ELLIPSE PERIMETER ESTIMATION USING NON-PARAMETRIC REGRESSION OF RBF NEURAL NETWORK BASED ON ELLIPTIC INTEGRAL OF THE SECOND TYPE

Sobhan Hemati[1]*, Peyman Beiranvand** and Mehdi Sharafi*
*Department of Electrical & computer engineering, University of Tehran, Iran
**Department of Civil Engineering, Razi University, Iran..

ABSTRACT
Methods for calculating the ellipse perimeter provided so far, including Kepler equation, Euler, Ramanujan, Lindner, Gauss Kumar, do not have acceptable accuracy in some cases. In this study, calculation of ellipse perimeter was done using non-parametric regression of RBF neural network. To train the neural network, the data from the numerical calculation of second type elliptic integral was used. We used the least squares and gradient descent optimization methods to train the neural network and the deviation of the output of these two methods from the accurate method was calculated, using generalization error and learning error measures. Studies show that after the training phase, the network as an individual model, can estimate the ellipse perimeter for different values of eccentricity and major diameter with great accuracy.

KEYWORDS: ellipse perimeter, RBF neural network, elliptic integral of the second type, least squares, gradient descent.

MSC: 62J02

RESUMEN
para calcular el perímetro de la elipse se proveen métodos que incluyen las ecuaciones de Kepler, Euler, Ramanujan, Lindner, Gauss Kumar, que poseen una aceptable exactitud en algunos casos. En este estudio el cálculo del perímetro de la elipse se realizó usando regresión no-paramétrica de redes neuronales RBF. Para entrenar la red neuronal los datos derivados de los cálculos numéricos fueron utilizados. Usamos métodos de optimización mínimos cuadráticos y de gradiente descendente para entrenar la red neuronal y las desviaciones de la salida de los dos métodos del método acurado se calculó usando medidas de los errores de generalización y de aprendizaje. Los estudio muestran que tras la fase de entrenamiento la red como un modelo individual, puede estimar el perímetro de la elipse con diversos valores de excentricidad y mayor diámetro con gran exactitud.

PALABRAS CLAVE: perímetro de una elipse, RBF red neuronal, integral elíptica de segundo tipo, mínimos cuadráticos, Gradiente descendente.

1. INTRODUCTION

Ellipse is a set of points in a plane that their sum of distances from two fixed points called foci of the ellipse is constant. The exact calculation of the ellipse perimeter leads to elliptic integral of the second type that has no analytical solution. In recent years, researchers have tried to provide a relationship for calculating perimeter of ellipse, but still there is no precise relationship for this calculation. In the articles [4,7,8] some single terms for estimating ellipse perimeter that have been introduced during the last 4 centuries are investigated. (Kepler, 1609) and (Euler, 1738) introduced some approximate relationships to calculate perimeter of ellipse, but their relationship had increasing error in the case of close to one eccentricity. (Muir, 1883), (Lindner, 1904), (Ramanujan, 1914), (Hudson, 1917) and (zafary, 2006) also provided equations for calculating the perimeter of ellipse that were not accurate enough. [1] Introduced a polynomial using Lagrange Interpolation Method to estimate the perimeter of ellipse and showed that this polynomial with a degree of greater than 7 can do this quite accurately. Although this method improves the accuracy of estimation compared to previous methods, but further examination showed that the perimeter estimated using this method is also a bit deviated from the perimeter calculated by the numerical solution of elliptic integral of the second type. This error was caused by

[1] hemati.sobhan@ut.ac.ir

two phenomena. First, they carried out the interpolation on the perimeters calculated by Euler's method and it's clear that Euler's method itself had error. Second, Lagrange interpolation uses Euler method to fit a polynomial in the calculated perimeters and we know that this prior premise of the model is not necessarily the best one possible! As we know, the perimeter of ellipse is a function of its major diameter and eccentricity. Another objection to this view is that the introduced polynomial was only a function of eccentricity of the ellipse which means they considered the major diameter of the ellipse to be constant i.e. by changing the major diameter, the estimated polynomial would change; so they couldn't offer an integrated model to estimate ellipse perimeter. Calculation of exact ellipse perimeter lead to elliptic integral of the second type and as we know there is no analytic solution for this integral. In this study we will train RBF neural network based on perimeters that they have calculated with numerical solution of elliptic integral of the second type rather than Euler's method and we do a nonparametric regression on this data by mentioned neural network. So the estimated perimeter with this network will be closer to exact ellipse perimeter because we use more accurate data for training the neural network and also we don't consider any prior model for estimate the perimeter. Note that after neural network training we can estimate the ellipse perimeter without using numerical method while the result of our method is very close to exact method. On the other hand, this network is trained by combined input of major diameter and eccentricity. So after training the network, we proposed a function to estimate ellipse perimeter which has the same parameters for all axes and doesn't change.

2. METHODS

As said before, we're going to fit a curve on perimeters calculated by numerical solution of elliptic integral of the second type, using neural network. In [10,6,2], principles of estimating a function using neural network is expressed. There are different kinds of neural networks, including multilayer perceptron, RBF neural network and wavelet neural network that have been established as non-parametric regression tools. RBF neural network which was used in this study is one of the most popular methods in non-parametric regression because of its simplicity. Figure 1 shows an overview of the RBF neural network.

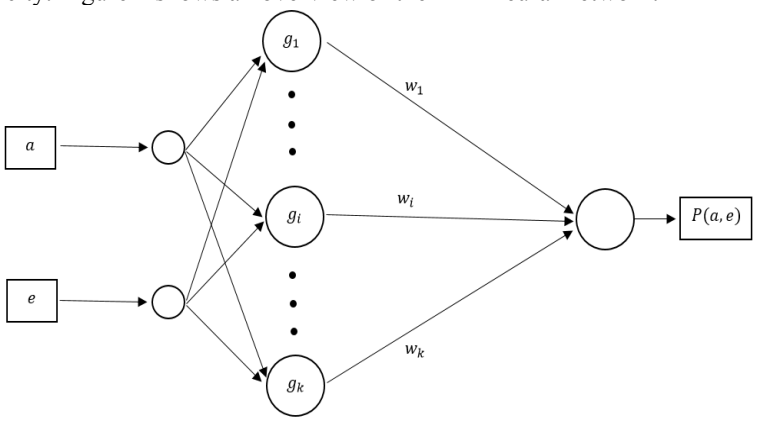

Figure 1. RBF neural network for non-parametric regression

Figure 1 shows the RBF neural network , $x = \begin{bmatrix} a \\ e \end{bmatrix}$ is the input vector of network which *a* and *e* are major diameter and eccentricity, respectively. After entering these values in the network, elements of this vector are used as input for radial basis functions $g_1$to g_k. These radial basis functions are usually considered to be Gaussian functions like Eq.2 that need two values of mean vector c and covariance matrix Σ. For each input $x = \begin{bmatrix} a \\ e \end{bmatrix}$, k outputs (Each output corresponding to a radial basis function) are appeared in the network and these obtained values are multiplied by the weights w_1 to w_k, respectively. These weights can also be seen on the network edge in Figure 1. Finally, by summing of all these values, RBF neural network output is obtained which is equivalent to P or the estimated perimeter of the ellipse. Eq.1 shows the relationship between the input and output of the network.

$$p_i(a,e) = \sum_{i=1}^{k} w_i g_i \quad (1)$$

in which g_i is the radial basis function defined as Eq. 2

$$g_i = e^{-\frac{1}{2}(x-c_i)^T \Sigma^{-1}(x-c_i)} \quad (2)$$

In this network, the number of neurons, values of vector c and covariance matrix Σ should be determined for each neuron. Also the weights of w_1 to w_k should be determined to which is done with neural network training but before this, we have to create training data. As previously mentioned, this is done using numerical solution of elliptic integral of the second type for each different values of the radius and eccentricity.
It's stated in the fundamentals of Calculus that the perimeter of ellipse can be proposed by parametric equation $\frac{X^2}{a^2}+\frac{Y^2}{b^2}=1$ as Eq.3 using arc length Integral:

$$z(a,e)=\int_0^{2\pi}\sqrt{1+y'^2}dx = 4\int_0^{\frac{\pi}{2}}\sqrt{1+y'^2}dx = 4a\int_0^{\frac{\pi}{2}}\sqrt{1-e^2sin^2x}\,dx \quad (3)$$

in which a is the major diameter and e is eccentricity.
But as previously stated, the phrase under the integral in equation (3) does not have a primary function. So, calculating the perimeter of ellipse accurately using major diameter and eccentricity would inevitably lead to the numerical solution of the above integral.
In this study, we calculate the perimeter of ellipse for major diameter in the range of $a\epsilon[0,10]$ with a step of 0.05 and eccentricity in the range of $e\epsilon[0,1]$ with a step of $\frac{1}{30}$ using numerical solution of Eq.3. The set generated from the training data is called Tr and demonstrated as a triple set Tr = (a, e, Z). It is obvious that Z is the output of integral (1) i.e. the perimeter of ellipse. In order to evaluate the performance of the neural network, we use a set of data that were not used in training phase. In order to generate test data, we use numerical solution of Eq.3 with axes range of $a\epsilon[0,9]$ with a step of 0.03 and eccentricity range of $e\epsilon[0,1]$ with a step of $\frac{1}{22}$.

2.1 Setting network parameters

In the next step, after creating the training data, parameters of the neural network should be determined. First, the number of neurons should be determined. Since our goal is fitting the curve as accurately as possible, we considered a neuron for each observation (each ordered pair of major diameter and eccentricity), so the number of neurons is equal to the number of training data (number of observations). As stated earlier, observations calculated by numerical solution of Eq.3 for different major diameters and eccentricities are monotonic in the feature space. As a result in order to the curve to intersect with all of data points, we considered each data a cluster; it means the value of c_i vector is equal to the ith ordered pair of $x=\begin{bmatrix}a\\e\end{bmatrix}$.
Covariance matrix is a 2×2 matrix$\left(\begin{bmatrix}\sigma_{1,1}^2 & \sigma_{1,2}^2\\ \sigma_{2,1}^2 & \sigma_{2,2}^2\end{bmatrix}\right)$ that 4 element must be determined for it. Since feature space is symmetrically covered by the data, we consider the g_i functions ones circular symmetrical and to accomplish this, the elements $\sigma_{2,1}^2$ and $\sigma_{1,2}^2$ have to be zero. Since we've considered a neuron (a radial basis function) for each ordered pair $x=\begin{bmatrix}a\\e\end{bmatrix}$, the elements $\sigma_{1,1}^2$ and $\sigma_{2,2}^2$ of covariance matrix are selected so small that each g_i function covering its corresponding observation, also covers the space between two adjacent g_i functions so we could have a good interpolation for unobserved data. Here we've chosen the $\sigma_{1,1}^2$ and $\sigma_{2,2}^2$ elements of the matrix 0.025^2 and 0.0125^2, respectively. All of these g_i functions have the same covariance matrix.
Now that the network parameters are determined, we fit a non-parametric curve to data generated by Eq.3 using neural network, without any prior knowledge about the model providing the perimeter of ellipse. To do this, we calculate the weights of this network by two different approaches called least squares and gradient descent and then we compare the results.

2.2 Training RBF neural network using least squares optimization method

Different methods have been proposed for calculating the weight of RBF neural networks. The least squares method is explained by [5,9] for training the network and calculating its weight. The algorithm used for calculating the weights is explained in detail in the following section. Suppose an ordered pair $x=\begin{bmatrix}a\\e\end{bmatrix}$ are inputted to network and corresponding output with K values of radial basis functions were calculated for this input. Now we do a normalization on the output of each neuron; i.e. we divide the output of each neuron by

the sum of all outputs. It'll cause all outputs to be in the same range which improve our estimates by a small amount. We also add a norm with value 1 in order to model the bias and improve the degree of freedom (Suppose that there is an additional neuron which has the output of 1, regardless of the input). Now according to Figure 1, if the weight of the network is known to us, we can calculate the neural network as follows:

$$p_i(a,e) = \sum_{i=1}^{k} w_i g_i \quad (4)$$

Parameters a, e, g_i, w_i and p_i are major diameter of ellipse, ellipse eccentricity, radial basis functions, weights of the network and the outputs, respectively. The purpose of the training process is to use Tr data sets for finding the weight of the network so that the learning error in the following formula is minimized:

$$E_l = \frac{1}{2n}\sum_{j=1}^{n}(z_j(a,e) - p_j(a,e))^2 \quad (5)$$

so

$$E_l = \frac{1}{2n}\sum_{j=1}^{n}[z_j(a,e) - \sum_{j=1}^{k} w_j g_j]^2 \qquad j = 1,\dots,k \ , i = 1,\dots,n \quad (6)$$

The matrix form of equation (6) is as below:

$$E_l = \frac{1}{2n}\ (GW - Z)^T(GW - Z) \quad (7)$$

Were W is $k\times1$ and k is number of RBF network weights ($k = n + 1$). G is a $n\times k$ matrix which its rows are the number of entries and the columns are equal to the number of neurons plus one. For example, the element in first row of the second column is the response of the second neuron to first input and the last column only includes 1 which is used for modelling the bias. And Z vector, is $n\times1$ where n is number of ellipse perimeters obtained from equation (3), for n distinct inputs.

Now the final equation of weights of RBF networks could be obtained by taking derivative from equation (7) with respect to W vector as follows:

$$W = (G^T G)^{-1} G^T Z \quad (8)$$

2.2.1 Least squares results

After calculating the weights of the network, we must examine the accuracy of the ellipse perimeter estimated by RBF neural network. To do this, we introduce the standards of learning error and generalization error which are expressed by equations (9) and (10). Learning error is calculated as follows:

$$E_l = \frac{1}{2n}\sum_{i=1}^{n}(z_i - p_i)^2 \quad (9)$$

were z_i is the perimeters calculated by Eq. 3 for the network training data, p_i is the perimeters calculated by neural network for the inputs used to train the network and n is the number of distinct ordered pairs of major diameter and eccentricity.

Generalization error can be calculated as follows:

$$E_g = \frac{1}{2m}\sum_{i=1}^{m}(z_i - p_i)^2 \quad (10)$$

were z_i is the perimeters calculated by Eq. 3 for the network training data, p_i is the perimeters calculated by neural network for the inputs that the network didn't observe them and m is the number of distinct ordered pairs of major diameter and eccentricity of test data. It should be noted that as it was mentioned before, test data are the data which were not used for training the network. For this method learning error and generalization error are zero (10^{-23}) and 0.000769, respectively.

Figure 2 shows the output of Eq.3 i.e. the exact value of perimeter of ellipse and Figure 3 shows the output of the RBF neural network i.e. the estimated perimeters of ellipse in terms of major diameter and eccentricity, for a subset of training data. Figure 4 shows the output of Eq.3 i.e. the exact value of perimeter of ellipse and Figure 5 shows the output of the RBF neural network i.e. the estimated perimeters of ellipse in terms of major diameter and eccentricity, for a subset of training data.

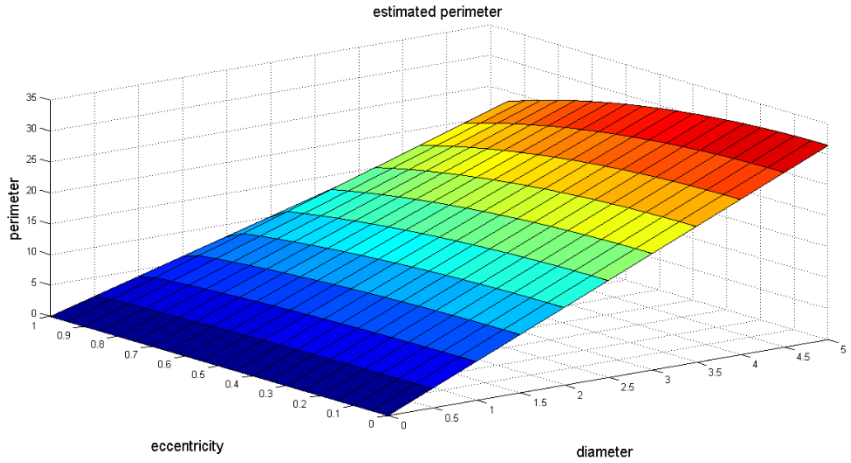

Figure 2. The calculated perimeters of ellipse in terms of major diameter and eccentricity, for a subset of training data using arc length integral.

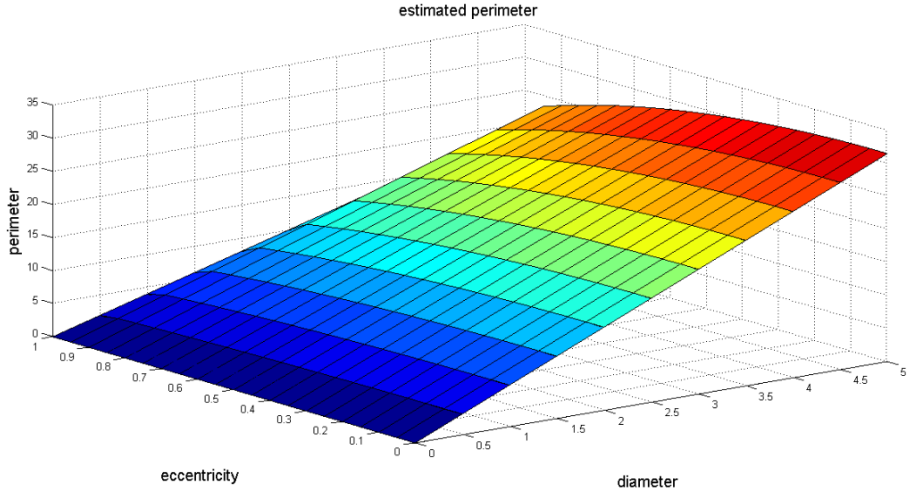

Figure 3. The calculated perimeters of ellipse in terms of major diameter and eccentricity, for a subset of training data using RBF neural network.

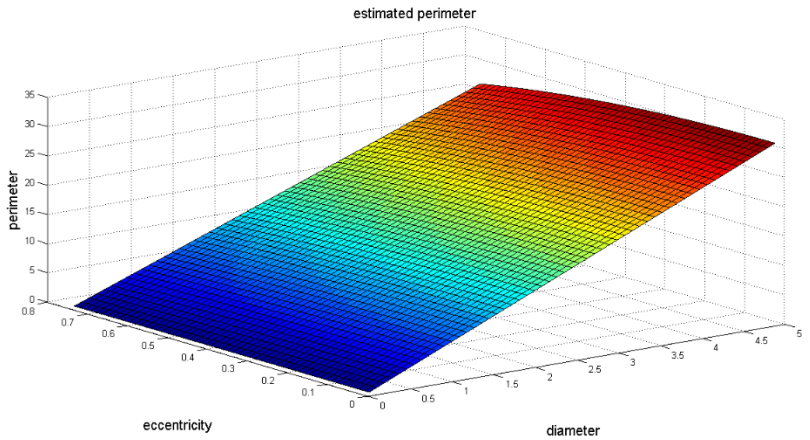

Figure 4. The calculated perimeters of ellipse in terms of major diameter and eccentricity, for a subset of test data using arc length integral.

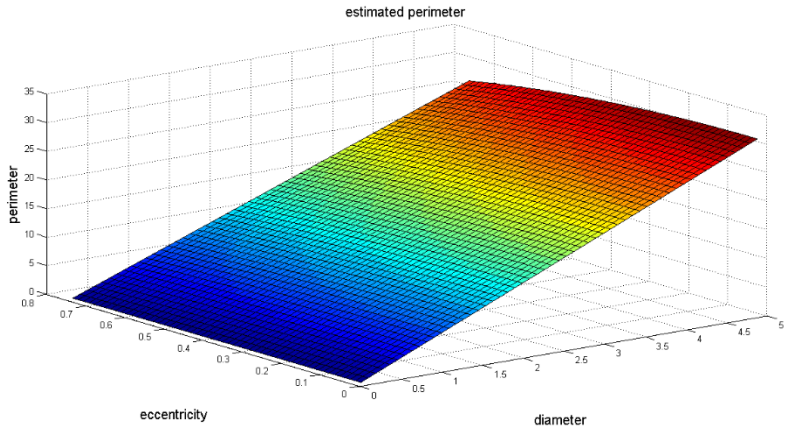

Figure 5. The calculated perimeters of ellipse in terms of major diameter and eccentricity, for a subset of test data using RBF neural network.

As it was predictable from training and generalization errors, the figures obtained from numerical solution of integral and neural network regarding training and test data are identical.

2.3 Training RBF neural network using gradient descent method

In this section like the last one, the goal is to calculate the weights of neural network. In this method the outputs of basic functions are normalized but the biased term is not considered and the other difference is in the method of training the network & calculating the weights. Using gradient descent for calculating the weights of the RBF, neural network is described by [3]. In the next steps, we explain how to use these algorithms and measurement.

As said before, Eq.9 shows the learning error of neural network and Eq.4 shows the output of network for each observation. The general equation for calculating learning error is obtained from Eq.9 & 4 as follows:

$$E_l = \frac{1}{2n}\sum_{i=1}^{n}(z_i - \sum_{j=1}^{k} w_j g_j)^2 \quad (11)$$

Now by placing Eq.2 for radial basis functions g in Eq.11, we have:

$$E_l = \frac{1}{2n}\sum_{i=1}^{n}\left(z_i - \sum_{j=1}^{k} w_j\, e^{-\frac{1}{2}(x_i-c_j)^T\Sigma^{-1}(x_i-c_j)}\right)^2,\ j = 1,\dots,k\ ,i = 1,\dots,n \quad (12)$$

By differentiating Eq.12 with respect to w_r, We'll have $r = 1,2,\dots,k$. So:

$$\frac{\partial E_l}{\partial w_r} = -\frac{2}{2n}\sum_{i=1}^{n}\left(\left(z_i - \sum_{j=1}^{k} w_j\, e^{-\frac{1}{2}(x_i-c_j)^T\Sigma^{-1}(x_i-c_j)}\right)\times e^{-\frac{1}{2}(x_i-c_r)^T\Sigma^{-1}(x_i-c_r)}\right) \quad (13)$$

Now we define Eq.14:

$$\Delta w_r = -\eta\frac{\partial E_l}{\partial w_r} \quad (14)$$

Were E_l is total learning error and η is training rate. Eventually, the updated equation for calculating weights w_r of network for $r = 1,2,\dots,k$ would be:

$$w_r = w_r + \Delta w_r \quad (15)$$

It should be noted that since this algorithm is a numeric solution for calculating the network weights, we have to consider an initial value for these weights. For this reason, this initial value is selected from uniform distribution between -1 and 1.

2.3.1 Gradient descent results

Figure 6 shows the changes in training and generalization error for different values of η with respect to 2000 are iterations:

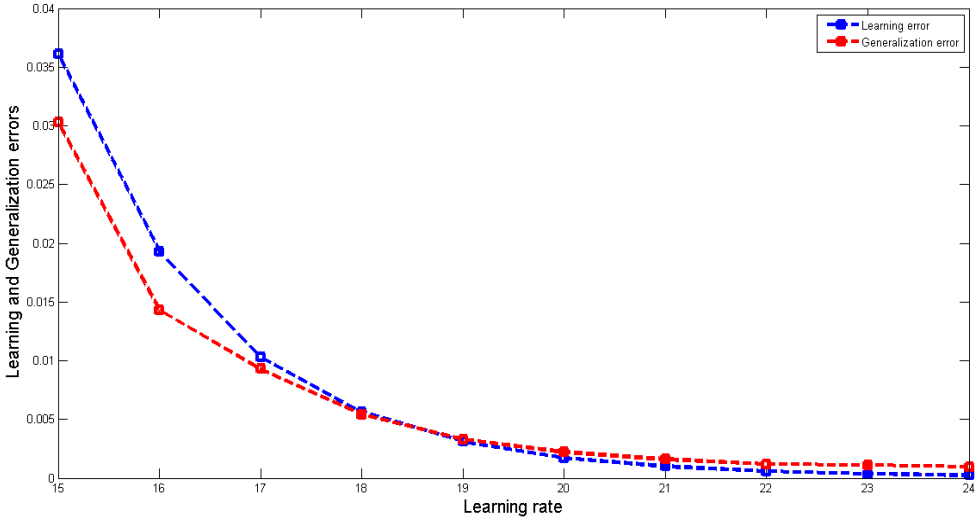

Figure 6. Learning and Generalization error in the gradient descent algorithm for different values of η

As you can see for η =24 and 2000 iterations the learning and generalization errors are 0.000246 and 0.000954, respectively.

We implemented this algorithm considering different values for network weights and the result of each single calculation was highly identical to the rest. It shows that this algorithm could find the absolute minimum of cost function very well.

3. COMPARISON AND CONCLUSION

Study [1] used a polynomial calculated by Lagrange's method to find a relationship for estimating perimeter of ellipse. It fitted the estimations obtained from Euler's method and showed that if the degree of polynomial is higher than 7 then it can accurately estimate the perimeter. Although this method improves the accuracy of estimation compared to previous methods, but it had some errors. First, they considered estimating model to be polynomial, while we know this is not necessarily the best model. Second, they fitted the model on the perimeters calculated by Euler's method and it's clear that Euler's method itself has error. The other problem was that this model was only a function of eccentricity and there for changing the radius would change the model. So they couldn't offer an integrated model to estimate ellipse perimeter based on a unified major diameter and ellipse.

The method proposed here partly solves these problems, because we've used a non-parametric method for fitting the curve, which means we didn't use any basic premise on the estimating model. We also did our fitting based on estimations from the numerical solution of elliptic integral of the second type; these are considerably more accurate than Euler's method. The last but not the least, this network acts as a function of major diameter and eccentricity, so we've proposed a function that doesn't change in response to changes in parameters of the ellipse and needs to be trained only once.

In this study, we trained the network only for the values of major diameter between 0-10, because increasing the major diameter would increase the number of $x = \begin{bmatrix} a \\ e \end{bmatrix}$ pairs drastically. This excessive increase in training data could cause some computational problems in our computer. Since this network only needs to be trained once, we may train it by a super computer for a wide range of axes and eccentricities, once and for all. So this problem could be solved.

In this study, the network was trained with two different approaches. We saw that when using the gradient decent method, increasing the rate of training in the range of 15-24 for an iteration rate of 2000 times, the training and generalization error would decrease slowly. The trend of these changes are shown in Table 1. Table 1 shows the training and the generalization errors are calculated for four decimal number for different values of η, considering 2000 iterations for the algorithm.

Table 1: Learning and Generalization errors for different values of in gradient decent method.

η	15	16	17	18	19	20	21	22	23	24
E_l	0.0361	0.0193	0.0103	0.0056	0.0031	0.0017	0.0009	0.0005	0.0003	0.0002
E_g	0.0303	0.0143	0.0093	0.0054	0.0033	0.0022	0.0016	0.0012	0.0011	0.0009

To better compare the two methods of gradient decent and least squares, we compared the results from training and generalization errors for $\eta = 50$. The results are shown in Table 2:

Table 2: Learning and Generalization errors from gradient decent and least squares methods.

	Least squares	Gradient descent($\eta = 50$)
E_l	$\cong 10^{-22}$	$\cong 10^{-7}$
E_g	0.000769	0.000769

As we can see, for η = 50 and 2000 iterations, generalization error is the same for both methods. Learning error of gradient decent method would not get any lower. As far as learning error is considered, the least squares method has better performance, although the error in both methods is inconsiderable. At the first glance, the comparison of these two methods may imply that the outcome of the least squares method was better, but it should be considered that the method of least squares needs matrix inverting; a process that gets harder by increasing the data. The gradient descent method doesn't have such restrictions and is preferred for the problems with large data sets.

RECEIVED: AUGUST, 2017
REVISED: DECEMBER, 2017

REFERENCES

[1] AMIRI, A. M., ASHRAFI, H. R., BEIRANVAND, P., MEHDIPOUR, S., and HOSSEINI, M. (2016): ESTIMATING THE SECOND TYPE ELLIPTIC INTEGRAL AND OVAL CIRCUMFERENCE BY INTERPOLATION METHOD, **Far East Journal of Mathematical Sciences (FJMS)**. 100, 2175–2182.

[2] ALMKVIST, G., and BERNDT, B. (1988): Gauss, Landen, Ramanujan, the Arithmetic-Geometric Mean, Ellipses, π, and the Ladies Diary. **The American Mathematical Monthly**, 95, 585-608.

[3] BISHOP, C. M., and ROACH, C. M. (1992): Fast curve fitting using neural networks. **Review of Scientific Instruments**. 63, 4450–4456.

[4] CĂLIN, E. (2010): **Multidimensional Function Approximation Using Neural Networks**. "Petru Maior" University of Targu Mures, Romania.

[5] DANON Y. and M. J. EMBRECHTS.(1992): Least squares fitting using artificial neural networks. **in Intelligent Engineering Systems through Artificial Neural Networks**. vol.2, NY: ASME Press.

[6] GOPALAKRISHNAN, S., and BOURBAKIS, N. (2016): Curve Fitting Methods. **International Journal of Monitoring and Surveillance Technologies Research**, 4, 33–53.

[7] ORR MARK .J.L.(1996): **Introduction to Radial Basis Function Networks, Center for Cognitive Sience**, Edinburgh University, EH9LW, Scotland, UK.

[8] SÝKORA, S., R. SANZIO and C. PRIMO.(2005): **On the circumference of ellipses and novel approximations for the complete elliptic integral E(x)**. http://www.ebyte.it/library/docs/math05a/EllipseCircumference05.html

[9] ZAFARY, S.(2006): **A Single Term Formula for Approximating the Circumference of an Ellipse**. http://mathforum.org/kb/servlet/JiveServlet/download/128-1921864-6683056-551607/circumference%20of%20an%20ellipse.pdf

[10] ZARITA Z. and ONG PAULINE.(2007). **Function approximation using artificial neural networks**. In Proceedings of the 12th WSEAS International Conference on Applied Mathematics (MATH'07), Metin Demiralp, Constantin Udriste, Gabriella Bognar, Ritu Soni, and Hamed Nassar (Eds.). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 140-145.