Testing Artificial Neural Network (ANN) for Spatial Interpolation

Veronika Nevtipilova1, Justyna Pastwa1, Mukesh Singh Boori1,2 and Vit Vozenilek1*

1Dept. of Geo-informatics, Palacky University Olomouc, Czech Republic
2Dept. of Civil Engineering, JECRC University Jaipur, Rajasthan, India

Abstract

The aim of this research is to test Artificial Neural Network (ANN) package in GRASS 6.4 software for spatial interpolation and to compare it with common interpolation techniques IDW and ordinary kriging. The entire package uses multi-layer perceptron (MLP) model trained with the back propagation algorithm. Evaluation methods were based mainly on RMSE. All the tests were done on artificial data created in R Project software; which simulated three surfaces with different characteristics. In order to find the best configuration for the multilayer perceptron many different settings of network were tested (test-and-trial method). The number of neurons in hidden layers was the main tested parameter. Results indicate that MLP model in the ANN module implemented in GRASS can be used for spatial interpolation purposes. However the resulting RMSE was higher than RMSE from IDW and ordinary kriging method and time consuming. When compared neural network packages in GRASS GIS and R Project; it is better to use the packages in R Project. Training of MLP was faster in this case and results were the same or slightly better.

Keywords: ANN; IDW; Kriging; Interpolation; GRASS GIS; Nnet; Neuralnet

Introduction

Spatial interpolation is quite frequently used method for working with spatial data. Currently there are many interpolation methods, each of which has its own application. The level of accuracy of these methods is limited, and therefore the spatial interpolation looking for new techniques and methods. One of these techniques is the use of neural networks. The principle of neural networks is known for a very long time, the first artificial neuron was constructed in 1943 [1]. However their use in the field of geo-informatics only started recently. From the available literature, it is evident that neural networks are using the spatial interpolation with good results, comparable with other interpolation methods, in some cases even better [2-4]. Using neural networks for spatial interpolation is not yet very widespread issue interpolation methods, in some cases even better [2-4]. Using neural networks for spatial interpolation is not yet very widespread issue among regular users of GIS, since most of the available GIS software is not implemented itself a neural network models. GRASS [5] GIS software is one of the few for which there is a module to work with neural networks, namely the multi-layer perceptron model (MLP). This work is engaged in testing of this module and its comparison with two of the mostly used in spatial analysis interpolation methods: IDW and simple kriging.

The aim of this research is to use MLP model for normal interpolation and determine whether the quality of the resulting interpolation comparable with other conventional methods. In this paper, first we mention objectives of the research work. Second summarizes the methods used and work progress. In third part we briefly describe the theoretical basis used in interpolation methods - that is, neural networks, IDW and kriging. This part also deals with the implementation of neural networks in two software; used in this work. Briefly assesses and compare examples from the literature on the use of neural networks for spatial interpolation. Fourth describe - data creation, selection of best MLP parameters, process, steps of used commands and settings for custom interpolation in the GRASS GIS software and R Project. The last part summarized results. This part present and evaluate outcomes of previously used interpolation methods; compare and evaluate their quality. Also compare MLP method in GRASS 6.4 software and R Project.

Methodology and Data Processing

Spatial interpolation: is a process in which the known values of a certain phenomenon estimating the value in places where not measured.

Neural Network: Biological neuron and its simplified features serve as the basic unit of artificial neural networks. These were created as a simplified mathematical model simulating operation of the human brain [6]. It consist of n inputs creating vector x=(x1, ..., xn). Each input is multiplied by the corresponding weight parameter, which can be positive or negative. Another input neuron x0=1 is rated by weight x0, which represent the bias [1]. A simple neuron model shown in Figure 1.

The sum of all weighted inputs yIN indicates the internal potential of the neuron:

\[ y_{in} = \sum_{i=1}^{n} w_i x_i \]  

(1)

The weighted sum is passed through a neuron activation function y = f(y_IN) and produce the final output of the neuron. Which turn can become stimulus for neurons in the next neural network layer. The simplest type of activation function is threshold function:

*Corresponding author: Vit Vozenilek, Department of Geo-informatics, Palacky University Olomouc, Czech Republic, Tel: 420 585 631 111; E-mail: vit.vozenilek@upol.cz

Received January 20, 2014; Accepted February 17, 2014; Published February 21, 2014


Copyright: © 2014 Nevtipilova V, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
The interconnection of neurons creates a neural network. The output from one neuron is the input of other neurons [1]. Neurons in the network are organized into layers (Figure 2). Each neuron in the input layer is connected to all the neurons of the above layer. Each neuron in the layer is connected to all the neurons of the above layer (following) layer.

MLP are the most popular type of neural networks used recently. They belong to a general class of structures so-called feed forward neural networks and present a very basic type of neural network.

Back-propagation algorithm: The MLP is trained using back-propagation algorithm. This is a supervised learning and it takes two stages. First is the feed-forward propagation. The second phase is back-propagation. For each neuron in input layer is calculated gradient of the error function at each iteration step, which is the part of error transmitted to the left of the unit (to previous layer) according to formula:

$$
δ_{in} = \sum_{k=0}^{m} δ_k w_{jk}
$$

With this sum and the derivative of the internal potential $z_{in}$ of neuron $z$ is calculated partial error $δ_j$:

$$
δ_j = δ_{in} f'(y - in_j)
$$

Calculated $δ_j$ is used to adjust the weights between input and internal layer of the network: $Δw_j = αδ_j x_i$, where $α$ is the coefficient of learning and $x_i$ is the input value of the network. With $δ_j$ are adjusted weights between inner and output layer of the network: $Δw_{jk} = αδ_j z_j$, where $z_j$ is the value of the output from neuron $Z_j$. Next, the weights on connections between neurons are updated. New weight is denoted as $n$ and old as $s$. Than $v_{ij}(n) = v_{ij}(s) + Δv_{ij}$ and $w_{jk}(n) = w_{jk}(s) + Δw_{jk}$

Topology of neural networks: First several MLP neural networks with different number of neurons in hidden layers were created. Then trained on training dataset using back-propagation algorithm and tested on smaller (testing) dataset, which was not used while training. The mean square error (RMSE) was calculated from trained MLP on test data and few of the best MLP configurations were selected for later calculations.

Custom Interpolation: Using three distinct datasets representing terrain with different characteristics, the interpolation was performed in software’s GRASS 6.4 and R. The interpolation method IDW, kriging and MLP were used. Interpolation was done for each simulated terrain. In software GRASS 6.4; 12 raster’s were created while analysis (three from MLP trained on raster data, six from MLP trained on vector data and three from IDW); in R software; 12 raster’s were created (six by MLP from each package (nnet and neuralnet), six by methods IDW and simple kriging).

Evaluation results of interpolation: For all interpolation results was calculated RMSE according to formula:

$$
RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i - z_{odi})^2}
$$

Where $n$ is number of points, $z_i$ is the value of point $i$; $z_{odi}$ is the real value in this point. In order to visually compare applied methods, results were subtracted and the difference between them was calculated. Each time raster created by IDW and simple kriging was subtracted from raster generated MLP.

**Implementation in GIS**

**GRASS 6.4**

Works with feed forward neural networks; trained with the back-propagation algorithm using 5 scripts written in Python programming language invoking Fast Artificial Neural Network (FANN) library [9]. This script works with raster data. Scripts and their functions are: ann. create creates ANN define file; ann.info displays information about defined ANN; ann.data.rast prepare the learning datasets using raster layer data. ann. learn perform learning and ann.run.rast run the trained ANN to create output raster layer (Netzel) [9].

**R project**

The R Project software work with feed forward ANN by nnet and neuralnet packages.

The nnet package [10] allow for training feed forward networks by back-propagation algorithm. This network has only one hidden
Figure 3: IDW [12].

IDW method has characteristic undesirable phenomena that arise as a result of the calculation of average weights. The method of calculation allows the creation of new values only in the range of existing values. If interpolated terrain directed by peaks and valleys, then peaks will appear as depressions and vice versa [12]. Another undesirable phenomenon is the formation of concentric contour lines around points, with the initial value called bull’s eyes [13]. IDW method is readily available in most GIS software such as ArcGIS, QGIS, GRASS GIS, IDRISI, and R (Figure 3).

**Ordinary kriging**

The basic idea behind kriging is to find certain general characteristics from measured value and applying these properties, when calculating the unknown value. From these properties the most important is smoothness. Theory says that the values in close location to searched point are more similar than in more distant points. The difference of values z between two points is calculated as:

\[(z(x)) - (z(xi))^2\]  \hspace{1cm} (10)

With increasing distance between points, it is probable that value z will be also increasing until certain distance and then it will not change [12]. Figure 4 is an example of semivariogram that illustrates how the differences in pairs of points from the measured values change. Sill indicates the maximum value the semivariogram, attains range describes the lag at which the semivariogram reaches the sill. The value of semivariance is never 0, even zero distance. Therefore the parameter nugget indicates the value of difference between two points in the same site, or at a very small distance. Crosses indicate the difference between selected pairs of points; wheels are averages of these values at a certain distance [12]. Ordinary kriging is standard and frequently used version of kriging. Value z in point \(s_j\) is calculated by following formula:

\[z(s_j) = \sum_{i=0}^{n} W_i (s_j) z(s_i) = \lambda_0 z\]  \hspace{1cm} (11)

Where \(n\) is the number of points used in calculations, \(w_i\) is a weight for point \(z(s_j)\) and \(z(s_i)\) is a point with known value \(z\). In short, therefore \(\lambda_i\) is the vector of weight \(w_i\) and \(z\) is the vector of \(n\) points of known values. Weight \(w_i\) are calculated using the system of equations [15]. As IDW, simple kriging interpolation method is known and available in the popular GIS software.

Figure 4: Example semivariogram [12].
Testing Interpolation Methods

Creating data

To provide interpolation using MLP. Three artificial datasets were created in R Project. The dataset simulate different roughness of the terrain. The model with higher roughness was signed as 1, with less roughness as 3. The datasets were randomly generated using function GRF (Gaussian random fields) from package geoR that create points and randomly assign values to them. These values are influenced by other parameters of the function.

 grf(pocetBodu, grid="reg", cov.pars=c(sill, range), nug=nugget, cov.model=cov Model, aniso.pars=c(anisotropy Direction, anisotropy Ratio), xlims = xlims, ylims = ylims)

Parameter grid="reg" indicate that points are generated in a regular grid. Parameter cov.model determines the type of variogram, here spherical. Values xlim and ylim were set in interval 0 - 1. There were 1024 points generated in a regular grid for each dataset. These points had three attributes: coordinates x and y from range 0 – 1 and value z which represented elevation. Parameters value describing roughness, used while creation of datasets is given in Table 1.

Figure 5 is shown in the distribution of points in different datasets. Larger diameter wheels indicate higher z. Training data consist of 724 randomly selected points and was used for learning MLP. Test data contained remaining 300 points and were used to calculate RMSE.

Selecting optimal configuration of MLP using nnet package

The configuration of MLP was determined using test-and-trial method. To make the work more effective, a script gradually creating datasets for all surface topography was created. For each roughness of terrain 10 datasets were created and these datasets were divided into training and testing part. In next step 10 MLP was learned using training data from each existing datasets. Later, for all testing sets RMSE was calculated. This procedure was carried out a total of fifteen times, each time through the varied number of neurons in hidden layer in the range 15 to 30. For each roughness RMSE was calculated. In the last step average RMSE was calculated for all roughness for each MLP configuration. Results are listed in Table 2. Best setting is found MLP with 28 neurons in the hidden layer. Selecting optimal configuration of MLP using neuralnet package: this is similar to nnet package. The difference is that neuralnet package allows trained networks with more hidden layers. Gradually networks were tested with one to four hidden layers. The number of neurons in the first hidden layer is always moved in the interval of 15 to 30. The number of neurons in other hidden layers has been fixed and was selected from a number of 5, 10, 15, 20, and 25. Average RMSE test set was again calculated for all datasets. 24, 15, 10, 5 number of neurons were selected as the best network with four hidden layers. Results of testing networks are in Table 3.

Selecting optimal configuration of MLP using GRASS 6.4

The module ANN allows training network with multiple hidden layers. As well as in previous methods, an optimal configuration for MLP was found using test-and-trial method. The network with three hidden layers and the number of neurons 32, 38, 27 was selected as most successful MLP setting.

IDW

This method is implementer in R Project software in several packages; in this work gstat package was used with the function idw.

 idw_result <- idw (z~x+y=train.set locations, New Data=grid, nmax=18, idp=1.0)

Function idw use parameter formula to distinguish coordinates and values which will interpolated. Parameter locations define data used for interpolation and newdata parameter specifies the new coordinates. Number of points that are used in interpolation is set with

<table>
<thead>
<tr>
<th>Roughness</th>
<th>Sill</th>
<th>Range</th>
<th>Nugget</th>
<th>Anisotropy ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.12</td>
<td>0.3</td>
<td>0.00001</td>
<td>0.8</td>
</tr>
<tr>
<td>2</td>
<td>0.08</td>
<td>0.5</td>
<td>0.00001</td>
<td>0.8</td>
</tr>
<tr>
<td>3</td>
<td>0.01</td>
<td>1.2</td>
<td>0.00001</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Table 1: Parameter values for all surface contours.
Calculation of RMSE and visual comparison

To assess the quality RMSE was calculated and compare tested interpolation methods. Assessment was carried out in R Project.

When estimating RMSE for surfaces interpolated in GRASS 6.4 from raster data, the vector layer of randomly located points was created by command v.random. To this points the value of original and interpolated raster were added. After these preparations, the data were exported to SCV format using command v.out.ogr. In the next step, the CSV file was imported to R Project where elevation values z were normalized by formula 12 and RMSE was calculated. When estimating RMSE for surfaces interpolated in GRASS 6.4 from vector data, at first the points from test.set were imported to GRASS 6.4. Using command v.what.rast the corresponding values from interpolated raster were added. Next, the file with points were imported to R Project, than the normalized values were transformed to real ones using formula 13 and RMSE was calculated. The steps while analyzing outputs from IDW method was analogous but the values after interpolation was real and there was not need to convert it. The calculation of RMSE was carried out in R Project in the same way for each interpolation methods. The RMSE was calculated for interpolation results coming from test data. The interpolation output from MLPs was transformed from normalized to real values according to formula 13. Next, the original values of elevation were added and finally the RMSE was calculated.

Results

Main results are following:

- Assessment of the quality of interpolation using ANN in GRASS GIS 6.4-svn,
- Comparison of ANN module and interpolation method IDW and kriging,
- Comparison of interpolation using neural networks in GRASS GIS and R Project,

Evaluation of the interpolation quality

first interpolation with the neural network trained on raster data was evaluated. RMSE values with decreasing segmentation data (falling range of values z) decreased. Learning time was longer with the increasing number of the vector points. Learning time was shorter with lower segmentation data. The less iterations were required to train the networks when more vector points were used. Table 4 summarizes data about training the neural network.

The range of values z in the interpolated grid is lower than the range of values in the original grid (Table 5). This could be due to poor distribution of random vector points. RMSE value in this case was 0.0646. In case of segmentation 1, neural network has missed extreme values. In segmentation 2, difference in the range is smaller; which means the random vector points were probably better distributed. However the extreme values are also omitted. The value of RMSE is 0.0427. In case of segmentation 3, network behaviour is similar to the previous two cases and the value of RMSE is 0.0085. The value of RMSE was then expressed in percentage according to the range of value z. The difference between the values of RMSE is only 1%; for each segmentation (Table 5).

Interpolation quality was further evaluated by visual comparison of resulting rasters. Neural network from ANN module trained on raster data was able to adapt quite well and resulting grid was very similar to the original grid. Figure 6 shows the original and interpolated grid for segmentation 1.
Comparison of ANN, IDW and kriging method

The input data selected for input neurons is same as used in IDW and ordinart kriging. The criterion for comparison was RMSE value, time demand and user friendliness.

Comparison by RMSE: Figure 8a compares the RMSE values of resulting raster’s. The network used in this comparison was the one with number of 38, 32, 27 neurons. RMSE values of raster generated by the network were in case of segmentation 1 and 2, higher than value of other two methods. This was due to the wrong setting of network parameters; may be network probably got over-trained. Table 8 records the RMSE values with accuracy of four decimal places.

The RMSE value of raster’s interpolated by neural networks were higher than the values of raster created by IDW and kriging. Although it was assumed that neural networks would have better results. There are several reasons: Despite the testing of neural networks settings. It is possible that inappropriate parameters were chosen to fit the nature of data and networks were trained poorly. The other reason for worse results of the neural networks might be insufficient number of input parameters – only two were used. Figure 9 shows comparison of the RMSE value for all methods in R Project. RMSE values of raster’s by neural networks from both packages were higher than values of raster’s by IDW and kriging methods.

RMSE values for the nnet package for segmentation 2 and 3 are
most similar to the RMSE value for the other methods. Table 9 shows the RMSE values with an accuracy of four decimal places. Figure 10 shows the resulting raster interpolated with neural networks, IDW and kriging in GRASS GIS for the segmentation 1.

Figure 11 show the differential bitmap for segmentation 1. Figures 11a and 11b show the difference in z values between raster’s created by neural network (20, 25, and 17) and other methods. The maximum value difference between the network and IDW were -0.320700 and 0.326500 and average value was 0.0030340. Maximal value of difference between the network and kriging were -0.272800 and 0.305600. The average value difference was -0.001977. Raster created by IDW and kriging methods has higher values. The differences between this network, IDW and kriging method is lower than in the first case.

Comparison by time-consuming: The time required to perform an interpolation is shown in Table 10. These values in case of neural networks include the time needed to train the network and time required to perform the calculation. In case of other methods, only time required to perform the computation is shown. A neural network is time consuming due to the long training time. If wrong parameters were chosen; then training took a very long time (Table 10). With the better parameters, training time was distinctively shorter. With decreasing segmentation of the data the training time decreased as

<table>
<thead>
<tr>
<th>Original data range</th>
<th>Interpolated grid</th>
<th>RMSE %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Articulation 1</td>
<td>-0.7396643 - 1.090838</td>
<td>-0.6768178 to 0.9450067</td>
</tr>
<tr>
<td>Articulation 2</td>
<td>-0.5203077 - 0.893016</td>
<td>-0.6193843 to 0.6952054</td>
</tr>
<tr>
<td>Articulation 3</td>
<td>-0.2012786 - 0.141199</td>
<td>-0.1532334 to 0.0899525</td>
</tr>
</tbody>
</table>

Table 7: Range of values z original and interpolated data on (20 25 17) network and comparison of RMSE for all segmentations.

<table>
<thead>
<tr>
<th>Articulation 1</th>
<th>n network(32, 38, 27)</th>
<th>IDW</th>
<th>Kriging</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2221</td>
<td>0.1398</td>
<td>0.1240</td>
<td></td>
</tr>
<tr>
<td>0.1285</td>
<td>0.0874</td>
<td>0.0770</td>
<td></td>
</tr>
<tr>
<td>0.0172</td>
<td>0.0177</td>
<td>0.0158</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Articulation 2</th>
<th>n network(20, 25, 17)</th>
<th>IDW</th>
<th>Kriging</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1407</td>
<td>0.1398</td>
<td>0.1240</td>
<td></td>
</tr>
<tr>
<td>0.0962</td>
<td>0.0874</td>
<td>0.0770</td>
<td></td>
</tr>
<tr>
<td>0.0252</td>
<td>0.0177</td>
<td>0.0158</td>
<td></td>
</tr>
</tbody>
</table>

Table 8: RMSE values for all segmentation for GRASS GIS.

<table>
<thead>
<tr>
<th>Articulation 1</th>
<th>n network(32, 38, 27)</th>
<th>IDW</th>
<th>Kriging</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1418</td>
<td>0.1427</td>
<td>0.1325</td>
<td>0.1240</td>
</tr>
<tr>
<td>0.0843</td>
<td>0.1002</td>
<td>0.0828</td>
<td>0.0770</td>
</tr>
<tr>
<td>0.0177</td>
<td>0.0264</td>
<td>0.0176</td>
<td>0.0158</td>
</tr>
</tbody>
</table>

Table 9: RMSE values for all the contours of the R Project.

<table>
<thead>
<tr>
<th>Articulation 1</th>
<th>n network(32, 38, 27)</th>
<th>IDW</th>
<th>Kriging</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2221</td>
<td>0.1398</td>
<td>0.1240</td>
<td></td>
</tr>
<tr>
<td>0.1285</td>
<td>0.0874</td>
<td>0.0770</td>
<td></td>
</tr>
<tr>
<td>0.0172</td>
<td>0.0177</td>
<td>0.0158</td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Time consumed for interpolation.
Comparisons were carried out in several GRASS GIS and R Project: knowledge of neural network. A brief manual is not too useful. Use of ANN module is difficult for inexperienced users due to lack of interface, but manual is not included. It has to be opened separately. The ANN module can be operated both in command line and graphical interface. When working in the graphical interface a manual is available. It describes each setting that, what is the affect when used and how to work with availability methods to help and comprehensibility of used methods. IDW method and script v.surf.idw is part of the main installation of GRASS GIS. It can be used via the command line or the graphical interface. When working in the graphical interface a manual is available. It describes each setting that, what is the affect when used and in the last a short theoretical summary. Kriging method does not exist as a module in GRASS GIS. It can be used by connecting the GRASS GIS with R Project program. Only the command line is available for the work with raster data only (in present version), but the interpolation method was the longest one, mainly because of the time needed to train the networks.

Comparison by the user friendliness: This review summarizes the work with availability methods to help and comprehensibility of used methods. IDW method and script v.surf.idw is part of the main installation of GRASS GIS. It can be used via the command line or the graphical interface. When working in the graphical interface a manual is available. It describes each setting that, what is the affect when used and in the last a short theoretical summary. Kriging method does not exist as a module in GRASS GIS. It can be used by connecting the GRASS GIS with R Project program. Only the command line is available for the user. Manual and instructions how to work with the kriging method is available in the R Project as well. Calculation time was very short; once the networks were properly trained. Segmentation of the data did not affect the speed of calculation; in any method. The longest computation time had the kriging. The fastest interpolation method was IDW. Interpolation with the neural network was the longest one, mainly because of the time needed to train the networks.

Evaluation and testing results shown that neural networks in GRASS GIS can be used for spatial interpolation, but it’s (MPL) not better than IDW and kriging method. The disadvantage of ANN module is work with raster data only (in present version), but the interpolation from vector data. Without the user intervention (in this research work using a custom script that transformed vector data in the desired format) the ANN module is not very useful for normal interpolation. After comparing the neural networks from both software’s for the purpose of normal interpolation the R Project is better than GRASS GIS, although neither network in the R Project had better results than the methods IDW and kriging. The use of multilayer perceptron for spatial interpolation is an interesting option to classical methods. But it’s requiring more knowledge of theory from the user and time consuming. The results are often uncertain and the training of MLP

<table>
<thead>
<tr>
<th>n network (32, 38, 27)</th>
<th>n network (20, 25, 17)</th>
<th>kriging</th>
<th>IDW</th>
</tr>
</thead>
<tbody>
<tr>
<td>articulation 1</td>
<td>42</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>articulation 2</td>
<td>47</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>articulation 3</td>
<td>14</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 10: Time required to performing interpolation (in minutes).

<table>
<thead>
<tr>
<th>Articulation</th>
<th>n network (32, 38, 27)</th>
<th>n network (20, 25, 17)</th>
<th>nnet</th>
<th>neuralnet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Articulation 1</td>
<td>0.1407</td>
<td>0.2221</td>
<td>0.1418</td>
<td>0.1427</td>
</tr>
<tr>
<td>Articulation 2</td>
<td>0.0982</td>
<td>0.1285</td>
<td>0.0843</td>
<td>0.1002</td>
</tr>
<tr>
<td>Articulation 3</td>
<td>0.0252</td>
<td>0.0172</td>
<td>0.0177</td>
<td>0.0264</td>
</tr>
</tbody>
</table>

Table 11: RMSE values for all segmentation for GRASS GIS and R Project.

The training speed of the networks in ANN module and R Project depends primarily on the number of neurons in the hidden layers, the segmentation of input data and size of training dataset. In case of the ANN module it also depends on appropriate parameter settings.

However the neural network packages in R Project train faster than the networks in the ANN module. Figure 13 shows the resulting bitmaps of the neural network for segmentation 1. The resulting raster in partial Figures 13a, 13b and 13d are visually quite similar and RMSE values is not much differ. Resulting raster in sub Figure 13c differ significantly both in visually and RMSE value.

Conclusions

Evaluation and testing results shown that neural networks in GRASS GIS can be used for spatial interpolation, but it’s (MPL) not better than IDW and kriging method. The disadvantage of ANN module is work with raster data only (in present version), but the interpolation from vector data. Without the user intervention (in this research work using a custom script that transformed vector data in the desired format) the ANN module is not very useful for normal interpolation. After comparing the neural networks from both software’s for the purpose of normal interpolation the R Project is better than GRASS GIS, although neither network in the R Project had better results than the methods IDW and kriging. The use of multilayer perceptron for spatial interpolation is an interesting option to classical methods.

Figure 13: Comparison of interpolated grid for segmentation.

12 compare the values of RMSE for the two networks from the GRASS GIS, nnet and neuralnet package. For segmentation 1, values were almost equal, except the network (32, 38, and 27) of GRASS GIS, which was probably over-trained. This also happened in case of segmentation 2. Values of RMSE for other networks were again similar and best results were given by the nnet package. In case of the segmentation 3; values were again quite similar for (20, 25, and 17) network. Network of neuralnet package showed signs of over-training. The best results for this segmentation were given by the network (32, 28, and 27) from GRASS GIS. Used neural networks in both programs were chosen as one of the best possible for the available data. RMSE values for each segmentation is not much different. If the neural networks with different parameters were used. Results would have probably differed more or less. Table 11 is recorded RMSE values from Figure 12.
has to be repeated many times to reach satisfactory results. The ANN module is in its current form cannot yet be regarded as equivalent to the conventional methods; however future development of this module might make a difference.

References
16. Vitalij Chalupnik (2012) Biologicke algoritmy (4) - Neuronove site. ROOT.CZ.

Submit your next manuscript and get advantages of OMICS Group submissions

Unique features:
User friendly/feasible website-translation of your paper to 50 world’s leading languages
Audio Version of published paper
Digital articles to share and explore

Special features:
300 Open Access Journals
25,000 editorial team
21 days rapid review process
Quality and quick editorial, review and publication processing
Indexing at PubMed (partial), Scopus, ASCI, Index Copernicus and Google Scholar etc
Sharing Option: Social Networking Enabled
Authors, Reviewers and Editors rewarded with online Scientific Credits
Better discount for your subsequent articles
Submit your manuscript at: http://www.omicsonline.org/submission