Probabilistic Topological Map and Binary data

Mustapha Lebbah[1,2], Christian Chabanon[1], Sylvie Thiria[2], and Fouad Badran[3]

[1] RENAULT, Direction de la recherche
TCR RUC T 62, 1 avenue du golf F 78288 Guyancourt Cedex
{Mustapha.Lebbah, Christian.Chabanon}@Renault.com
[2] Laboratoire LODYC, Université Paris 6, Tour 26-4^{e}étage,
4 place Jussieu 75252 Paris cedex 05 France.
{lebbah, thiria}@lodyc.jussieu.fr
[3] CEDRIC, Conservatoire National des Arts et Métiers,
292 rue Saint Martin, 75003 Paris, France
{badran}@cnam.fr

Abstract. The Self Organizing Map (SOM) proposed by Kohonen [7] is a well known neural model which provides both quantization and clustering of the observation space. In this paper, we adapt the Bernoulli mixture approach, proposed by [6], to the model of binary topological map [2] and show that using a probabilistic formalism gives rise to better quantization process and classification performances.

1 Introduction

The Self Organizing Map (SOM) proposed by Kohonen [7] is a well known neural model which provides both quantization and clustering of the observation space. The formalism of the dynamic clustering is a general framework which leads to the batch version of SOM. In the paper [2], we have presented a new algorithm (BinBatch) derived from the batch version of SOM which is dedicated to binary data. A Bernoulli mixture approach already exist which allows to perform cluster analysis on binary data using the maximum likelihood [6]. In this paper, we adapt the Bernoulli mixture approach to the model of binary topological map and show that using a probabilistic formalism gives rise to better quantization process and classification performances. The probabilistic formalism is adapted in order to take into account the topological order of the binary map. The probabilistic model of topological map (BinBatch-Pro) dedicated to binary data is tested on simulated data, the performances presented in section 5 show the interest of the present approach which provides homogeneous clusters.

2 General information about a binary data

Very often, a binary vector represents a coding of discrete features which have a finite, usually small, number of possible values. Let β^d be a binary data space and $\mathcal{A} = \{\mathbf{z}_i; i = 1, \ldots, N\}$ a subset of observations, where each observation $\mathbf{z}_i = (z_i^1, z_i^2, ..., z_i^d)$ is a binary vector of β^d. We introduce two different similarity index: the Hamming distance $\mathcal{H}$ and the index of Tanimoto $\mathcal{T}$. Their values will provide different ways to compare the binary vectors $\mathbf{z}_1$ and $\mathbf{z}_2$. The Hamming distance measures the number of mismatch between $\mathbf{z}_1$ and $\mathbf{z}_2$:

$$\mathcal{H}(\mathbf{z}_1,\mathbf{z}_2)=\sum_{j=1}^{d}|\mathbf{z}_1^j-\mathbf{z}_2^j| \tag{1}$$

The Hamming distance allows to compute the central value of any subset of binary observations, called the median center; the most important characteristic of the median center is to be a binary vector which has the same interpretation (same coding) as the observations [10]. The second similarity is the Tanimoto index:

$$\mathcal{T}(z_1,z_2)=\frac{a}{a+\mathcal{H}(\mathbf{z}_1,\mathbf{z}_2)} \tag{2}$$

where a is the number of coincident occurrence of 1 between the two vectors $\mathbf{z}_1$ and $\mathbf{z}_2$, $a=\sum_{j=1}^{d} z_1^j z_2^j$. The Tanimoto index represents the ratio of coincident occurrence of one to the number of informative components of $\mathbf{z}_1$ and $\mathbf{z}_2$. Its value stands for the compactness of a cluster: a large value indicates a compact cluster mean while a small value represents a scattered one.
In the following we will use the Hamming distance during the auto organization process provided by the BinBatch algorithm which is a topological map dedicated to binary data. The Tanimoto index will be used during the validation phase in order to appreciate the quality of the resulting quantization. We first present the BinBatch algorithm which is the first step of the probabilistic algorithm.

3 BinBatch Quantization

The standard Binbatch algorithm [2] consists of a discrete set $\mathcal{C}$ of neurons called the map. As for traditional topological maps [7], we use a map with a discrete topology defined by an indirect graph; usually it is a regular grid in one or two dimensions, we denote N_{neuron} the number of neurons in $\mathcal{C}$. For each pair of neurons (c,r) on the map, the distance $\delta(c,r)$ is defined as being the shortest path between c and r on the graph.
In the following, in order to control the neighborhood order, we introduce a Kernel positive function K ($\lim\limits_{|x|\to\infty} K(x)=0$) and its associated family K_T parametrized by T : $K_T(\delta)=[1/T]K(\delta/T)$.
The standard BinBatch algorithm, assigns each neuron c to its referent vector $\mathbf{w}_c$ in β^d. The set of parameters $\mathcal{W}=\{\mathbf{w}_c; c\in\mathcal{C}\}$, which fully determine the BinBatch map, have to be estimated from $\mathcal{A}$. This is done iteratively, minimizing a cost function:

$$\mathcal{E}^T(\mathcal{W})=\sum_{\mathbf{z}_i\in\mathcal{A}} d(\mathbf{z}_i,\phi^T(z_i)) \tag{3}$$

Where $\phi^T(\mathbf{z}_i)$ represents a particular neuron of $\mathcal{C}$ assigned to $\mathbf{z}_i$. In the following we choose for d and Φ

$$d(\mathbf{z}, c) = \sum_{r \in C} \mathcal{K}_{\mathcal{T}}(\delta(r, c))\mathcal{H}(\mathbf{z}, \mathcal{W}_r) \tag{4}$$

$$\phi^T(\mathbf{z}) = \arg\min_{c \in C} d(\mathbf{z}, c) \tag{5}$$

and we denote $P_{\Phi^T} = \{P_c, c \in C\}$ the partition of $\mathcal{A}$ associated to the assignment function Φ^T; each subset of observations assigned to a neuron c is defined as $P_c = \{\mathbf{z}_i, \Phi^T(z_i) = c\}$.
As for the Batch version of Kohonen algorithm [1], the minimization can be done using a dynamic cluster algorithm [3, 4] operating in two steps : an assignment step which assigns each observation $\mathbf{z}_i$ to one cell c using the assignment function defined in (5), followed by an optimization step which computes for each cell c the weighted median center $\mathbf{w}_c^T$ of the subset P_c. The median center $\mathbf{w}_c^T$ or referent of c is computed weighting each observation $\mathbf{z}_i$ of $\mathcal{A}$ by $\mathcal{K}_T(\delta(\phi^T(z_i), c))$.
The minimization of $\mathcal{E}(\mathcal{W})$ is run by iteratively performing the two steps with decreasing value of T until stabilization. At the end, $\mathbf{w}_c^T$ which share the same code with the observations can be decoded in the same way, allowing a symbolic interpretation of the topological map. The nature of the topological model reached at the end of the algorithm, the quality of the clustering (or quantization) and those of the topological order induced by the graph greatly depend on the first value of T (T^{max}), its final value (T^{min}) and the number of times (N_{iter}) where the BinBatch algorithm is run.

4 Binary clustering and Bernoulli mixtures of probabilities

4.1 Probabilistic Dynamic cluster

In this first approach, we no more consider the topological order. We assign to each neuron c a probability function f_c which admits as mean vector the referent value $\mathbf{w}_c$. We will assume that each component of $\mathbf{z}$ is an independent Bernoulli distribution with the same parameter ε_c, so:

$$f_c(\mathbf{z}, \mathbf{w}_c, \varepsilon_c) = \prod_{j=1..d} \varepsilon_c^{|z^j - w_c^j|}(1 - \varepsilon_c)^{1-|z^j - w_c^j|} \tag{6}$$

And we assume that the data are generated by a mixture of such distributions:

$$f(z, \mathcal{W}, \epsilon) = \sum_{c \in \mathcal{C}} p_c f_c(\mathbf{z}, \mathbf{w}_c, \varepsilon_c) \tag{7}$$

where $\sum_{c \in C} p_c = 1$ and $\epsilon = \{\varepsilon_c, c \in C\}$.
This modelization of Bernoulli distributions has been introduced by Govaert and Celleux in [6], where different variation of their formalism have been proposed. As in ([6],[8]), we will maximize the classification likehood criterion (CML) which

assumes that the data set $\mathcal{A}$ is partitioned in N_{neuron} clusters and that every data $\mathbf{z}_i \in P_c$ is generated by the probability function f_c.
The logarithm of the associated CML criterion can be written:

$$\mathcal{C}(\mathcal{P},\mathcal{W},\epsilon) = \sum_{c\in\mathcal{C}} \sum_{\mathbf{z}_i\in P_c} \ln[\varepsilon_c^{\mathcal{H}(\mathbf{z_i},w_c)}(1-\varepsilon_c)^{d-\mathcal{H}(\mathbf{z_i},w_c)}] \tag{8}$$

Hence

$$\mathcal{C}(\mathcal{P},\mathcal{W},\epsilon) = \sum_{c\in C} \sum_{\mathbf{z}_i\in P_c} [\ln(\frac{\varepsilon_c}{1-\varepsilon_c})\mathcal{H}(\mathbf{z}_i,w_c) + d\ln(1-\varepsilon_c)] \tag{9}$$

The relation (9) shows that maximizing $\mathcal{C}(\mathcal{P},\mathcal{W},\epsilon)$ with respect to ϵ and $\mathcal{W}$ can be performed separately. Determining the parameters of the models: $\mathcal{W},\epsilon$ and χ require to minimize $-\mathcal{C}(\mathcal{P},\mathcal{W},\epsilon)$ with respect to $\mathcal{W}$, ϵ and the partition $\mathcal{P}$.

$$-\mathcal{C}(\mathcal{P},\mathcal{W},\epsilon) = \sum_{c\in C} \sum_{z_i\in P_c} [\ln(\frac{1-\varepsilon_c}{\varepsilon_c})\mathcal{H}(\mathbf{z}_i,w_c) - d\ln(1-\varepsilon_c)] \tag{10}$$

In this context we can use also the dynamical clustering formalism [3]. The two steps for the (t+1)th iteration is:

- **The assignment step**. Assume that the referents vectors $\mathcal{W}^t$ and ε^t have been computed at the tth iteration and are fixed. The relation (10) shows that the optimal assignment function χ^{t+1} is defined by:

$$\chi^{t+1}(z_i) = \arg\min_{c\in\mathcal{C}}[(\ln(\frac{1-\varepsilon_c^t}{\varepsilon_c^t})\mathcal{H}(z_i,w_c^t) - d\ln(1-\varepsilon_c^t))] \tag{11}$$

  In this case, the new partition P^{t+1} is defined by :

$$P_c^{t+1} = \{\mathbf{z}_i/\chi^{t+1}(\mathbf{z}_i) = c\} \tag{12}$$

- **The minimization step**. Assume that the assignment function χ^{t+1} defined at the preceding step is fixed. The minimization of (10) with respect to the parameters $\mathcal{W}$ and ϵ gives :

$$\mathbf{w}_c^{t+1}\text{the median center of } P_c^{t+1}$$

$$\varepsilon_c^{t+1} = \frac{\sum_{\mathbf{z}_i\in P_c^{t+1}} \mathcal{H}(\mathbf{z}_i,w_c^{t+1})}{n_c d} \tag{13}$$

  where n_c is the number of observation $\mathbf{z}_i$ assigned to neuron c.

If we initialize the procedure with the parameters ($\mathcal{W}^0,\varepsilon^0$ and χ^0) provided by BinBatch and run the two steps iteratively until convergence; the algorithm gives a new partition and a new set of parameters. At the end of learning phase the BinBatch map has a probabilistic interpretation, each neuron being a Bernoulli distribution.

4.2 BinBatch-Pro Algorithm

In this section we introduce the BinBatch-Pro algorithm which provides the probabilistic topological map. BinBatch-Pro adapts the preceding algorithm (section 4.1) in order to take into account the topological order given by the BinBatch map. This adaptation will concern mainly the assignment function (11) which will be replaced by:

$$\chi^{t+1}(\mathbf{z}_i) = \arg \min_{c \in V_{\chi^t(\mathbf{z}_i)}} (\ln(\frac{1-\varepsilon_c^t}{\varepsilon_c^t})\mathcal{H}(z_i, w_c^t) - d\ln(1-\varepsilon_c^t)) \tag{14}$$

Where $V_c = \{r, \delta(c,r) \leq 1\}$.
The minimization step remains the same as described in section 4.1. But an observation $\mathbf{z}_i$ which belongs to neuron c at the begining of the BinBatch-Pro adaptation, can only be assigned with BinBatch-Pro to a neuron in the vicinity V_c of c. This modification allows the algorithm to keep the topological order of the map. So BinBatch-Pro algorithm is run taking as initial parameters $\mathcal{W}^0$, ϵ^0, χ^0 provided by BinBatch. The new set of referent vectors given by BinBatch-Pro presents an order similar to the BinBatch one.
In the following, we validate the quality of a quantization by computing the mean Tanimoto index of each subset P_c.

$$T_{P_c} = \frac{\sum\limits_{x_i x_j \in P_c, x_i \neq x_j} \mathcal{T}(\mathbf{z}_i, \mathbf{z}_j)}{\frac{1}{2} n_c (n_c - 1)} \tag{15}$$

If the value of T_{P_c} is close to 1, the observation of P_c are very similar, they are the same if $T_{P_c} = 1$ exactly. The map provided by BinBatch algorithm and BinBatch-Pro, can be used in classification task. This can be done by using the majority vote rule and labelling each neuron c of the map. At the end of the labelling process the set of neurons C_i which have the same label l_i corresponds to different density functions which approximate the probability density function of class l_i. A new pattern $\mathbf{z}$ can be classified by computing the a posteriori probability of each class l_i:

$$p(l_i/\mathbf{z}) = \frac{\sum_{c \in C_i} n_c f_c(\mathbf{z}, \mathbf{w}_c, \varepsilon_c)}{\sum_{c \in C} n_c f_c(\mathbf{z}, \mathbf{w}_c, \varepsilon_c)} \tag{16}$$

Using these probabilities allows to perform Bayes classification.

5 Experiments

In the following, we illustrate the behavior of BinBatch-Pro on two different examples. The first one is a toy problem where the observations have been generated according to a three component Bernoulli mixture with equal proportion, ($p_c = \frac{1}{3}$); this experiment shows the ability of BinBatch-Pro to increase the classification performances. The second experiment, taken from the literature allows to compare different algorithms and to show the quality of the clustering according to the Tanimoto index.

5.1 Simulated data

The learning data set is a sample of 1500 observations described by 12 binary variables drawn from a three-component Bernoulli mixture with equal proportion. The sample is drawn according to (7)

$$f(\mathbf{z}) = \sum_{i=1}^{3} \frac{1}{3} f_i(\mathbf{z}, \mathbf{w}_i, \varepsilon_i)$$

with the w_i^j and ε_i given in table 1. So we simulated a classification task, each class made of 500 observations for the learning data set ; we simulated a test set of 300 observations according to the same distributions as the learning set (100 for each class).

var component	1	2	3	4	5	6	7	8	9	10	11	12	ϵ
1	0	0	0	0	1	1	1	1	1	1	1	1	0.4
2	0	0	0	0	1	1	1	1	0	0	0	0	0.3
3	1	1	1	1	0	0	0	0	1	1	1	1	0.2

Table 1. The parameters w_i^j of the samples

We train a 8×8 BinBatch map using the learning data set and apply BinBatch-Pro at the end of the learning phase. We summarize the behavior of the two algorithms using the number of misclassified observations. Table (2) presents the misclassification error rate computed on the test set for the two methods, BinBatch and BinBatch-Pro, using formula (16). We compute also the misclassification rate of the Bayes classifier which is equal to 26.33% on the test set. in this case. Beside the increase in performances, BinBatch-Pro provides estimations of the posteriori probabilities which can be used later on when dealing with real application.

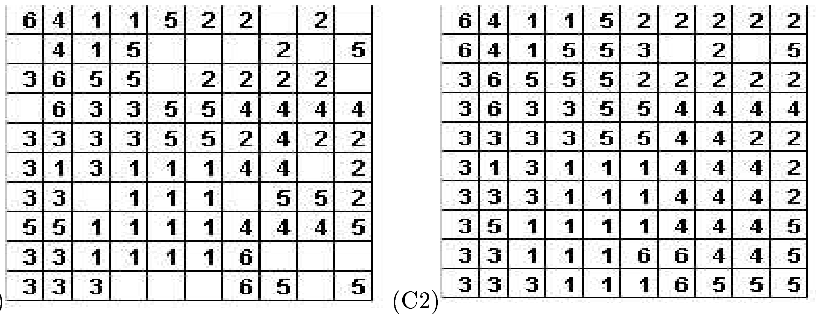
method	misclassification rate
BinBatch	30.67
BinBach-pro	29.00

Table 2. Misclassification error rate

5.2 The pool experiment

The second example is taken from [5, 9]. In this example, each observation $\mathbf{z}_i$ represents the answer of an individual to a pool. The pool uses 4 latent variables, each one made of 3 manifest variables. So one observation $\mathbf{z}$ consists of a

12 binary vector which represents a given individual. A latent variable describes, for example, the general interest in cultural activities during holidays; it is represented by 3 manifest variables representing the interest in museum, theater or opera. The total data set is made of 6 classes of individuals sharing the same interests, each one made of 1000 observations. For our experiment, we use as training set $\mathcal{A}$, 5000 individuals (or observations) taken at random. The test set is made using the 1000 remaining observations. We train a two dimensional 10×10 map using the Binbatch algorithm. We denote C_1 the BinBatch map obtained at the end of the learning phase. Then BinBatch-Pro was applied to C_1 giving rise to a second map C_2. Figure 1 shows the two maps C_1 and C_2, each neuron being labeled using the learning data set and using the majority vote. For the two algorithms, the topological order allows a good visualization of the 6



(C1)

6	4	1	1	5	2	2		2	
	4	1	5				2		5
3	6	5	5		2	2	2	2	
	6	3	3	5	5	4	4	4	4
3	3	3	3	5	5	2	4	2	2
3	1	3	1	1	1	4	4		2
3	3		1	1	1		5	5	2
5	5	1	1	1	1	4	4	4	5
3	3	1	1	1	1	6			
3	3	3				6	5		5

(C2)

6	4	1	1	5	2	2	2	2	2
6	4	1	5	5	3		2		5
3	6	5	5	5	2	2	2	2	2
3	6	3	3	5	5	4	4	4	4
3	3	3	3	5	5	4	4	2	2
3	1	3	1	1	1	4	4	4	2
3	3	3	1	1	1	4	4	4	2
3	5	1	1	1	1	4	4	4	5
3	3	1	1	1	6	6	4	4	5
3	3	3	1	1	1	6	5	5	5

Fig. 1. (C1)Learning done with BinBatch algorithm. (C2)Learning done with BinBatch-Pro. Each cell of the grid is a neuron of the map. The number presented in each neuron gives the label of the neuron provided by the majority vote rule

different classes, but clearly BinBatch-Pro allows to obtain a better clustering; in C_1 20 neurons are empty and only 2 in C_2. We compute the mean Tanimoto index for each partition provided by BinBatch and BinBatch-Pro. Figure 2 shows the comparisons cluster by cluster; it can be seen that most of the time the mean Tanimoto index is greater for the neurons of C_2 indicating more compact clusters. Finally we compute the classification performances, for BinBatch, we find a misclassification error rate of 18.32% and for BinBatch-Pro 18%. This is an accordance with the performances reached in the benchmark proposed by [5], where error rate of 18% reached by the different algorithm (Hard Competetive Learning, Neural Gas, SOM), the traditional k-means algorithm giving a 36% error rate. In this case, the major advantage of BinBatch and BinBatch-Pro is to provide a very precise quantization. The data space is divided in 92 clusters allowing a symbolic interpretation of the referents, while each algorithm of the comparison only deals with 6 clusters.

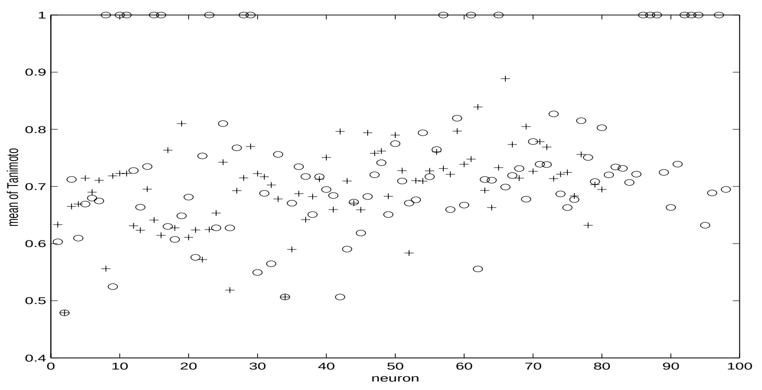


Fig. 2. Mean tanimoto of each neuron on the two grids. The circle point presents the variation of the index on BinBatch-Pro map. The sign '+' presents the value of the index given by Binbatch map.

6 conclusion

In this paper we show that a probabilistic formalism adapted to binary topological map allows to provide accurate quantization and classification. The topological order of the map giving useful insight of the existing topological relation ship between the clusters. The introduction of the probabilities on the topological maps improve the quality of the representativity of the map.

References

1. Anouar, F. Badran, F. Thiria, S: Probabilistic self-organizing map and radial basis function networks. Neurocomputing 20, 83-96. (1998).
2. Lebbah, M. Badran, F. Thiria: S Topological Map for Binary Data ,ESANN 2000 , Bruges, April 26-27-28, 2000, Proceedings
3. Diday, E. C,Simon: Clustering Analysis, in :K.S. Fu(ED), Digital pattern recognition, Springer,New York.An Introduction to Symbolic Data. (1996).
4. Diday E. and Simon J.C.: Clustering Analysis. In Digittal Pattern Recognition. Edited by K.S.FU. Springer-Verlag (1976)
5. Dolinicar, Weingessel , A. Buchta, C. Dimitriadou, E: A Comparison of several cluster algorithms on artificial binary data, scenarios from travel market segmentation. Working paper series 19, SFB (adaptive information systems and modelling in economics and management science. (1998).
6. Celeux, G. Govaert, G: Clustering criteria for discrete data and latent class Models. Journal of classification 8:157-176(1991)
7. Kohonen, T: Self-Organizing Map. Springer, Berlin.(1994).
8. Nadif, M. Govaert, G: Clustering for Binary data and mixture Models- Choice of the model. Appl. Stochastic Models Data Anal. 13,269-278(1998).
9. Leich, F. Weingessel, A. Dimitriadou, E: Competitive Learning for Binary Data. Proc of ICANN'98, septembre 2-4. Springer Verlag. (1998).
10. Marchetti, F: Contribution à la classification de données binaires et qualitatives, thèse de l'université de Metz.(1989).