SPATIAL PREDICTION OF RADIOACTIVITY USING GENERAL REGRESSION NEURAL NETWORK

Vadim Timonin

Nuclear Safety Institute (IBRAE) of RAS, Moscow, Russia
Correspondence to: vadim@ibrae.ac.ru

Elena Savelieva

Nuclear Safety Institute (IBRAE) of RAS, Moscow, Russia
Correspondence to: esav@ibrae.ac.ru

This work describes an application of General Regression Neural Network (GRNN) to spatial predictions of radioactivity. GRNN belongs to a class of neural networks widely used for mapping continuous functions. It is based on a non-parametric (kernel) Parzen-Rosenblatt density estimator. The kernel size is the only tuning parameter, and it allows the user to implement a GRNN in an automatic mode. An important advantage of the GRNN is its very simple and fast training procedure. The most important drawbacks are high smoothing and dependence on the spatial density of the monitoring data set. The current case study is performed on the SIC2004 data sets, and the results obtained here can be compared with those obtained by the other participants using other approaches.

1. INTRODUCTION

The SIC2004 exercise deals with testing different approaches to automatic spatial predictions of radioactive contamination. These approaches are considered as possible candidates for implementation into a system for monitoring regularly radioactivity levels of potentially dangerous objects. Such methods require on-line functioning, in other words: high reaction speed and absolute automation. Also, it is very important that the prediction be unique: weak (or even no) dependence of the results on the initial conditions of the training procedure. This is not the case for the best-known Neural Network – a MultiLayer Perceptron (MLP), which is a very powerful and flexible instrument for spatial predictions (Kanevsky 1995). So in the current study another kind of neural networks was used – General Regression Neural Network (GRNN). A case study with GRNN for monitoring environmental variables was already presented by Kanevsky (1998).

GRNN is a nonlinear interpolator based on a sound mathematical foundation. Its main advantage is its simplicity and fast training, being independent of the initial condition. The most important drawback is smoothing. It means that prediction always overestimates the minimum and always underestimates the maximum values. Thus, it is not a very good instrument for predicting extreme and outlier values, but it allows the user to detect their presence. Results from the second ("joker") data set confirm this point. And it seems to be enough for an automatic monitoring system if it is able to detect

that "something is wrong" in the study area, position the flux and provide a preliminary level of values. More detailed prediction of extremes can be made later (if needed) using other approaches and/or by including additional measurements.

Just as for all interpolation techniques, GRNN results depend on the spatial density of the monitoring data set. The more homogeneously the input samples are distributed, the better performs the GRNN. Dense samples clusters (or empty holes) may distort the efficiency of GRNN. The SIC2004 exercise samples (200 training points) are distributed rather homogeneously, so this feature of GRNN does not influence the good result.

The SIC2004 exercise included some prior information. A quick look at these data sets led to the conclusion that the GRNN could be applied in this exercise. Also prior information allowed us to define input parameters for the GRNN optimisation procedure. The detailed parameter tuning was performed on real exercise data in an automatic mode.

2. METHODOLOGY

2.1. KERNEL REGRESSION WITH NEURAL NETWORK: GENERAL REGRESSION NEURAL NETWORK

Consider a non-linear regression problem, described by a model whose observable output z_i in response to an input vector x_i is defined by

$$z_i = f(x_i) + \varepsilon_i \quad i = 1,2\ldots,N \tag{1}$$

where $f(x_i)$ is a "smooth curve", and ε_i is a random residual drawn from a white noise process of zero mean and variance σ^2. That is

$$E[\varepsilon_i] = 0 \quad \forall i \qquad and \qquad E[\varepsilon_i \varepsilon_j] = \begin{cases} \sigma^2 & i = j \\ 0 & otherwise \end{cases} \tag{2}$$

The problem is to reconstruct the underlying function, $f(x_i)$, given the training sample $\{(\boldsymbol{x_i}, \boldsymbol{z_i})\}_{i=1}^{N}$. As a reasonable estimate of an unknown regression function $f(x)$, one may take a mean of observations (i.e. values of the model output z) from a neighborhood of the point x. This approach is successful if the local average is confined to observations in a small neighborhood (i.e. receptive field) of the point x as observations corresponding to points away from x will generally have different mean values. More precisely, $f(x)$ is equal to the conditional mean of z given x (i.e., the regression of z on x). Using the formula for the expectation of a random variable, it can be written

$$f(x) = E[z \mid x] = \int_{-\infty}^{\infty} z f_Z(z \mid x) dz \tag{3}$$

where $f_Z(z|x)$ is the conditional probability density function (p.d.f.) of z, given x. From probability theory, we have

$$\boldsymbol{f_Z(z \mid x)} = \frac{\boldsymbol{f_{X,Z}(x,z)}}{\boldsymbol{f_X(x)}} \tag{4}$$

where $f_{X,,Z}(x,z)$ is the joint p.d.f. of x and z and $f_X(x)$ is the marginal p.d.f. of x. Hence, using (4) in (3), we obtain the following formula for the general regression function

$$f(x) = \frac{\int_{-\infty}^{\infty} z f_{X,Z}(x,z) dz}{f_X(x)} \tag{5}$$

Note that we can derive the marginal p.d.f. $f_X(x)$ by integrating the joint p.d.f. $f_{X,Z}(x,z)$ over dz:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Z}(x,z) dz \tag{6}$$

Our particular interest is in a situation where the joint p.d.f. $f_{X,Z}(x,z)$ is unknown. All that we have available is a set of training samplesb $\{(x_i, z_i)\}_{i=1}^{N}$. To estimate $f_{X,Z}(x,z)$ and $f_X(x)$ therefore, we may use a non-parametric estimator known as the *Parzen-Rosenblatt density estimator* (Rosenblatt 1956, 1970; Parzen 1962). The basis for the formulation is a *kernel*, denoted by $K(x)$, which has properties similar to those associated with a p.d.f. (a continuous, bounded, and real function of x with the total volume under the surface equal to a unit). Whereas the kernel is a symmetric function around the origin where it attains its maximum value, this is not a requirement for a p.d.f.

Assuming that (x_i,z_i) are independent, identically distributed random vectors, we may formally define the *Parzen-Rosenblatt density estimator* of joint p.d.f. $f_{X,Z}(x,z)$ in the joint input-output space as (Parzen 1962; Rosenblatt 1956):

$$\hat{f}_{X,Z}(x,z) = \frac{1}{\sigma^{p+1} N} \sum_{i=1}^{N} K\left(\frac{x - x_i}{\sigma}\right) K\left(\frac{z - z_i}{\sigma}\right) \quad \textit{for} \quad x \in \Re^p \textit{ and } z \in \Re^1 \tag{7}$$

where the smoothing parameter σ is a positive number called *bandwidth* or simply *width*. It controls the size of the kernel $K(x)$. Assuming that the scaling for input and output variables is the same does not affect the result, as will be seen later. Using (6) or integrating $\hat{f}_{X,Z}(x,z)$ in (7) with respect to z, we can define the *Parzen-Rosenblatt density estimator* of p.d.f. $f_X(x)$ as

$$\hat{f}_X(x) = \frac{1}{\sigma^p N} \sum_{i=1}^{N} K\left(\frac{x - x_i}{\sigma}\right) \quad \textit{for} \quad x \in \Re^p \tag{8}$$

Note that an important property of this estimator is that it is *consistent* (i.e. asymptotically unbiased) in the sense that if $\sigma=\sigma(N)$ is chosen as a function of N such that

$$\lim_{N \to \infty} \sigma(N) = 0$$

then

$$\lim_{N \to \infty} E[\hat{f}_X(x)] = f_X(x)$$

For this latter equation to hold, x should be a point of continuity for $\hat{f}_X(x)$.

Thus, using (7) we can rewrite the numerator of (5) as

$$\int_{-\infty}^{\infty} z\hat{f}_{X,Z}(x,z)dz = \frac{1}{\sigma^{p+1}N}\sum_{i=1}^{N} K\left(\frac{x-x_i}{\sigma}\right)\int_{-\infty}^{\infty} zK\left(\frac{z-z_i}{\sigma}\right)dz \tag{9}$$

Changing the variable of integration by setting $a = (z - z_i) / \sigma$, and using the symmetric property of the kernel $K(\cdot)$, we obtain the result

$$\int_{-\infty}^{\infty} z\hat{f}_{X,Z}(x,z)dz = \frac{1}{\sigma^{p}N}\sum_{i=1}^{N} z_i K\left(\frac{x-x_i}{\sigma}\right) \tag{10}$$

The denominator of (5) is (8). Canceling common terms, we obtain the following estimate of the regression function $f(x)$:

$$\hat{f}(x) = \frac{\sum_{i=1}^{N} z_i K\left(\frac{x-x_i}{\sigma}\right)}{\sum_{i=1}^{N} K\left(\frac{x-x_i}{\sigma}\right)} \tag{11}$$

We can define the *normalized weighting function* as a function of x

$$W_j(x) = \frac{K\left(\frac{x-x_j}{\sigma}\right)}{\sum_{i=1}^{N} K\left(\frac{x-x_i}{\sigma}\right)} \quad j = 1,2\ldots,N \tag{12}$$

The denominator of (12) gives us the normalization property

$$\sum_{j=1}^{N} W_j(x) = 1 \quad \forall x \tag{13}$$

Now we can rewrite (13) in the simplified form

$$\hat{f}(x) = \sum_{j=1}^{N} W_j(x) z_j \tag{14}$$

In this form, equation (11) and property (13) describe $\hat{f}(x)$ as a *weighted average* of the z observations. The particular form of weighting function $W_j(x)$ given in (12) was originally proposed by Nadaraya (1964) and Watson (1964). Accordingly, the approximation function in (14) is often called the *Nadaraya-Watson Kernel Regression Estimator (NWKRE)*, or referring to the starting points (3) and (5), the *general regression estimator*.

There is a variety of possible kernel functions $K(\cdot)$. The most widely used among them is a multivariate Gaussian kernel:

$$K(x) = \prod_{i=1}^{p} K_i(x_i) = \frac{1}{(2\pi)^{p/2}} \exp\left(-\frac{\|x\|^2}{2} \right) \quad x = (x_1, \ldots, x_p) \tag{15}$$

with $\|x\|$ here and further meaning the norm defined in the p-size space. Centering the kernel on a data point x_i and scaling the width with the smoothing parameter σ, we obtain the following form

$$K\left(\frac{x - x_i}{\sigma} \right) = \frac{1}{(2\pi\sigma^2)^{p/2}} \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2} \right) \quad i = 1,2\ldots, N \tag{16}$$

With the kernel (16), the Nadaraya-Watson Kernel Regression Estimator looks like:

$$\hat{f}(x) = \frac{\sum_{i=1}^{N} y_i \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2} \right)}{\sum_{i=1}^{N} \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2} \right)} \tag{17}$$

where the denominator term represents the Parzen-Rosenblatt density estimator (without normalization) as a sum of N multivariate Gaussian distributions centered on the data points x_i (Specht 1991). To be unbiased this term should be divided by the normalizing constant $\left(2\pi\sigma^2\right)^{p/2} N$. The form (17) presents the Nadaraya-Watson Gaussian Kernel Regression Estimator.

The form (17) can be viewed as a normalized form of the Gaussian radial basis functions (Bishop 1995, p. 177) defined in the input space. Each basis function is centered on a data point, and the coefficients in the expansion are given by the target values z_i.

2.2. NWKRE AS A NEURAL NETWORK: GRNN

The NWKRE in the form (17) can be implemented in terms of a *neural network* (Specht 1991), and the corresponding neural network is called a *General Regression Neural Network* (GRNN). Figure 1 is a graphical representation of GRNN in terms of a neural network.

GRNN consists of four layers: *input*, *pattern*, *summation* and *output*. The input layer transfers an input signal – a p-dimensional vector x, into the next pattern layer. In the spatial mapping case p=2. The number of neurons (kernels) in the pattern layer is equal to the number of training samples (N), each neuron corresponding to a training sample. For an input vector all pattern layer neurons compute the Euclidian distance between the input vector and the corresponding neuron location. These distances are assumed as an activation function (a Gaussian kernel in our case) and transferred to the following layer. The summation layer consists of two neurons that calculate the nominator and denominator respectively in equation 17. Each of these neurons computes a weighted sum of the output from a previous layer. The weights correspond to the links between the neurons. Each link from the pattern layer neuron to the nominator neuron is a target value

associated with the corresponding neuron location. All links between pattern layer neurons and a denominator neuron are equal to a unit. The output layer neuron performs division operation and transfers the output value.

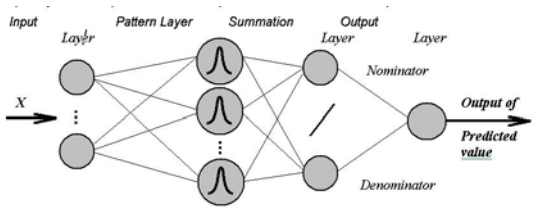

Figure 1
NWKRE as a general regression neural network (GRNN)

The only non-neural operation (division) in this architecture is that of the output layer. There is a general (though not universal) agreement that biological neurons are incapable of division. This is a point of criticism against interpreting the NWKRE as a neural network. Still, such an interpretation is very attractive because of the simple schematic illustration of the interpolation process (Figure 1) and the possibility to use machine learning terminology.

2.3. TRAINING OF THE GRNN

The only adaptive (free) parameter in the GRNN model with Gaussian kernel (equation 17) is σ, the width of the kernel. To demonstrate how a GRNN works, let us consider a simple one-dimensional problem (Figure 2). A simple sine function represents a true function of an underlying structure of a collection of sample data. In order to generate a training set, this sine function is sampled at a wide variety of points, and random noise is added. Furthermore, one wild point (outlier) is included. This true function and training samples are graphed in Figure 2a.

Now let us examine the effect produced by a variety of σ values, the smoothing parameter. In Figure 2b we see what happens when a very small value is used. The GRNN follows the training data closely, almost moving from point to point. If the data is known to be clear, the GRNN makes an excellent interpolation algorithm, analogous to the nearest neighbor method. Note, however, that this result will be acceptable only if the density of the training data is high enough. In other cases, an "*overfitting*" effect, which is well-known in neural networks, may appear and such solutions will not be a failure. So, since we know that the data are distorted by noise in most cases, straightforward interpolation is not what we want.

Figure 2c is obtained by using a moderately small smoothing parameter. Observe that the wild point (an outlier) succeeds in pulling the function a bit out of the line. If we are confident that any supposed wild point is actually a valid datum, then this is just what we want. But usually it is due to some kind of data distortion. A somewhat larger smoothing parameter gives us the result shown in Figure 2d, which looks ideal.

Finally, we must be aware of taking a good thing too far. Figure 2e illustrates the effect of too large a smoothing parameter. The global shape of the training set has swamped out all the details. Granted, we usually want the fine details, which are due to noise, to be eliminated.

Thus we can make the conclusion that choosing the value of the width parameter σ is vital for the GRNN model and problem-dependent.

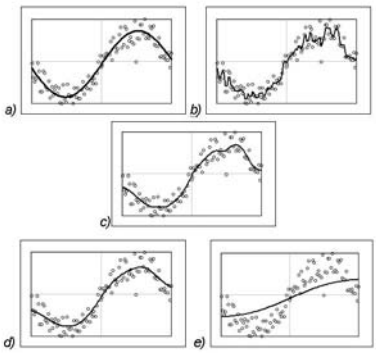

Figure 2
A simple problem to illustrate the influence of the parameter on the GRNN result: a) true function and noised training samples with an outlier; b) too small σ; c) perhaps a bit small σ; d) a perfect σ; e) too large σ

The ordinary GRNN training procedure is a mean-square-error (MSE) minimization procedure, accomplished using a cross-validation (*k-fold*, *leave-one-out*) approach (Haykin 1999; Bishop 1995). A target function (MSE) is computed from results for all samples taken out, one by one, and estimated based on the other samples and a given σ value. The values of σ change, taking values from an interval $[\sigma_{low}, \sigma_{high}]$:

$$\sigma_i = \sigma_{low} + (i-1)\frac{\sigma_{high} - \sigma_{low}}{M} \tag{18}$$

where M is the number of steps to perform. The initial interval of possible σ values and the number of steps must be defined. The interval and the number of steps have to be consistent in order to catch the expected optimal value. To make the interval selection more general, it is useful to transform the input space values to the [-1, 1] interval using, for example, a simple linear transform. After that all possible widths are within a unit. Thus, the starting interval and the number of steps are not very important. Choosing too wide an interval and/or a large number of steps only lead to spending more time on calculations. But with modern computers and a data set with about 200 points this factor is not very significant.

A value of σ which minimizes the MSE can be adopted as a solution. Furthermore, any gradient search method can be used for tuning this value. With an excellent starting point – a well-defined minimum, and a small search area – the gradient search method finds the solution quickly and easily. Even if an optimal value of σ is outside the initial interval, it is not a problem for the gradient search method to do some steps out, as it is not limited by any boundaries.

The SIC2004 exercise deals with two-dimensional data in the (x,y) coordinate space. In this case, an advanced GRNN realization may be applied, using an anisotropic 2D parameter $\sigma=(\sigma_x,\sigma_y)$. Consequently, a 2D (rectangular) interval is used for σ optimizing the algorithm analogous to one described above for an isotropic case. Only there would

be $M_x * M_y$ values for σ to check. Usually this 2D error function has one smoothed, well-defined minimum. The value of σ giving the minimum error may be adopted as a solution or tuned using a gradient search method. In the current study a conjugate gradient search method was used for final tuning of σ. This algorithm is absolutely automatic, not requiring any optimization parameters and also very fast and effective. Its source code and application details can be found in Masters (1995). Applying a gradient search for final parameter tuning allowed us to catch a value for σ good enough for the second ("joker") data set, which appeared to fall outside the initial interval.

Finally, the procedure for applying GRNN for the SIC2004 exercise was the following:

1. Normalize the input space to the interval [-1, 1] (by linear transformation).
2. Define σ_{low} as 0.03 and σ_{high} as 0.4 for both directions (in transformed coordinates). The bounds were selected based on analysis of the prior data. The number of calculations is 10 for both directions. So we have 10*10=100 different values of σ.
3. Download data sets from the web site. Start the competition.
4. Start the leave-one-out cross-validation routine for each 100 values of σ. Calculate the cross-validation error for each value. Figure 3 presents cross-validation errors as a function of σ. The lower is the MSE, the lighter is the color. The scale for errors is not important because we search for minimum only and do not pay any attention to the values themselves.
5. Select the σ value that minimizes the cross-validation error. Use this value as a starting point for the conjugated gradient tuning procedure.
6. Using the transformation parameters from step 1, do back transformation for the optimal σ value. Table 1 present the corresponding results. Compare with Table 3 for results of σ tuning for prior data.
7. Perform prediction for 808 points using the optimal σ value.
8. Send results to the exercise organizers. End of the competition.

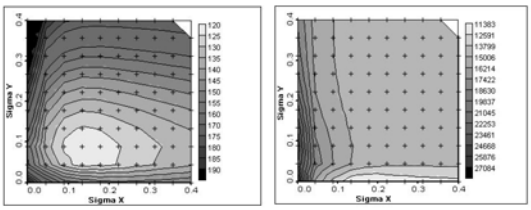

Figure 3
Contour lines of the cross-validation errors (MSE) for the 1st (left) and the 2nd data set (right). Crosses indicate locations of the σ values from the initial interval

Data sets	σ_X		σ_Y	
	Original	Transformed	Original	Transformed
First data set	27 287	0.16	34 285	0.13
Second data set	21 861	0.10	6 959	0.02

Table 1
Optimal anisotropic $\sigma=(\sigma_x,\sigma_y)$ values after step 6

2.4. USE OF PRIOR INFORMATION

The GRNN training procedure with an initial interval of [0.01, 0.6] was applied to all prior data sets. Table 2 presents the root-mean-square error (RMSE) and Pearson's correlation coefficient r for all 10 data sets. Optimal anisotropic σ values are presented in Table 3.

Prior day	1	2	3	4	5	6	7	8	9	10	Mean	Std. Dev.
RMSE	8.85	8.87	8.41	8.52	8.37	7.99	8.34	8.39	8.72	8.70	**8.52**	**0.26**
r	0.89	0.89	0.89	0.87	0.87	0.87	0.86	0.87	0.88	0.87	**0.88**	**0.01**

Table 2
Training errors for all prior data sets

Prior	1	2	3	4	5	6	7	8	9	10	Mean	Std. Dev.
σ_x	0.16	0.16	0.16	0.15	0.17	0.17	0.20	0.16	0.16	0.14	**0.16**	**0.01**
σ_y	0.09	0.09	0.09	0.10	0.10	0.09	0.08	0.10	0.10	0.11	**0.10**	**0.01**

Table 3
Optimal σ values for all prior data sets (in transformed coordinates)

The main purpose of the prior data analysis was to define the initial interval for the optimisation procedure. The interval was selected as [0.03, 0.4] (in the transformed coordinates), the same for both directions. The expected optimal kernel width was about 0.16 for σ_x and about 0.1 for σ_y. It was completely confirmed for the first data set, similar in features to the prior data sets.

For the second data set the optimal σ values differ significantly: 0.13 and 0.02. It is a good indicator that there is "something wrong" with this data set. But automatic cross-validation combined with a gradient tuning procedure allowed us to choose good parameters even if they appeared to be outside the initial interval. Our guess is that participants who used only the prior information for training and did not perform any automatic tuning algorithms would not obtain very good results from the "joker" data set.

2.5. MEASUREMENT OF THE UNCERTAINTY

Under the assumption of independent, identically distributed (i.i.d.) random variables over the whole input space (in our case a normal (Gaussian) distribution), a GRNN estimate (Z^*) can be treated as a mean of a random variable. Variation can be considered as the squared of the residuals – $(Z^*-Z)^2$. Thus, uncertainty (variation) is computed from the residuals at the training locations (where the real values are known) by interpolating over the whole space.

Interpolation can be performed with the help of an already trained GRNN (σ parameter tuned for data prediction). It is necessary to keep in mind that such an estimation of variations, first, depends on the data points' distribution (density), and, second, is biased (as based on the biased mean estimation). The i.i.d. assumption is too strong, and it allows us to obtain only rude, approximate estimates of the real uncertainty. In reality it only allows us to detect areas where unexpected (far from predicted) values may occur with positive probability.

The other possible approach is to build a new special GRNN to predict the uncertainty, as was done in the current case. The process is fully automatic (as well as the main routine), and it was implemented as part of the automatic decision making system together with a GRNN itself. It only adds an additional training procedure where squared residuals at the training points are used as a target function. The optimal value of σ for this uncertainty GRNN is found by the same procedure as described above. The trained model can be used to estimate the uncertainty at the 808 unknown locations or anywhere else.

3. RESULTS

3.1. OVERALL RESULTS

Table 4 presents the basic statistics (minimum, maximum, mean, median and standard deviation) of the 808 estimated and observed values for the two data sets.

N = 808	Min.	Max.	Mean	Median	Std. Dev.
Observed (first data set)	57.0	180.0	98.0	98.8	20.0
Estimates (first data set)	69.9	125.9	96.8	98.8	13.9
Observed (second data set)	57.0	1528.2	105.4	99.0	83.7
Estimates (second data set)	58.7	1330.5	104.9	100.3	71.5

Table 4
Comparison of the estimated and measured values (nSv/h)

The mean absolute error (MAE), the bias (or mean error, ME), the root-mean-squared error (RMSE), Pearson's correlation coefficient r between the estimated and true values and normalized RMSE (NRMSE) for the predictions at the n = 808 locations are presented in Table 5. NRMSE is computed from

$$NRMSE = RMSE / Std.Dev.$$

where *Std.Dev.* is the standard deviation of the observed values. The NRMSE shows the relation between the RMSE of the estimated values and the standard deviation of the observed values. Thus only values of NRMSE less than 1 may be accepted as satisfactory. Otherwise, if the predictor presents the mean of the observed values as the result, then NRMSE=1.0 is obtained.

Data sets	MAE	ME	RMSE	Pearson's *r*	NRMSE
First data set	9.40	– 1.25	12.59	0.78	0.63
Second data set	14.85	– 0.51	45.46	0.84	0.54

Table 5
Comparison of errors

The first three error characteristics (MAE, ME, RMSE) depend on values of the data sets presented. So it is difficult to compare the quality of prediction for different data sets with these errors. Another two errors (r, NRMSE) are relative and do not depend on the range of the data. Values of MAE and RMSE for the second data set are worse, but the correlation coefficient and NRMSE are much better. Thus, the second data set is predicted better than the first one. Negative values of the ME indicate that GRNN, on the

average, more tends to reduce (underestimate) high values than increase (overestimate) low ones. It is the case for both data sets.

Three sets of maps with isolines are presented: they show the estimated values (Figure 4) and the associated uncertainties (Figures 5 and 6). Figure 6 presents the uncertainties from the black spot of Figure 5 (right), which are not distinguishable in the original scale. In all these figures crosses indicate point locations of the estimated values and boxes depict locations of the input values.

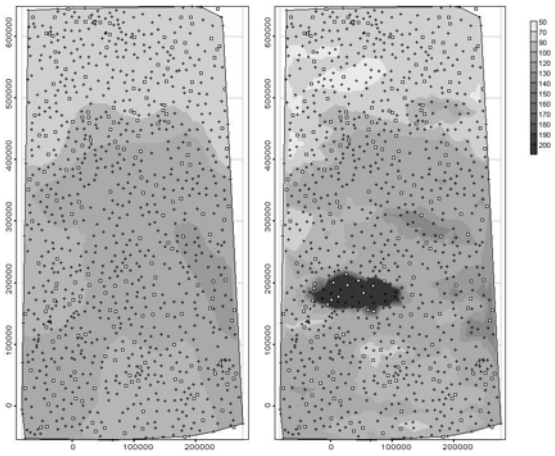

Figure 4
Isoline levels (nSv/h) for the 1st set (left) and the 2nd set (right)

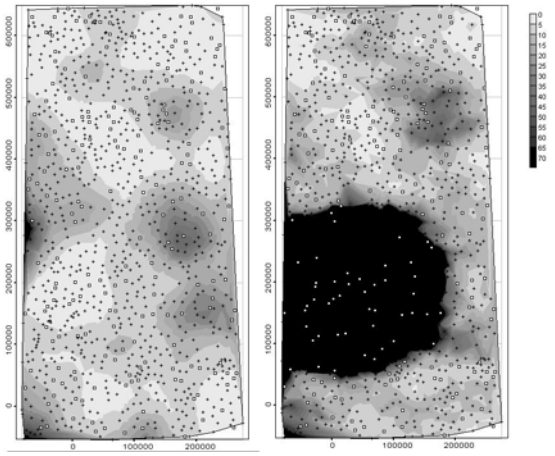

Figure 5
Isoline levels showing the uncertainty associated to the estimations obtained for the 1st set (left) and the 2nd set (right)) in the same colour scale for comparison

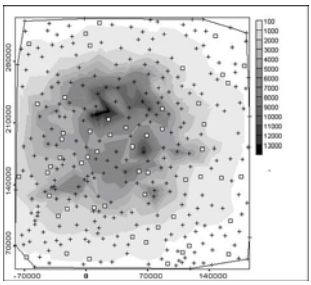

Figure 6
Isoline levels showing the uncertainty for the black spot of Figure 5 (right)

3.2. DETECTING ANOMALIES AND OUTLIERS

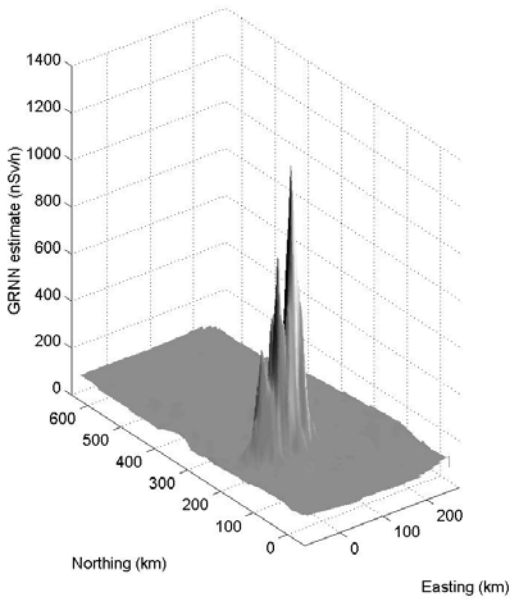

Figure 7
3D map showing extreme values found in the 2nd set by GRNN

Figure 7 presents the peak found in the 2nd data set by the GRNN approach. The smoothing characteristic for GRNN lowered the height of the peak. But visually it does not look smoothed – a collection of several rather sharp tops. Automatic GRNN allows us to adapt quickly to completely new data and to make very good predictions.

4. DISCUSSION

A General Regression Neural Network (GRNN) is an implementation of a non-parametric kernel estimator. In the current work it is treated as an automatic spatial predictor of radioactivity. It was successfully applied to two different data sets – usual routine data and unusual emergency data. GRNN is very fast – raw training and prediction calculations took about 30 seconds for each data set on a Pentium III 500 MHz PC.

The most important problem with GRNN predictor is smoothing. The smoothing degree depends on GRNN parameters tuned for the data. The presence of smoothing for the current tasks can be seen in Table 4. Minima are overestimated, maxima are underestimated, and standard deviations for the predicted distributions are much lower than the ones of the real fields. But the means and medians are reproduced very well. It means that GRNN provided a globally unbiased, smoothed estimate.

The main problems with applying GRNN are whether such a degree of smoothing is satisfactory and whether the results from the modeling are acceptable for decision making. It evidently depends on the main goal of the study: if we are interested in predicting extreme values precisely, then GRNN is not what we need. But if we need a fast, easily trained tool for recording a flux as an event, then GRNN is good enough. GRNN defined the presence of a hot spot and positioned it well. To be more certain about its applicability, we should test the model on other tasks involving emergency situations.

Let us remark here that it is not a problem to build a GRNN (define a σ) with all training errors being zero and a correlation coefficient of 1. It is enough to use σ values close to zero; a similar example was discussed for a simple problem (Figure 2b). But most probably it would imply an overfitting, because such a GRNN could not be generalized, i.e. used to predict values for new, unknown points. The generalization ability is the most important feature of any predicting model.

But a small σ does not always imply overfitting. For the second data set the optimal σ is much smaller than for the first data set (and for all prior data sets). But still the GRNN presents very good generalization abilities, which can be observed through a high correlation coefficient between real and estimated values (Table 5). Also the basic statistics (Table 4) shows even better reproduction of global distribution than for the first data set.

There is some trick with uncertainty estimation while working with a GRNN – uncertainty is estimated as a regression of known squared residuals. Perhaps a more theoretically proven approach is to pass to a probabilistic treatment of the uncertainty – to estimate the probability of exceeding a set of critical levels. It means to change a regression problem to a classification one. A required classifier can be constructed based on the same mathematical principles – Parzen's non-parametric density estimation. A probabilistic description of uncertainty is more interpretable and, thus, can be even more attractive for decision makers than direct values of uncertainty.

5. CONCLUSIONS

The current study led to the main conclusion that GRNN is a promising tool for automatic spatial prediction of radioactivity in both routine and emergency situations. It is very fast, easy to apply and interpret. It provides reasonable results, even for complex, unexpected data. The smoothing effect is a visible drawback. But it can be ignored if we are not too much interested predicting accurately high and low values, but want to detect the presence of a hot spot and indicate that a critical level is exceeded. We only need to keep in mind that maximums would always be underestimated – smoothed.

The direct estimation of the prediction uncertainty has some theoretical problems. Initially the method does not provide a measure of uncertainty; it is calculated under rather strong assumptions. But there are some ways, also discussed in this work, to introduce uncertainty estimation in the process. The other possibility is to move to a probability treatment of uncertainty.

CODES

The source codes of all GRNN-related algorithms and procedures were taken from Masters (1995). They were assimilated in a special, automatic GRNN training and prediction system.

REFERENCES

Bishop, C.M.; *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press; 1995.

Haykin, S.; *Neural Network: A Comprehensive Foundation*. Second Edition. New York: Macmillian Publishing Company; 1999.

Kanevsky, M.; 'Spatial Predictions of Soil Contamination Using General Regression Neural Networks'. *Systems Research and Info. Systems*, 1998 Mar; 8: pp 241-256.

Kanevsky, M.; 'Artificial Neural Networks and Spatial Interpolations. Case study: Chernobyl fallout'. Preprint 95-0 Moscow: Nuclear Safety Institute; 1995.

Masters, T.; *Advanced Algorithms for Neural Networks*. A C++ Sourcebook. Toronto: John Wiley & Sons, Inc; 1995.

Nadaraya, E.A.; 'On estimating regression'. *Theory of Probability and its Applications*, 1964; 9: pp. 141-142.

Parzen, E.; 'On estimation of a probability density function and mode'. *Annals of Mathematical Statistics;* 1962; 33: pp. 1065-1076.

Rosenblatt, M.; 'Density estimates and Markov sequences'. In: Puri, M., editor. *Nonparametric Techniques in Statistical Inference*. London: Cambridge University Press; 1970. pp. 199-213

Rosenblatt, M.; 'Remarks on some nonparametric estimates of a density function'. *Annals of Mathematical Statistics*, 1956; 27: pp. 832-837.

Specht, D.E.; 'A General Regression Neural Network'. *IEEE Transactions on Neural Networks*, 1991; 2: pp. 568-576.

Watson, G.S.; 'Smooth regression analysis'. *Sankhya: The Indian Journal of Statistics, Series A*, 1964; 26: pp. 359-372.

Cite this article as: Timonin, Vadim; Savelieva, Elena. 'Spatial prediction of radioactivity using general regression neural network'. *Applied GIS*, Vol 1, No 2, 2005. pp. 19-01 to 19-14. DOI: 10.2104/ag050019