impulse spread by applying the trial-and-error method.

The patent "Embedding neural networks into spreadsheet applications" [5] describes an artificial neural network with a plurality of processing elements called neurons arranged in layers. A network has an input layer, an output layer, and one or more "hidden" layers in between, necessary to allow solutions of non-linear problems. Each unit (in some ways analogous to a biological neuron: dendrites – input layer, axon – output layer, synapses – weights, soma – summation function) is capable of generating an output signal which is determined by the weighted sum of input signals it receives and an activation function specific to that unit. A unit is provided with inputs, either from outside the network or from other units, and uses these to compute a linear or non-linear output. The unit's output goes either to other units in subsequent layers or to outside the network. The input signals to each unit are weighted by factors derived in a learning process. In his patent, Ruggiero details a network structure (multi-level), an activation function (sigmoidal), a coding method (polar), etc. He presents a mathematical apparatus for network training and determines a method of data exchange between a spreadsheet processor nucleus and an add-in to it. The patent author suggests storing input data in columns, maximum and minimum values for each column of input data, the number of learning patterns. Data can be normalized or reduced to the polar range [0; 1] both in spreadsheets and add-ins.

The authors of [6] give an example of applying the non-linear optimization tool, Microsoft Excel Solver, to forecasting stock prices using the "grey-box" concept, in which the model is evident, yet, the details of its realization are hidden.

According to [7], Anderson first selected *Iris versicolor*, the common blue flag, because he believed it to be clearly defined, and it was common and easily observed. He recorded several morphological characters in more than 2,000 individuals belonging to 100 populations, data far more extensive than those that any botanist had yet obtained on a single species. In the famous article "The Use of Multiple Measurements in Taxonomic Problems" indicating that "Table I shows measurements of the flowers of fifty plants each of the two species *Iris setosa* and *I. versicolor*, found growing together in the same colony and measured by Dr E. Anderson, to whom I am indebted for the use of the data" [8]. Fisher's article contained only three references two of which to Anderson's works – that of [9] and that of [10] marked with "(in the Press)". The set of data used by Fisher and collected by Anderson was introduced as "Iris flower data set" (or "Iris data set" and "Iris data"). The phrase "Fisher's Iris data set" traditionally expresses Fisher's role as the founder of linear discriminant analysis, but not the authorship of the data set. So, it is possible to pay tribute to Edgar Anderson by naming this data set after him – Anderson's Iris data set.

2. Methods

Let's consider the pattern classification problem by taking a Anderson's Iris data, composed of data on 150 measurements of three Iris species – *Iris setosa*, *Iris virginica* and *Iris versicolor*) – including 50 measurements for each species. There were measured four features: sepal length (SL), sepal width (SW), petal length (PL), and petal width (PW).

To draw a grounded conclusion on the *Iris* type, let's build a three-layered neural network with the following architecture (**Fig. 1**):

– the input layer is a four-dimensional arithmetical vector (x_1, x_2, x_3, x_4) the components of which are corresponding measured features of Anderson's Irises (SL, SW, PL, PW) normalized according to the network activation function;

– the hidden layer has dimension 9 (the minimal required number according to Kolmogorov-Arnold representation theorem) and is described by the vector (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9);

– the output layer is a three-dimensional arithmetical vector (y_1, y_2, y_3) the components of which are probabilities indicating the correspondence of the data set to one of the three *Iris* types.

The bias neuron equal to 1 (marked red in **Fig. 1**) is added to the neurons of the input and hidden layers. The bias neurons are noted for not having synapses so they cannot be located in the output layer.

3. Results

Let's first introduce Anderson's Irises into spreadsheets with the following values of cells: A1 is *Iris Data*, A2 is SL, B2 is SW, C2 is PL, D2 is PW, E2 is *Species*. The table cells A3:E152 include Anderson's Irises. It isn't possible to input the data of the given set into the input layer as the value of the four characteristics is beyond the range limits [0; 1]. The next step is normalization of columns A, B, C and D to meet the given range and coding of *Iris* types from column E.

Each *Iris* type is coded by the three-dimensional arithmetical vector: for *i*-Iris (*Iris setosa* is 1, *Iris versicolor* is 2, *Iris virginica* is 3) let's set the *i*-th component in 1, and the other ones – in 0. To do this, let's introduce the following values into the cells: G1 is *encoding*, G2 is *setosa*, H2 is *versicolor*, I2 is *virginica*, G3 is =if($E3=G$2,1,0).

Next, let's copy the formula from the cell G3 to the range G3:I152 and obtain the following model codes for the three *Iris* types: for *Iris setosa* – (1, 0, 0), for *Iris virginica* – (0, 0, 1) and for *Iris versicolor* – (0, 1, 0).

Each column is normalized separately. To perform this, let's find minimum and maximum values by introducing the following values: E154 is min, E155 is max, A154 is =min(A3:A152), A155 is =max(A3:A152). Let's apply the cells A154:A155 to the range B154:D155 and introduce the following values into the cells: K1 is *normalization*, K2 is x_1, L2 is x_2, M2 is x_3, N2 is x_4, K3 is =(A3–A$154)/(A$155–A$154). The latter formula is applied to the range K3:N152. Its essence is explained by: normalization=(value–min)/(max–min). This approach results in the minimum value normalized to 0, while the maximum one – to 1.

According to the chosen architecture, let's add the bias neuron to the four neurons of the input layer by introducing its name (x_5) into the cell O2 and its value (1) into the range O3:O152. On this stage, the input layer is formed as x_1, x_2, x_3, x_4, x_5.

The next step includes transmission of a signal from the input layer to the hidden one of the neural network. Let's denote the weight coefficient of the synapse connecting the neuron x_i (i=1, 2, 3, 4, 5) of the input layer with the neuron h_j (j=1, 2, ..., 9) of the hidden layer by w_{ij}^{xh}, while the weight coefficient connecting the neuron h_j of the hidden layer with the neuron y_k (k=1, 2, 3) of the input layer is denoted by w_{jk}^{hy}. In this case, the force of the signal coming to the neuron h_j of the hidden layer is determined as a scalar product of signal values on the input signals and corresponding weight coefficients. To determine a signal going further to the output layer, let' apply the logistic function of activation $f(S)=1/(1+e^{-S})$, where S is a scalar product. The formulae for determining the signals on the hidden and output layers will look like:

$$h_j = f\left(\sum_{i=1}^{4+1} x_i w_{ij}^{xh}\right), \quad y_k = f\left(\sum_{j=1}^{9+1} h_j w_{jk}^{hy}\right).$$

Accordingly, two matrices should be created. The matrix w^{xh} of 5×9 contains weight coefficients connecting five neurons of the input layer (the first four contain normalized characteristics of Anderson's Irises, while the fifth one is the bias neuron) with the neurons of the hidden layer. The matrix w^{hy} of 10×3 contains weight coefficients connecting ten neurons of the hidden layer (nine of which are calculated and the tenth one is the bias neuron) with the neurons of the output layer. For the "untaught" neural network, initial values of the weight coefficients can be set either randomly or left undetermined or equal to zero. To realize the latter, let's fill the cells with the following values: R1 is w^{xh}, Q2 is input/hidden, R2 is 1, S2 is =R2+1, Q3 is 1, Q4 is =Q3+1, R3 is 0, R9 is w^{hy}, Q10 is hidden/output, R10 is 1, S10 is =R10+1, Q11 is 1, Q12 is =Q11+1, R11 is 0. To create the matrices, let's copy the cells R3 into the range R3:Z7, R11 – into R11:T20, S2 – into T2:Z2, Q4 – into Q5:Q7, S10 – into T10, Q12 – into Q13:Q20.

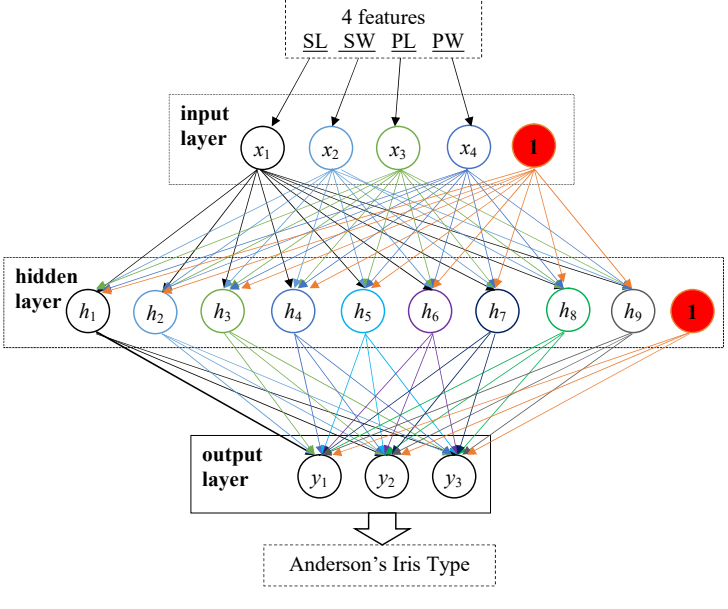

Fig. 1. Architecture of the neural network to solve the problem of Anderson's Iris classification

To calculate the scalar product of the vector row of the input layer values by the matrix vector-column of the weight coefficients w^{hy}, let's apply the matrix multiplication function: AB1 is calculate the hidden layer, AB2 is h_1, AC2 is h_2, AD2 is h_3, AE2 is h_4, AF2 is h_5, AG2 is h_6, AH2 is h_7, AI2 is h_8, AJ2 is h_9, AK2 is h_{10}, AB3 is =1/(1+exp(mmult($K3:$O3,R$3:R$7))), AK3 is 1. Next, let's copy the cell AK3 into the range AK4:AK152, while AB3 – into AB3:AJ152.

Considering the fact that all the matrix elements of the weight coefficients w^{xh} equal to zero, after duplicating the formulae, the calculated elements of the hidden layer will be equal to 0.5.

In the same way, let's calculate the output layer elements: AM1 is calculate the output layer, AM2 is y_1, AN2 is y_2, AO2 is y_3, AM3 is =1/(1+exp(mmult($AB3:$AK3,R$11:R$20))). Next, let's copy the cell AM3 to the range AM3:AO152.

Neural network training is performed by varying weight coefficients so that with each training step the difference between the calculated values of the output layer and the desired (reference ones) reduces. To solve the problem, the three-dimensional vectors resulted from coding of the three Iris types are reference. To find the difference between the calculated and the reference output vectors let's apply the Euclidean distance: AQ2 is distance, AR2 is sum of distances, AQ3 is =sqrt((AM3-G3)^2+(AN3-H3)^2+(AO3-I3)^2), AR3 is =sum(AQ3:AQ152). Next, let's copy the cell AQ3 to the range AQ4:AQ152. The cell AR3 contains general deviation of the calculated output vectors from the reference ones.

Under this approach, the neural network training can be treated as an optimization problem in which the target function (the sum of distances in the cell AR3) will be minimized by varying the matrix weight coefficients w^{xh} (the range R3:Z7) and w^{hy} (the range R11:T20). To solve this problem, application of cloud-based spreadsheets (Google Sheets) is not enough and it is necessary to install an additional cloud-based component (add-in) Solver. Adjustment of the add-in Solver to solve the set goal: the target function (Set Objective) is minimized (To: Min) by changing the values (By Changing) of the matrix weight coefficients in the range (Subject To) from –10 to +10 by one of the optimization methods (Solving Method). To reduce the total distances, the actions with Solver can be done repeatedly as it is expedient to experiment with combination of various optimization methods by changing the variation limits of the weight coefficients. It is not necessary to try to reduce the value of the total distances to zero as this can be a greater (quite smaller) value.

On the assumption of the chosen coding method, the output vector actually contains three probabilities: y_i denotes the probability of the given sample being the i-type Iris, where i=1 for *Iris setosa*, 2 for *Iris versicolor* and 3 for *Iris virginica*. Then, to find out which *Iris* type describes the input vector (SL, SW, PL, PW), the most probable component should be determined. To do this, let's fill the cells in the following way: AT2 is Calculated Iris species, AT3 is =if(max(AM3:AO3)=AM3,G2,if(max(AM3:AO3)=AN3,H2, I2)), AU3 is =if(AT3=E3,»right!»,»wrong»). Next, the range AT3:AU3 is copied to the range AT4:AU152.

The obtained result enables to visualize pattern recognition simulated in spreadsheets. The built model will be considered relevant in all 150 cases, the column AU contains the value «right!» To check the limits of the built model application, let's try to input the vector not coinciding with any reference input vector. For this, let's copy the table row 152 to 158 and delete the content of the cells E158:I158, AQ158, AU158. Let's introduce averaged values borrowed from the description of *Iris versicolor* in the article by Anderson [10]: 5.50, 2.75, 3.50 and 1.25. The reference values x_1=0.3333, x_2=0.3125, x_3=0.4237, x_4=0.4792 are conveyed to the input layer, while on the hidden layer there are calculated h_1–h_9 and the values of the output layer y_1=0.0000, y_2=1.0000, y_3=0.0000. As the maximum value of the output layer 1.0000 corresponds to the other *Iris* type, it is possible to conclude that *Iris versicolor* is identified.

4. Discussion and conclusions

The conducted review makes it possible to find the following solutions of the problem of computer simulation of neural networks in the spreadsheet environment:

1) joint application of spreadsheets and neural network tools, in which data is exported to the unit calculating weighting factors imported to spreadsheets and used in calculations;

2) application of third-party add-ins for spreadsheets, according to which structured spreadsheet data is processed in the add-in, calculation results are arranged in spreadsheet cells;

3) macros development enables direct software control over neural network training and creation of a user's specialized interface;

4) application of standard add-ins for optimization calls for transparent network realization and determination of an optimization criterion (minimization of a squared deviation total of the calculated and etalon outputs of the network);

5) creation of neural networks in the spreadsheet environment without add-ins and macros requires transparent realization of a neural network with evident determination of each step of adjustment of its weighting factors.

Edgar Anderson appeared to be not a simple botanist whose data were the basis for Fisher's known method. Anderson's Irises resulted from his long experience of working out relevant models to describe changes in specific populations by means of a limited number of characteristics. Yet, Anderson had also coped with the opposite problem of building simple multi-dimensional data interpretation 40 years before Chernoff faces appeared.

The described methods of applying cloud-based spreadsheets as a machine learning tools can enable solution of all basic problems of neural network simulation. The only limitation is not so much the volume of a spreadsheet as the memory space and the speed of the device processing it. In the special course projects if the limitation is overcome, this becomes a stimulus for replacing the simulation environment by a more relevant one.

References

1. Semerikov, S. O., Teplytskyi, I. O., Yechkalo, Yu. V., Kiv, A. E. (2018) Computer Simulation of Neural Networks Using Spreadsheets: The Dawn of the Age of Camelot. CEUR Workshop Proceedings, 2257, 122–147. Available at: http://ceur-ws.org/Vol-2257/paper14.pdf
2. Semerikov, S. O., Teplytskyi, I. O., Yechkalo, Yu. V., Markova, O. M., Soloviev, V. N., Kiv, A. E. (2019). Computer Simulation of Neural Networks Using Spreadsheets: Dr. Anderson, Welcome Back. CEUR Workshop Proceedings, 2393, 833–848. Available at: http://ceur-ws.org/Vol-2393/paper_348.pdf

3. Zaremba, T. (1990). Case Study III: Technology in Search of a Buck. Neural Network PC Tools, 251–283. doi: https://doi.org/10.1016/b978-0-12-228640-7.50018-0
4. Hewett, T. T. (1985). Using an Electronic Spreadsheet Simulator to Teach Neural Modeling of Visual Phenomena (Report No. MWPS-F-85-1). Drexel University, Philadelphia.
5. Ruggiero, M. (1993). Pat. No. 5,241,620 US. Embedding neural networks into spreadsheet applications. declareted: 31.08.1993.
6. Kendrick, D. A., Mercado, P. R., Amman, H. M. (2006). Computational Economics. Princeton University Press. doi: https://doi.org/10.1515/9781400841349
7. Stebbins, G. L. (1978). Edgar Anderson 1897-1969. National Academy of Sciences, Washington.
8. Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7 (2), 179–188. doi: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x
9. Anderson, E. (1935). The irises of the Gaspé Peninsula. Bulletin of the American Iris Society, 59, 2–5.
10. Anderson, E. (1936). The Species Problem in Iris. Annals of the Missouri Botanical Garden, 23 (3), 457. doi: https://doi.org/10.2307/2394164

Received date 09.10.2019
Accepted date 04.11.2019
Published date 23.11.2019