Fast Interpolation using Kohonen Self-Organizing Neural Networks

Olivier Sarzeaud[1] and Yann Stéphan[2]

[1] ECTIA, 1 rue de la Noë, BP 92119, 44321 Nantes cedex 3, France,
Olivier.Sarzeaud@ectia.ec-nantes.fr,
WWW home page: http://www.ec-nantes.fr/ectia
[2] EPSHOM - CMO, 13 rue du Chatellier, BP 426,
29275 Brest cedex, France

Abstract. This paper proposes a new interpolation method based on Kohonen self-organizing networks. This method performs very well, combining an accuracy comparable with usual optimal methods (kriging) with a shorter computing time, and is especially efficient when a great amount of data is available. Under some hypothesis similar to those used for kriging, unbiasness and optimality of neural interpolation can be demonstrated. A real world problem is finally considered: building a map of surface-temperature climatology in the Mediterranean Sea. This example emphasizes the abilities of the method.

1 Introduction

Physical data interpolation is a common issue in Geosciences. For many variable of interest, the measurements are often sparse and irregularly distributed in time and space. Analyzing the data usually requires a numerical model, which samples the data on a regular grid. Mapping irregular measurements on a regular grid is done by interpolation, which aims to generalize, but not to create, information. A popular method to map geophysical data is kriging [1].

This method, based on the hypothesis that the measurements are realizations of a random variable, has been proven to be optimal under certain conditions. It requires to solve a system of linear equations at each point where the interpolation must be done, which might be computationally heavy.

This paper proposes an original interpolation method based on Kohonen self-organizing networks. The method is applied on the problem of building a surface-temperature climatology in the Mediterranean Sea. The method performs very well, combining an accuracy comparable with usual kriging methods with a much shorter computing time, and is especially efficient when a great amount of data is available.

The paper is organized as follows. Section 2 recalls the backgrounds of kriging techniques. Section 3 describes the adaptation of self-organizing maps to the spatial interpolation problem. The results of actual data interpolation in an oceanographic problem are presented and discussed. The last section draws conclusions and perspectives.

2 Optimal Interpolation

A model of a physical variable aims at predicting its value anywhere at any time. The simplest model is a numerical one, that is a discrete representation of the variable. To be efficient, this representation must be done under two constraints: on the one hand no information must be missed, on the other hand a reasonable amount of storage capacity is required. It must also be done on a regular grid, in order to be usable by most analyzes tools (plotting a map of the variable, computing Fourier Transform, ...).

2.1 Definition of Interpolation

Considering n values (obtained by measurements) of a variable Z_i at locations $x_i, 1 \leq i \leq n$, interpolation aims at building a numerical model of the variable on a regular pre-defined grid. A straightforward way to interpolate data on a specific location (a point of the grid) is to make a linear combination of the data:

$$Z^* = \sum_{i=1}^{n} \lambda_i Z_i \tag{1}$$

where Z^* is the estimated value. The problem is to compute the weights λ_i in order to minimize the estimation error. Practically, this is not feasible, because the true values are not known. It is thus necessary to make assumptions on the behavior of the variable to define the optimality.

The simplest methods give higher weights to the nearest data. The weights are somehow inversely proportional to the distance. This corresponds to an implicit assumption of continuity of the variable, which seems reasonable for physical variables. Anyway, it is possible to do better, taking into account the spatial correlation of the data. In this case, the weights are the solutions of a system of linear equations, that can be obtained by writing the minimization of the estimation error. This is kriging.

2.2 Kriging

Kriging is based on a statistical interpretation of the measures. Indeed, it assumes that the data are realizations of a random variable, that is: $Z_i = Z(x_i)$. Some hypothesis are required on the behavior of this random variable, usually that the expectation of an increment is null, and its variance only depends on the distance (intrinsic random variable [5]):

$$E[Z(x+h) - Z(x)] = 0 \tag{2}$$

$$Var[Z(x+h) - Z(x)] = C(h) \tag{3}$$

Therefore, on each point x_0 where the interpolation is to be done, it is possible to write analytically the expectation and variance of the estimation error $Z^*(x_0) - Z(x_0)$.

Unbiasness. The nullification of the expectation (ensuring that the estimation is not biased) leads to a constraint on the weights λ_i:

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{4}$$

Optimality. The minimization of the variance (that is the optimality of the estimation) under the constraint of Eq. 4 leads to a system of linear equations, with coefficients depending on a model of the variance of the increment of the data [5]:

$$\begin{bmatrix} C_{11} & \dots & C_{1n} & 1 \\ \dots & \dots & \dots & \dots \\ C_{n1} & \dots & C_{nn} & 1 \\ 1 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \dots \\ \lambda_n \\ \mu \end{bmatrix} = \begin{bmatrix} C_{10} \\ \dots \\ C_{n0} \\ 1 \end{bmatrix} \tag{5}$$

where $C_{ij} = Var[Z(x_i) - Z(x_j)] = E[(Z(x_i) - Z(x_j))^2]$ and μ is a Lagrange multiplier.

Estimation Error. Once the weights are found, it is also possible to compute the (residual) variance of the estimation error at each point where an estimation is performed:

$$Var[Z^*(x_0) - Z(x_0)] = \frac{1}{2} \sum_{i=1}^{n} \lambda_i C_{i0} \tag{6}$$

This approach is also called objective analysis.

When a great amount of data is available, kriging at each point cannot be performed using all data, because it would lead to huge systems that may not be handled. Instead, it is necessary to choose a few data around the point where to interpolate. Furthermore, these data have to be chosen to avoid singularity of the system, which is usually done with the help of many geometric parameters in kriging products. Anyway, it remains that a system of linear equations must be solved on each point of the final grid, which is computationally heavy.

The main advantage of kriging is that it relies on strong theoretical backgrounds, which demonstrate that the interpolation is unbiased and optimal. The main drawbacks are:

- The hypothesis done on the random variable are strong. It is possible to relax them (allowing a determinist drift on the data for example), but the kriging system is then more complex. In any case, a model of variance of the increment of the variable must be computed, which can be very long when a lot of data is available.
- It is difficult to ensure that a system built with some data, even carefully chosen, will be regular. Therefore, especially with big data sets, it is possible to have wrong estimates. Very wrong estimates can usually be detected, because they simply are out of the range of the variable. Anyway, it remains

the possibility to have wrong estimates that are not far enough from the expected value to be detected.
- To make a numerical model on a grid, it is necessary to interpolate, that is to solve the system of equations, on each point of the grid. This again might be very long, depending on the desired resolution of the model.

3 Neural Interpolation

Kohonen networks are artificial neural networks. Some work has already been done and is presented elsewhere [8][9] on their use for adaptive meshing. It was shown that a simple modification of the basic Kohonen self-organizing algorithm (to constrain the peripheral neurons of the network to stay on the border of the domain) allows to produce valid meshing, with some advantages over classical methods. The use of Kohonen networks for neural interpolation also relies on a slight modification of the basic self-organizing algorithm.

3.1 Kohonen Self-organizing Networks

In their widely used form, Kohonen networks consist of a matrix of neurons, each neuron being connected to its four nearest neighbors through fixed connexions (this form is called a map). All neurons are also excited by the same input, a vector of any dimension, through weighted connexions (figure 1). The role of the fixed connexions is to create a competition process between the neurons, so that the one whose weights are the closest to the current input produces the higher output. This competition is usually simulated by a simple computation of the distances between the input and all the neurons, and selection of the neuron with the smallest distance. This neuron is called the cluster of the network.

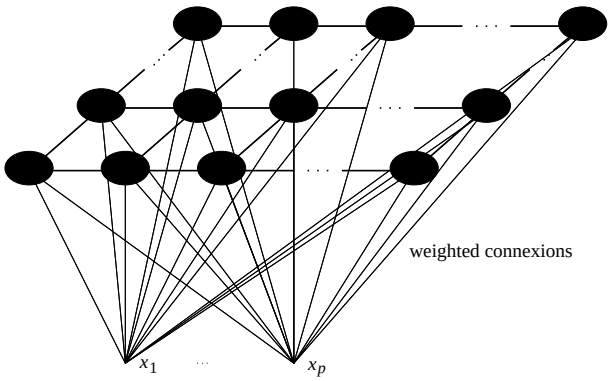

Fig. 1. Structure of a self-organizing map.

Kohonen has proposed a learning rule, that modifies the weights of the network, so as to produce interesting representations of the input space [4]. Indeed, at the end of the learning process:

- each neuron is sensitive to a particular zone of the input space;
- neighbor neurons are sensitive to near zones;
- the neurons distribution tends to approximate the probability density of the inputs presented during learning.

For these reasons, the usual representation of a self-organizing map is done by plotting the neurons, linked by their fixed connexions, using the weights as coordinates in the input space (see figure 2).

Learning Rule. Let:

- p be the dimension of the input space;
- $x(t) = (x_1(t), x_2(t), ..., x_p(t))$ be the input vector at time t;
- $w^k(t) = (w_1^k(t), w_2^k(t), ..., w_p^k(t))$ be the weight vector of neuron k at time t.

At each time step t, the cluster $c(t)$ of the network is searched:

$$\begin{aligned} c(t) &= c\left(\{w^k(t)\}, x(t)\right) \\ &= k / \|w^k(t) - x(t)\| \leq \|w^l(t) - x(t)\| \quad \forall l \end{aligned} \tag{7}$$

where the norm is usually the euclidian distance. The weights of the neurons are then adapted with the following rule:

$$w^k(t+1) = w^k(t) - \alpha(t) h(k, c(t))(w^k(t) - x(t)) \tag{8}$$

where $\alpha(t)$ is a time-decreasing gain factor, and $h(k, c(t))$ a neighboring function. This function depends on the topologic distance, measured on the map, between the neuron k and the cluster $c(t)$. It takes a maximum value of 1 if the distance is null (neuron k is the cluster), and decreases when the distance increases. The topologic distance between two neurons is the distance between their row and column indices in the map.

This algorithm is very robust, and numerous constraints can be applied to the neurons without changing its properties. For example:

- Initializing the network as a regular grid in the whole space considerably reduces the computation time.
- Constraining the peripheral neurons of the network to slide on the border of the domain allows the production of naturally adapted meshing [9]. The right part of figure 2 gives an illustration of such a meshing, produced with the same parameters as the left part, except the constraint.

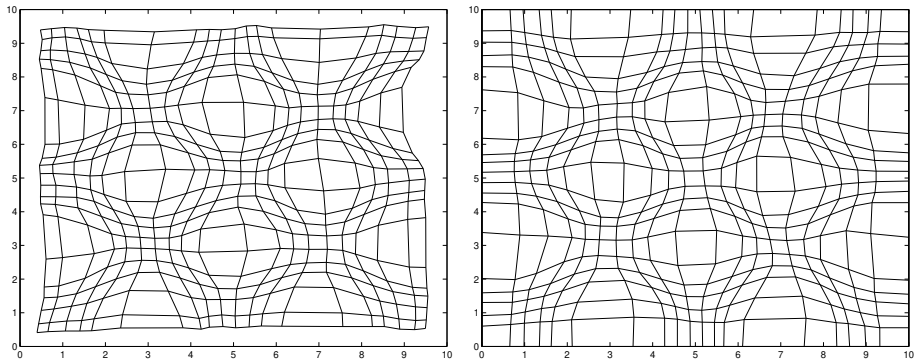

Fig. 2. Two maps organized using data chosen randomly on the black squares of a chess. The only difference between the two is that peripheral neurons are constrained to slide on the border of the domain on the right.

3.2 Neural Interpolation Algorithm

At each time step t, the input $x(t)$ of the network is a three-dimensional vector, the first two dimensions giving the location of the measure, and the third one its value. We will note this decomposition: $x(t) = (x_{loc}(t), x_{val}(t))$ ($loc = 1, 2$ and $val = 3$). Each neuron k thus has a three-dimensional weight vector $w^k(t)$. The only required modification of the basic self-organizing algorithm is that the selection of the cluster $c(t)$ be performed on the first two dimensions only:

$$\begin{aligned} c(t) &= c(w^k_{loc}(t), x_{loc}(t)) \\ &= k / \|w^k_{loc}(t) - x_{loc}(t)\| \leq \|w^l_{loc}(t) - x_{loc}(t)\| \quad \forall l \end{aligned} \tag{9}$$

This means that the cluster is chosen in the geographical space, according to the data location only, and is completely independent from the measure value. The idea is to trust the locations rather than the measures, allowing thus very different values measured on close points to be combined. Once the cluster is found, the weight modification applies on all three weights of the neurons, as presented in Eq. 8.

In this approach, the interpolation points cannot be chosen beforehand. Instead, they are determined during the learning process, and correspond to the final locations of the neurons. Therefore, at the end of the algorithm, we still do not have the values on a regular grid, and a post-processing is required. The question is thus: what is the advantage of the final irregular distribution of the neurons over the initial irregular distribution of the data? If the number of neurons is lower than the number of measures, complexity is reduced without loss of information. The neurons set is the best representation in space of the data set [6]. If the associated values are optimal, it is then possible to re-grid the values with a simple interpolation method, with no loss of precision.

At any time step t, it can easily be shown that the weight vector of a neuron k is a combination of the input vectors presented until then and of its initial value:

$$w^k(t) = \sum_{i=1}^{t} a^k(i) \prod_{j=i+1}^{t} (1 - a^k(j)) x(i) \quad + \quad \prod_{i=1}^{t} (1 - a^k(i)) w^k(0) \tag{10}$$

where $a^k(i) = \alpha(i) h(k, c(i))$. The initial value $w^k_{val}(0)$ can always be set to 0 without loss of generality. The weights of the combination do not depend on the third dimension (value) of the input and neuron weight vectors, but rather on the gain factor and on the locations of the input and neuron. Therefore, the third weight of the neuron is at any time a true *linear combination* of the data presented until then:

$$w^k_{val}(t) = \sum_{i=1}^{t} \lambda^{kt}(i) x_{val}(i) \quad with \quad \lambda^{kt}(i) = a^k(i) \prod_{j=i+1}^{t} (1 - a^k(j)) \tag{11}$$

Is this linear combination optimal? We will show this in the same way as for kriging. We first assume that the data are realizations of a random variable Y. But this time, this variable is not stationary, and is the sum of an intrinsic random variable (the Z variable used in kriging) and a determinist *linear* drift m:

$$Y(x) = Z(x) + m(x) \tag{12}$$

We can first make the following analogies with kriging notations:

$$\begin{array}{l} x_i = x_{loc}(i) \;\; Y(x_i) = x_{val}(i) \\ x_0 = w^k_{loc}(t) \; Y^*(x_0) = w^k_{val}(t) \\ \lambda_i = \lambda^{kt}(i) \end{array}$$

With these notations, the formula of Eq. 11 can be rewritten:

$$Y^*(x_0) = \sum_{i=1}^{t} \lambda_i Y(x_i) \tag{13}$$

If t is sufficiently big, the second term of Eq. 10 (influence of the initial weight vector of the neuron) can be neglected, therefore leading to:

$$x_0 = \sum_{i=1}^{t} \lambda_i x_i \tag{14}$$

Unbiasness. The expectation of the estimation error at point x_0 can be written:

$$E[Y^*(x_0) - Y(x_0)] = E\left[\sum_{i=1}^{t} \lambda_i Z(x_i) - Z(x_0)\right] + \sum_{i=1}^{t} \lambda_i m(x_i) - m(x_0) \tag{15}$$

This expression has a first probabilist term and a second determinist term. Each of them must be nullified to ensure unbiasness. The second term is naturally

null, because the drift is linear and the final location x_0 of the neuron is a linear combination of the location of the data presented (Eq. 14):

$$m(x_0) = m(\sum_{i=1}^{t} \lambda_i x_i) = \sum_{i=1}^{t} \lambda_i m(x_i) \tag{16}$$

We can thus say that *moving the neurons filters the drift.*

Nullity of the first term requires the same condition as for kriging: the sum of the λ_i must be 1. Let us note A_t this sum. Developping A_t using Eq. 11 leads to:

$$A_t = 1 - \prod_{i=1}^{t} (1 - a^k(i)) \tag{17}$$

A_t tends to 1 when t increases iff the *log* of the product tends to minus infinity. A first order development gives:

$$log(\prod_{i=1}^{t} (1 - a^k(i))) = -\sum_{i=1}^{t} a^k(i) + o(a^k(i)^2) \tag{18}$$

If the gain factor $\alpha(t)$ follows a decreasing law of the type $1/t^\beta$ with $0 \leq \beta \leq 1$, which is the usual convergence condition of Kohonen networks [7], then the sum goes to infinity, and A_t converges to 1.

Optimality. To show the optimality, that is the minimization of the variance of the estimation error, is much more difficult. Indeed, the neural interpolation algorithm never uses an explicit knowledge of the variance of the increment of the random variable. Therefore, an assumption is needed on how this variance is taken into account in the algorithm. We suppose first that the variance only depends on the distance between data points in the representation space of the map instead of the input space. Furthermore, we suppose that the neighboring function is a good representation of this variance:

$$C_{ij} = C_0(1 - h(c(i), c(j)) \tag{19}$$

where C_0 is a normalisation factor. This assumption is intuitively true for very big data sets. Indeed, in this case, measurements were made where variations were expected rather than where the variable was known to be stable. The resulting distribution thus reflects the variability of the measures.

The (determinist) drift naturally disapears when writing the variance of the estimation error. Optimality is therefore ensured under the same condition as for kriging (first n lines of the system of Eq. 5):

$$\sum_{i=1}^{t} \lambda_i C_{ij} = C_{j0} \quad \forall j \tag{20}$$

where $C_{j0} = C_0(1 - h(c(j), k))$, k being the considered neuron. Under the hypothesis of a unitary neighborhood, λ_i is non zero only when $c(i) = k$, thus when $C_{ij} = C_{j0}$. The sum of the λ_i being 1 as shown above, this demonstrates optimality.

Estimation Error. If a model of variance of the increments is available, the variance of the estimation error can be iteratively computed during the learning process. An updating rule similar to the one of Eq. 8:

$$C^k(t+1) = C^k(t) - \alpha(t)h(k, c(t))(C^k(t) - C_{t0}) \tag{21}$$

would lead to the following result:

$$C^k(t) = \sum_{i=1}^{t} \lambda_i C_{i0} \tag{22}$$

which is simply twice the variance of the estimation error defined in Eq. 6.

However, as no model of variance is required for neural interpolation, we would prefer not to have to compute one at all. This model is an analytic representation of the mean of the squared increments between data, function of their distance. Neural interpolation being a stochastic process, we propose to use at each time step the squared increment between the data presented and each neuron, instead of a model of what this value should be. This leads to the following new updating rule:

$$C^k(t+1) = C^k(t) - \alpha(t)h(k, c(t))(C^k(t) - (w^k_{val}(t) - x_{val}(t))^2) \tag{23}$$

This rule allows a better understanding of the local variability of the measures. Outliers can be more easily detected on the resulting map, because the local error they produce is not smoothed, as would be the case with a general model of variance. The error map rather reflects the variability of the data than their density.

4 Comparison

4.1 Algorithmic Complexity

It is important to compare the algorithmic complexity of kriging and neural interpolation, according to the numbers of data n, interpolation points m and iterations t.

Concerning kriging, two steps are required: computation of the model of variance first, and estimation itself. The first step needs a constant number of operations a for all n^2 couples of data points. The second step consists in solving a system of p linear equations with p unknowns, where p is the number of data points considered. Remember that when the data set is big, all data cannot be considered to build the system of equations. We do not take into account the time needed to cleverly choose these p points. The resolution needs $bp^3/10$ operations, where b is a constant depending on the chosen method, the same order as a. For m estimation points, kriging complexity is thus: $an^2 + bmp^3/10$.

Concerning neural interpolation, two steps are also required at each time step. First, the distance between the considered data point and all neurons is

computed (we consider that we have m neurons). The cluster can be found in constant time, under some simple hypothesis [10]. Then, all neurons are updated according to their distance to the cluster. These two steps require cmt operations, where c is the same order as a and b.

Neural interpolation and kriging have a comparable cost if the number of iterations t is the same order as the maximum of n^2/m and $p^3/10$. Clearly, when very few data points are available, kriging is faster, while when a lot of data points are available neural interpolation is faster. Two numerical examples are given in table 1. The number p of points used to build the systems of equations for kriging is 20. Although quite low when compared to the number of data points available, it is often sufficient. The number t of iterations for neural interpolation is chosen from our experience to allow convergence of the algorithm.The last column gives (an order of) the number of operations required for neural interpolation, while the one before gives this number for kriging. Remember that the problem of choosing the right p points for kriging is not taken into account to evaluate the number of operations.

n	m	p	t	# op. kriging	# op. neur. int.
100	1,000	20	1,000	810,000	1,000,000
10,000	1,000	20	10,000	100,800,000	10,000,000

Table 1. Numerical examples of the algorithmic complexity of each method.

4.2 Practical Results

The neural interpolation method has been used on synthetic and real data sets. The results are available in [10]. The synthetic data sets aimed at controlling the optimality of the results, and were therefore not too big. Some criteria were defined to assess the mean bias and optimality of the interpolations for all the neurons of a network. It was shown that the bias is nearly null if there are enough modifications of the weights, that is if the number of iterations and the gain are sufficiently high. Optimality is ensured with a very low relative error (less than 3%), and requires the same conditions as nullity of the bias. The residual relative error must be compared with the error between the experimental variance and the model used in kriging, which is generally about 8%.

Kriging has been used during the european project MEDATLAS to make an atlas of the temperature climatology field over the Mediterranean sea [2]. The aim was to produce a series of maps of the temperature climatology state of the sea, for each month and at some 29 standard depths (that is 348 maps). The number of data available was related to the depth considered, from about 260,000 in surface to less than 1,000 at the bottom. The domain to model was the whole Mediterranean basin on a grid of about 25 km step, which makes more than

15,000 grid points. To achieve this goal in a reasonable computation time, the kriging approach was based on an adapted meshing of the data set, that allowed to select the points where it was worth computing. For the chosen points, a universal kriging method was used, based on a regional model of variance. The results were then re-gridded on a regular grid using a weight-distance linear combination of the 4 closest estimated values. An example of temperature map is given in figure 3.

A real data set was taken from the MEDATLAS project, and aimed at showing the computing time gain. The data set contained about 25,000 measures. The interpolation was required on a 204x72 points grid. The result was checked using cross validation. With this aim, the regular estimation and error grids were first reinterpolate to the data points using a classical bilinear interpolation. Then, the error between the value used to build the map and the value given by the map at each data point was computed. The mean error on all data points was found to be -0.02, which is close enough to 0 to say that the estimation is unbiased. Finally, the error on each point was compared with the predicted error. The mean value was found to be 0.82, which is close enough to 1 to say that the error map is coherent with the estimation. However, there can be cases where the estimation is very bad, although not biased, and coherent with the error computed. Therefore, it was necessary to make a graphical comparison between the maps. The map built by neural interpolation compared well with the MEDATLAS one (figure 4). The neural interpolation method required less than 1 minute of computing time, while the method used in MEDATLAS (which hopefully limited the number of points where to interpolate with the help of an adapted meshing) required more than four hours.

5 Conclusion

An original method for fast interpolation of big data sets is presented in this paper. The method relies on some basic modification of the standard Kohonen algorithm. Its unbiasness and optimality are demonstrated under hypothesis similar to those used in kriging. The method has been applied on several synthetic and actual data sets. In every cases, the results compare perfectly well with those obtained by kriging. However, the method is much faster than kriging when the data set is large, which is practically the case for actual problems. Future work will deal with taking into account an error on each data point, what kriging can do. Other studies will deal with the use of the Kohonen algorithm for data fusion and assimilation.

Acknowledgment: This work was funded by the Service Hydrographique et Océanographique de la Marine (SHOM) under contract 98.87.054. The authors wish to thank Didier Jourdan (CMO) for helpful comments and for making available a part of the MEDATLAS data set.

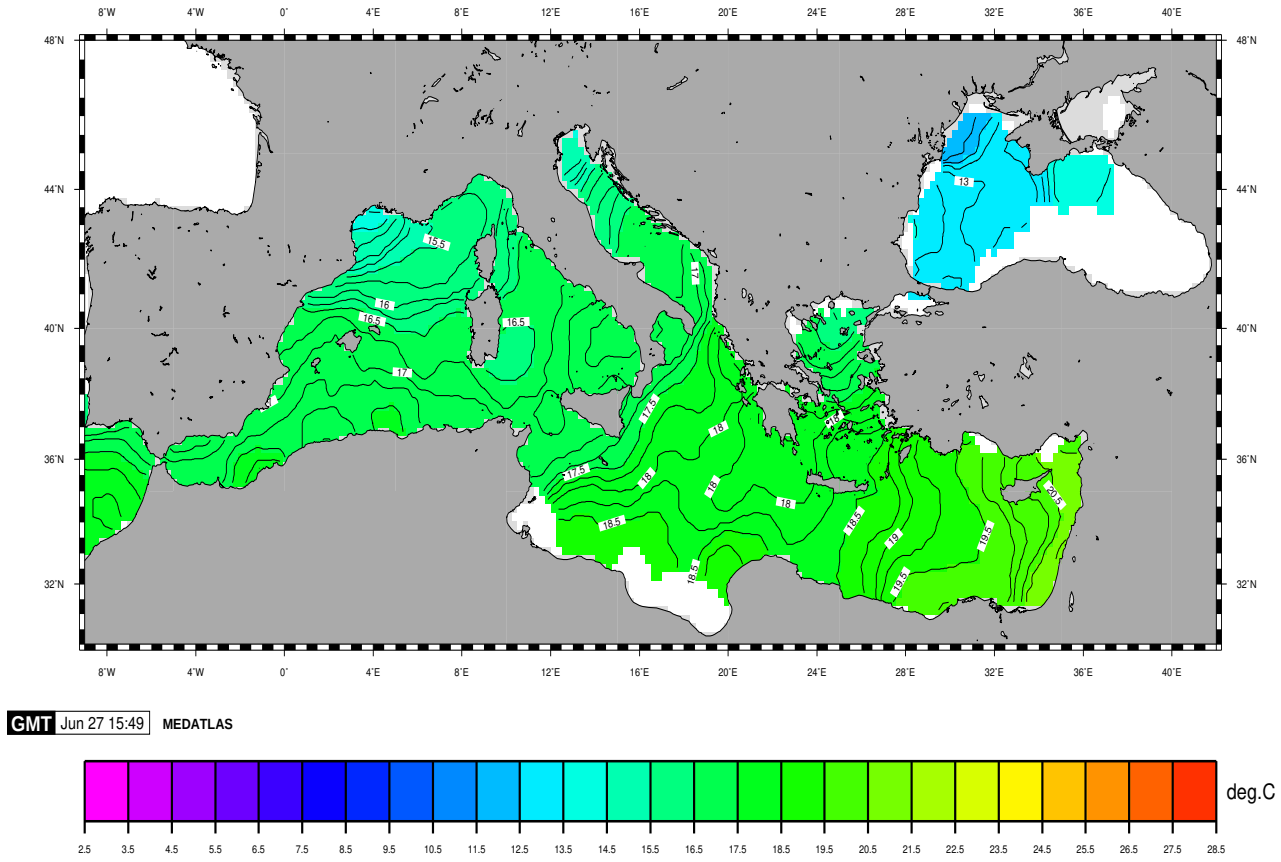

Fig. 3. MEDATLAS map of May at 10 m immersion [3].

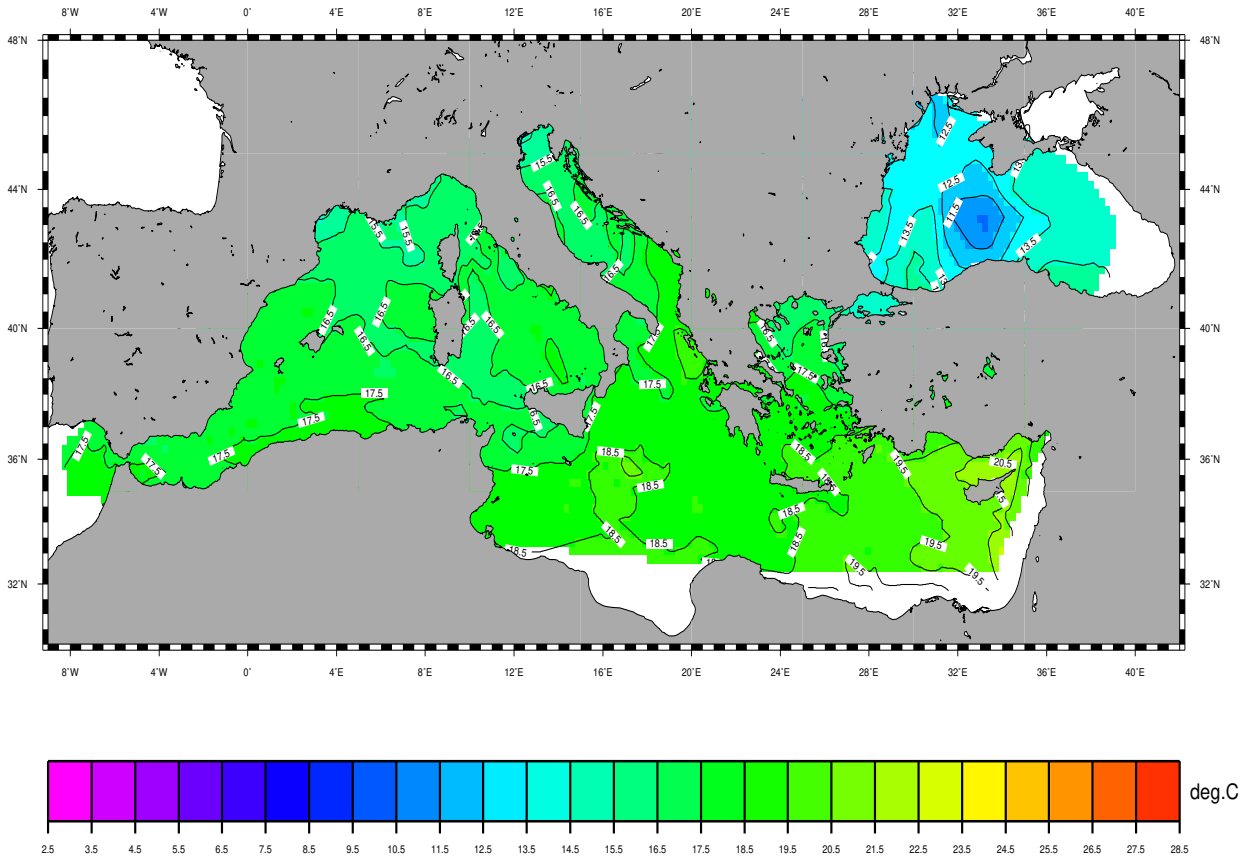

Fig. 4. Map of the Med data set produced by neural interpolation.

References

1. David, M., Crozet, D., Robb, J.M.: Automated mapping of the ocean floor using the theory of intrinsic random functions of order k. Marine Geophysical Researches **8** (1986) 49–74
2. Jourdan, D., Balopoulos, E., Garcia-Fernandez, M.-J., Maillard, C.: Objective Analysis of Temperature and Salinity Historical Data Set over the Mediterranean Basin. OCEANS'98, IEE/OES Ed. (1998) vol. 1, 82–87
3. Jourdan, D.: Bilan du projet MEDATLAS. Technical report 02P98, Service Hydrographique et Océanographique de la Marine (1998)
4. Kohonen, T.: Self-organizing maps. Springer-Verlag (1995)
5. Matheron, G.: The intrinsic random functions and their applications. Advances in Applied Probability **5** (1973) 439–468
6. Ritter, H., Schulten, K.: On the stationnary state of Kohonen's self-organizing sensory mapping. Biological Cybernetics **54** (1986) 99–106
7. Ritter, H., Schulten, K.: Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection. Biological Cybernetics **60** (1988) 59–71
8. Sarzeaud, O.: Les réseaux de neurones, contribution à une théorie. Ouest Editions (1994)
9. Sarzeaud, O., Stéphan, Y., Le Corre, F., Kerleguer, L.: Neural meshing of a geographical space in regard to oceanographic data location. OCEANS'94, Brest, France (1994)
10. Sarzeaud, O.: Interpolation optimale et assimilation de données par réseaux de Kohonen. Technical report OS/99003 (1999)