Introduction to Self-Organizing Map Modelling for Imputation – Techniques & Technology

Pasi PIELA
Statistics Finland
FIN-00022 STATISTICS FINLAND, FINLAND
e-mail: Pasi.Piela@stat.fi

Abstract: In this paper I will discuss one modern approach to imputation. Many traditional methods of imputation use some kind of classification trying to get observations with missing values into as homogenous groups as possible. Self-organizing map (SOM) is an iterative method for classification and can thus also be used in finding the imputation classes. Imputations are made within clusters, located by corresponding neurons, in several ways which can be based on both traditional and neural methods.
SOM methods are included in the versatile program named NDA, Neural Data Analysis, which was made by the research group on Software Engineering and Computational Intelligence of the University of Jyvaskyla, SECI, Finland.
Imputation methods have been implemented into NDA in co-operation with the research group of Statistics Finland. This paper is based on research work in the Euredit FP5 project.

Keywords: Kohonen algorithm, tree-structured self-organizing maps, neural

1. Introduction

The research group on Software Engineering and Computational Intelligence (SECI) of the University of Jyvaskyla (JyU), Finland, developed a software called the Neural Data Analysis environment (NDA) with Professor *Pasi Koikkalainen* as the group leader, and the NDA is naturally still under heavy development. The software provides a generic application platform for computational intelligence with many proven examples of real world applications. The main emphasis of the software is to aid methodological development of knowledge discovery, data analysis, and modelling in general [2][1]. Techniques are mainly

[1] Erkki Häkkinen's doctoral thesis (JyU) is the main reference in this paper, see References.

based on neural networks, but the system also includes a large data set of data manipulation and visualisation techniques, fuzzy sets etc. However, the most important method used in the NDA is the Tree-Structured Self-Organizing Map [6], usually abbreviated as TS-SOM. TS-SOM is a modification of the Self-Organizing Maps (SOM) [3].

During the development of the NDA it was noticed that one of the main problems with the use of neural networks in practice is incomplete data with missing and erroneous units. Modern neural oriented methods for error localization and imputation are under research and development especially in the EU/FP5 project EUREDIT[2] (*The Development and Evaluation of New Methods for Editing and Imputation*) in which JyU and Statistics Finland are working together to develop SOM based techniques for editing and imputation.

This paper gives a rough introduction to the TS-SOM methodology for imputation. Practical example, in which the data are derived from the Euredit project, is also presented.

2. Tree-Structured Self-Organizing Map Modelling

The basic SOM defines a mapping from the input data space $\boldsymbol{R}^n$ onto a latent space consisted typically of a two-dimensional array of nodes or neurons [3]. A parametric reference or weight vector $\boldsymbol{w}_i$ is chosen for each neuron i from the discrete set $i = \{1, 2, \ldots, N\}$. Now, let $\mathbf{x} \in \boldsymbol{R}^n$ be a stochastic, random data vector. Usually the smallest of the Euclidean distances $||\mathbf{x} - \mathbf{w}_i||$ is made to define the best matching unit (BMU) for the vector $\mathbf{x}$, denoted by the subscript b, that is, $b = \text{argmin}\{||\mathbf{x} - \mathbf{w}_i||\}$. Then SOM algorithm updates the weight vectors of the BMU and *in its neighbouring neurons* i as follows:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + h_{bi}(t)[\mathbf{x}(t) - \mathbf{w}_i(t)]$$

where $h_{bi}(t) = a(t)H\left(\frac{\mathbf{v}_b - \mathbf{v}_i}{s}\right)$ defines the neighbourhood kernel $H(\cdot)$ over the lattice points and the learning rate $(0 < a(t) < 1)$ at iteration step t. One common, Gaussian, kernel function is $H(v) = exp(-||\mathbf{v}||)$. Thus, each iteration starts with the new random sample. The idea is to start with large neigbourhood and reduce it along with the value of the learning rate parameter. Specifically, the SOM algorithm can be interpreted as an discretized approximation procedure for computation of principal curves or surfaces [7].

The map is usually created in a non-stochastic way by using the so-called *batch algorithm* in which all the data points are associated with their BMUs in each iteration, and then all the prototypes are updated at a time.

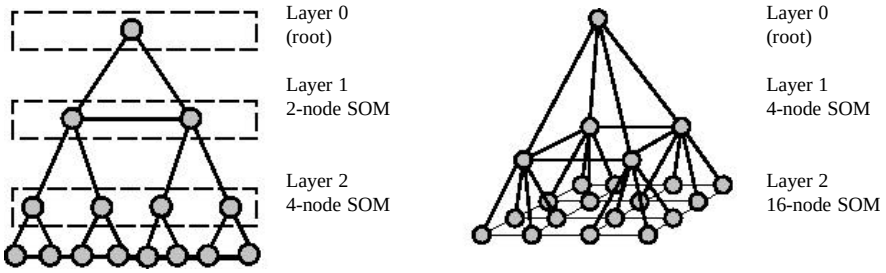
The Tree-Structured Self-Organizing Map is made of several SOMs arranged to a tree structure (see Figure 1). The topmost layer ($L = 0$) has one neuron. Layer 1 has four neurons in two-dimensional and two neurons in one-dimensional case. Thus, each neuron has its own associated subgroup of data, four subgroups on layer 1 but one group, the data set itself on layer 0. The subgroup forms the cluster of which centroid is the weight

[2] For more information about the project Euredit: *http://www.cs.york.ac.uk/euredit/* .

vector of the best matching unit b, $\mathbf{w}_b$. Furthermore, the centroid on layer 0 defines the mean of all data.

The training is repeated layer by layer using knowledge about the BMU of the frozen layer l–1 in the search of the BMU on next layer l. That is, the search of the BMU for the layer l is restricted into a small set of neurons: sons and sons of neighbours of the BMU of the previous layer. This reduces clearly the *computational complexity* when compared to the basic SOM. We will discuss this later.

Figure 1. Illustrations of one and two-dimensional TS-SOM structures.



The training is usually made with the batch algorithm [4]. During each epoch the BMUs are searched for all data vectors using the tree search, and then new centroids $\mathbf{m}_i(t)$ are the weight vectors $\mathbf{w}_i(t)$ computed using the rule:

$$\mathbf{w}_b(t+1) = \frac{1}{N_b + \sum_{i \in N_c(b)} a\, N_i} \left(N_b m_b(t) + \sum_{i \in N_c(b)} a\, N_i \mathbf{m}_i(t) \right)$$

where $N_c(b)$ is a set of indices of neighbours of b, and N_i is the number of data records in the Voronoi region (cluster) i. The smoothing is partially controlled through the parameter $a \in [0..1]$. One side advantage here is that the size of the neighbourhood can be kept constant, and the usual problem with the basic SOM, namely, relation between neighbourhood size and the learning rate parameter during each epoch, does not exist.

3. TS-SOM Based Imputation

A natural approach to the missingness problem is now imputation within the clusters located by associated neurons. A simple starting point is to derive analogous processes from the classical imputation methods. Nearest neighbour imputation can be made by filling missing components of the data vector from the nearest data vector within the same cluster. Group means imputation (replacing the missing value by the average value of the observations belonging to the same class/subgroup) can be made by taking the centroid of the cluster, $\boldsymbol{m}_b$, and replacing the missing component j of the data vector $\mathbf{x}_i$ by corresponding $\boldsymbol{m}_b(j)$, which is actually the fastest way to impute.

Besides previous simple imputation models, it is obvious that more complex, regression based, imputation modelling should be taken under consideration when trying to take advantage of TS-SOM mapping, and thus take values for missing components from these models which are generalizations of the local distributions of each data cluster. The MLP network with the backpropagation algorithm is often used to model the relationship between complete and imputation variables (MLP theory and methodology, see [1]). From the SOM mapping point of view, the MLP networks can be trained separately in each cluster of the final TS-SOM layer. Thus, we have as many *local* MLP models as we have neurons on the last TS-SOM layer.

However, MLP training as presented above only takes into account the resulting condition of the TS-SOM mapping. But Koikkalainen [5] has shown that the TS-SOM can be seen as a hierarchical Gaussian mixture model where each neuron is a Gaussian generator. It is thus possible to use this idea in creating an imputation model by observing the posterior probabilities of the neurons. Häkkinen [2] presents imputation methods based on this idea, calling them iterative methods with probabilistic imputation. We will not discuss these methods here any further, just state that new methods applying the idea of probabilistic distributions of building TS-SOM into imputation are being tested and developed.

4. TS-SOM Based Imputation in the Danish Labour Force Survey Data Set

One of the data sets for the evaluation and development of imputation methods in the project Euredit is the Danish Labour Force Survey (1996), consisting of Danish population register records for individuals selected for interview. The very carefully created synthetic version of this data set was given as training and development data for imputation methods. It is structurally very simple and will be used here. The data consist of only 14 variables with little information, four relating to type of response. Annual income (DKK) is the only variable needed to impute, the missingness rate being 26.8 %. The data are at person level having 200,000 observations, information about households is not available here. These 13 auxiliary variables are categorical, except person's age. They have 2-4 classes, except AREA (area of living), BUSINESS (last employment) and EDUCATION, for example, have 4 classes:

1 = Private Industry	1 = Primary school only
2 = Other private business	2 = Craftsman, Skilled labour, High school only
3 = Government employed	3 = Long education, school teacher, university, etc.
-9 = Not applicable	-9 = No information

The complete data vectors are used here as training data. That is, we select all 146,323 observations with known value of income. Another alternative would be selecting all the observations and training TS-SOM for these without the variable INCOME, but this is not necessarily appropriate in this simple case due to losing information of the relationship between INCOME and explanatory variables.

First we make data analysis by building TS-SOM for our training data set by NDA software and viewing its different layers. For example, it is easy to observe graphically the basic

statistics of INCOME for different clusters and comparing them to the statistics of background variables.
Variables are scaled for nearest neighbour imputation (this thus assumes that each variable has the same weight), where for the missing component $\mathbf{x}_i(j)$ that belongs to a cluster b

$$\mathbf{x}_i(j) = \mathbf{x}_n(j), \quad n = \text{argmin}_{k \in cl_b}(||\mathbf{x}_i - \mathbf{x}_k||), \tag{4.1}$$

so that all categorical variables are binarized. Thus, for variable l of m classes we have corresponding m (0/1)-variables. The continuous background variable, AGE, is scaled to [0, 1] based on min/max ranges:

$$\mathbf{x}' = \frac{\mathbf{x} - min(\mathbf{x})}{max(\mathbf{x}) - min(\mathbf{x})}.$$

However, in the case of several skewed distributed continuous variables there are naturally other appropriate methods for equalization. Also, when missingness of several components occurs the distances in (4.1) can be weighted by simply comparing the number of missing components and the number of variables for every data vector.

An obvious problem in this kind of hot-deck type of imputation is its *computational complexity* (related to computation time) that is $O(N^2)$ due to the full search among N data vectors. By using the above method and TS-SOM the complexity can be reduced to $O(N\log_p M + N^2/M)$ where the complexity of the TS-SOM algorithm is $O(N\log_p M)$ [6], p being the number of sons of each node (in two dimensional case: $p = 4$), and for the imputation within M clusters it is $O(N^2/M)$. In practice, this was clearly observed for the example in question as follows:

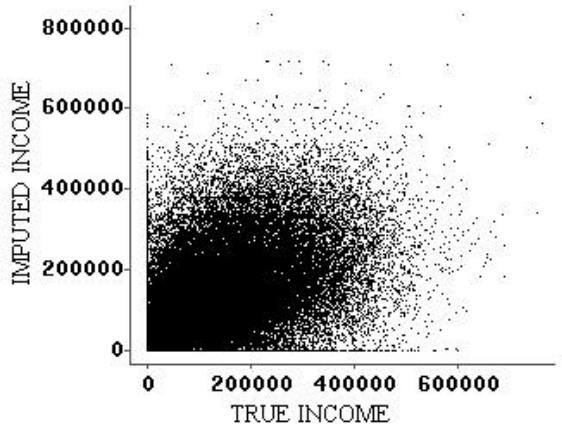
Nearest Neighbour Imputation	Computation Time, Pentium® II Processor (500)
without TS-SOM mapping (layer = 0)	> 12 hours
layer 2 as imputation layer (16 neurons)	59 minutes
layer 3 as imputation layer (64 neurons)	20 minutes

Table 1. Computation time of the nearest neighbour imputation in practice. Synthetic Danish Labour Force Data set, N = 200,000.

As shown in Figure 2, the imputation does not give good results at the unit level, which was expected because of lack of background information. But for this case the method really seems to work at the aggregate (or data) levels. Specifically, the line for observations with $\hat{Y} = 0$ and $Y^* > 0$ is quite similar to the line for observations with $\hat{Y} > 0$ and $Y^* = 0$, but the lines are long including several wrong imputed values. Furthermore, the linear regression model (y = true income, x = imputed income) has surprisingly high estimate of the intercept parameter (102034) but when modelling without intercept quite a good model with high R-square is observed.

As Häkkinen [2] points out, there are some obvious problems in this kind of traditional method together with TS-SOM. There might be clusters having only missing values and there is a risk to lose the nearest data vector to another cluster, which might be a problem here too (see Table 2).

Figure 2. Scatterplot of imputed INCOME ($\hat{Y}$) and *corresponding* true values (Y^*) from nearest neighbour imputation on the 3rd TS-SOM layer.



	True value	Centroid, 4th layer	Lin. Regression	Nearest Neighbour, 0th layer	Nearest Neighbour, 1st layer	Nearest Neighbour, 2nd layer	Nearest Neighbour, 3rd layer
Mean	**158108**	169177	170287	158068	158094	159799	159408
Stddv	**107193**	56806	61498	107856	107855	108380	108253
95%	**362639**	259743	272202	363148	362674	363374	363712
Q3	**221423**	204815	215989	220904	221534	224541	223804
Md	**140971**	173937	168978	140105	139715	141657	141616
Q1	**76691**	122951	122344	77004	76869	77914	77779
Q3-Q1	**144732**	81864	93645	143900	144666	146627	146025
D_{L1}	**0**	73079	73678	90865	91072	91593	91639
D_{Lmax}	**0**	564345	520718	664950	664950	671202	664950

	True value	MLP, method I	MLP, method II
Mean	**158108**	167354	166806
Stddv	**107193**	69143	68920
95%	**362639**	280030	277061
Q3	**221423**	225255	219764
Md	**140971**	173607	162920
Q1	**76691**	107494	109096
Q3-Q2	**144732**	117762	110668
D_{L1}	**0**	67109	67279
D_{Lmax}	**0**	508255	518621

MLP Method	1st Hidden Layer	2nd Hidden Layer	TS-SOM Imp. layer
I	4 neurons	6 neurons	1 (4 clus.)
II	10 neurons	10 neurons	3 (64 clus.)

Activation functions are sigmoidal.

Table 2. Results for Linear Regression (using SOLAS 3.0[TM3]), Nearest Neighbour, Cluster Centroid and MLP imputations of INCOME variable. Note, *true mean / standard deviation of the missing incomes is 158,108/107,193 and true mean / standard deviation of the*

[3] Solas™ and Solas 3.0™ are trademarks of Statistical Solutions LTD.

whole data set is 181,724/116,064.

$$d_{L1}(\hat{\mathbf{Y}}, \mathbf{Y}^*) = \sum_{i=1}^{n} w_i \left| \hat{Y}_i - Y_i^* \right| \Big/ \sum_{i=1}^{n} w_i \ , \ \text{here}: \ w_i = 1 \ \forall \ i \in \mathbf{N}.$$

Cluster centroid imputation, which replaces the missing values with corresponding values of the centroid of the same cluster, is not appropriate in this case. Totals are clearly over-estimated and variances under-estimated. On the other hand, the structure of TS-SOM gives possibilities to solve problematic situations: the centroid can be interpolated from a parent neuron of the upper TS-SOM layer or it is possible to use neighbour clusters and neurons as well in finding the appropriate donor or the centroid needed.

There are, naturally, a number of ways to impute by MLP models. Table 2 presents results for two structurally different groups of local backpropagation MLP imputation models for modelling dependencies between background and INCOME variables. In method I, for example, we first build TS-SOM and on its first level we use the backpropagation algorithm to create four local MLP models with two hidden layers of four and six neurons connected sequentially by logistic sigmoid units, one for each neuron (see Table 2). But these kinds of MLP models are clearly problematic for the case in question; they do not give any 0s as estimates of INCOME, and variances are under-estimated.

5. Conclusions

Self-organizing mapping is widely used and a known method for many kinds of data analyses. Efficient tree-structured self-organizing mapping is implemented to the NDA software that gives a powerful tool for data analysis, missingness analysis and imputation with graphical presentation facilities. Classical imputation methods can be used on different levels of the TS-SOM structure, but also regression based and more complex methods applying the mathematical idea behind TS-SOM can be utilised. In the imputation example, TS-SOM can also be useful after imputation tests in trying to find some specific subgroups of high differences between imputed and corresponding true values or other abnormalities, in other words, trying to get the zone in Figure 2 narrower by keeping the fitted regression line (for imputed and their true values) close to $\hat{y} = y^*$. These modern methods as well as software tools are under intensive further development.

References

[1] Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, United Kingdom.

[2] Häkkinen, E. (2001). *Design, Implementation and Evaluation of the Neural Data Analysis Environment*. PhD thesis. Jyväskylä University Library, Jyväskylä, Finland.

[3] Kohonen, T. (1997). *Self-Organizing Maps*. Springer, Berlin, Heidelberg.

[4] Koikkalainen, P. (1995). Fast Deterministic Self-Organizing Maps. In Fogelman-Soulié, F. and Gallinari, P., eds., *Proc. ICANN'95, Int. Conf. on Artificial Neural Networks*, volume II, pages 63-68, Nanterre, France. EC2.

[5] Koikkalainen, P. (1999). Tree Structured Self-Organizing Maps. In Oja, E. and Kaski, S., eds., *Kohonen Maps*, pages 121-130. Elsevier, The Netherlands.

[6] Koikkalainen, P. and Oja, E. (1990). Self-Organizing Hierarchical Feature Maps. In *Proc. IJCNN-90-Wash-DC, Int. Joint Conf. on Neural Networks*, volume II, pages 279-285, Piscataway, NJ., IEEE Service Center.

[7] Ritter, H., Martinez, T., and Schulten, K. (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Addison-Wesley, Reading, MA.