Sparse online self-organizing maps for large relational data

Madalina Olteanu, Nathalie Vialaneix

Sparse online self-organizing maps for large relational data

Madalina Olteanu[1] and Nathalie Villa-Vialaneix[2]

[1] SAMM - Université Paris 1 Panthéon-Sorbonne
90, rue de Tolbiac, 75013 Paris, France
madalina.olteanu@univ-paris1.fr
[2] INRA, UR 0875 MIAT
BP 52627, F-31326 Castanet Tolosan, France
nathalie.villa@toulouse.inra.fr

Abstract. During the last decades, self-organizing maps were proven to be useful tools for exploring data. While the original algorithm was designed for numerical vectors, the data became more and more complex, being frequently too rich to be described by a fixed set of numerical attributes. Several extensions of the original SOM were proposed in the literature for handling kernel or dissimilarity data. Most of them use the entire kernel/dissimilarity matrix, which requires at least quadratic complexity and becomes rapidly unfeasible for 100 000 inputs, for instance. In the present manuscript, we propose a sparse version of the online relational SOM, which sequentially increases the composition of the prototypes.

Keywords: relational data, online relational SOM, sparse approximations.

1 Introduction

The self-organizing map (SOM) algorithm, [1], was proven, over the years, to be a powerful and convenient tool for clustering and visualizing data. While the original algorithm was designed for numerical vectors, the available data in the applications became more and more complex, being frequently too rich to be described by a fixed set of numerical attributes only. This is the case, for example, when data are described by relations between objects (individuals involved in a social network) or by measures of ressemblance/dissemblance (professional trajectories).
During the past twenty years, the SOM algorithm was extended for handling relational data, either described by kernels (see [2] for the online version and [3] for the batch version), or by dissimilarities (see [4] for the online version and [5] for the batch version). All these extensions are based on the same underlying principle : the dissimilarity or the kernel implicitly define an Euclidean (or pseudo-Euclidean) space in which the prototypes can be expressed as convex combinations of the embedded input data. However, when the goal is to explore

large data sets, the relational approaches may become rapidly unfeasible. Indeed, complex relational data often have a large dimensionality. Moreover, kernel and relational SOM rely on the knowledge of the dissimilarity matrix for the entire data set, which generates at least quadratic complexity for the algorithms. As stressed in [5], algorithms will be slow for data sets with 10,000 observations and impossible to run on a normal computer for 100,000 input data. In addition to the complexity issue, expressing prototypes as convex combinations of the entire data set has a second drawback, as emphasized in [6]: the interpretability of the prototypes and of the model is lost.

In order to tackle these two issues, several approaches were introduced for relational data, all of them seeking for a sparse representation of the prototypes and a linear (in the number of observations) computational cost. [7] use the natural sparsity of the prototypes in batch relational k-means in order to reduce the complexity. The natural sparsity is enhanced by selecting the K (K fixed) closest inputs to each prototype. In [5], the complexity is reduced using iterative "patch clustering". First, the data are split into P patches of size n_P (P fixed). A prototype-based clustering algorithm in batch version (neural gas or SOM) is then run on a patch P_t and the resulting prototypes, which may be viewed as compressed representations of the data already seen, are added as new data points to the next patch, P_{t+1}. Moreover, the full vector of coefficients is replaced by the K closest input data (K fixed). With this method, linear time and constant space representation are obtained. Another technique consists in using the Nyström approximation ([8]) for the dissimilarity matrix. This technique also leads to a linear computational cost in the number of input data, but is strongly dependent on the intrinsic dimensionality of the given dissimilarity matrix, which has to be of low rank and entirely known in advance. All these cited approaches are batch algorithms.

In the online framework, [9] propose a bagging approach for kernel SOM. Data is split into B subsamples of size n_B (B fixed), the online kernel SOM is trained on each subsample and, after training, the most representative K observations are chosen for each prototype (K fixed). Eventually, a final map is trained on the resulting most representative observations. The algorithm has the advantage of being parallelizable, although it does not consider all the advantages of an online implementation.

In the present paper, we propose a sparse version of the online relational SOM algorithm, which takes further advantage of the online setting. Instead of expressing prototypes as convex combinations of the entire data set from the beginning, the size and the composition of the prototypes are sequentially increased with each new input fed to the algorithm. When the size of the prototypes becomes too large, prototypes are made sparse by deleting all the insignificant coefficients. Different approaches for selecting the most interesting observations are reported in [6]. In this manuscript, we use a slightly different technique, by interpreting the coefficients as a probability distribution and by selecting the most probable observation: a global probability mass ν is fixed and the largest coefficients summing to ν are kept. In this way, more flexibility is allowed to the prototypes

which are no longer represented by a fixed number K of observations, but by the necessary number of observations allowing an "almost complete" knowledge of the composition of the prototypes (if ν is chosen close to 1).
The rest of the paper is organized as follows: Section 2 recalls the online relational SOM, while Section 3 introduces the sparse version of the online relational SOM. The equivalent algorithm for kernels is briefly described in Section 4, while Section 5 contains some examples on real data-sets.

2 Online relational SOM

In this section we shall briefly recall the principles of the online relational SOM (RSOM) algorithm, as introduced in [4]. Throughout the rest of the paper, let us suppose that the input data, $x_1, ..., x_N$, belong to some arbitrary space $\mathcal{G}$ and can be described through a dissimilarity measure δ, such that $\delta_{ij} = \delta(x_i, x_j)$. The dissimilarity measure is supposed to verify some basic assumptions: symmetry ($\delta_{ij} = \delta_{ji}$) and non-negativity ($\delta_{ij} \geq 0$), for all $i, j = 1, ..., N$, and also $\delta_{ii} = 0$, for all $i = 1, ..., N$.
The online RSOM algorithm aims at mapping the input data onto a low dimensional grid (usually a two-dimensional rectangle), composed of U units, each of them described by a prototype p_u, $u = 1, ..., U$. The units are linked together by a neighborhood relationship H, expressed as a function of the distance between the units on the grid, $d(u, u')$. The distance on the grid, d, may be chosen, for example, as the length of the shortest path between the units. The U prototypes are initialized either as random convex combinations of the input data or randomly among the input data.
The extension of the original SOM algorithm is based on two key ideas:

- First, prototypes are written as (symbolic) convex combinations of the input data, $p_u = \sum_{i=1}^{N} \beta_{u,i} x_i$, with $\beta_{u,i} \geq 0$ and $\sum_{i=1}^{N} \beta_{u,i} = 1$, for all $u = 1, ..., U$. This definition is justified by the fact that, when a dissimilarity is given, it can be viewed as the dot product of the images by a mapping function ϕ into a pseudo-Euclidean space [10]: the prototypes are thus truly the convex combinations of $(\phi(x_i))_i$ in this space (see [5, 4] for further explanations).
- Second, the distance between an input data x_i and a prototype p_u can be written only in terms of the dissimilarity matrix of the input data and the coefficients $\beta_{u,i}$ as follows:

$$\|x_i - p_u\|^2 = \mathbf{\Delta}_i \beta_u^T - \frac{1}{2} \beta_u \mathbf{\Delta} \beta_u^T \ , \tag{1}$$

  where $\mathbf{\Delta} = (\delta_{ij})_{i,j=1,...,N}$, $\mathbf{\Delta}_i$ represents the i-th row of the matrix $\mathbf{\Delta}$ and $\beta_u = (\beta_{u,1}, ..., \beta_{u,N})$ is the vector of coefficients for the prototype p_u.

Expressing the prototypes as convex combinations of the input data and computing the distances between observations and prototypes as in Equation (1) consists, in fact, in a generalization of the original SOM algorithm. Indeed, one

can easily see that the two are equivalent if the dissimilarity δ is the squared Euclidean distance and if the prototypes of the original SOM are initialized within the convex hull of the input data.
This general framework allowing an elegant writing of the algorithm for complex data described by dissimilarities was introduced initially for the online version of kernel SOM (KSOM) in [2]. Afterwards, extensions and rediscoveries were described for batch relational SOM in [5], batch kernel SOM in [3] and online relational SOM in [4]. A detailed and complete comparison of these methods and their equivalences may be found in [11].
The distance computation in Equation (1) may be theoretically justified in the very general setting of dissimilarities by extending the Hilbert embedding for kernels to a pseudo-Euclidean embedding, as shown, for example, in [5].
The online relational SOM algorithm is summarized in Algorithm 1. The neighborhood function H is supposed to verify the following assumptions: $H : \mathbb{R} \to \mathbb{R}$, $H(0) = 1$ and $\lim_{x\to+\infty} H(x) = 0$. In the setting of Algorithm 1, H^t decreases piecewise linearly, while $\mu(t)$ vanishes at the rate $\frac{1}{t}$.

Algorithm 1 Online relational SOM

1: For all $u = 1, \ldots, U$ and $i = 1, \ldots, N$, initialize β_{ui}^0 such that $\beta_{u,i}^0 \geq 0$ and $\sum_i^N \beta_{u,i}^0 = 1$.
2: **for** $t = 1, \ldots, T$ **do**
3: Randomly choose an input x_i
4: *Assignment step*: find the unit of the closest prototype

$$f^t(x_i) \leftarrow \arg\min_{u=1,\ldots,U} \left[\mathbf{\Delta}_i \left(\beta_u^{t-1}\right)^T - \frac{1}{2} \beta_u^{t-1} \mathbf{\Delta} (\beta_u^{t-1})^T \right]$$

5: *Representation step*: $\forall\, u = 1, \ldots, U$,

$$\beta_u^t \leftarrow \beta_u^{t-1} + \mu(t) H^t(d(f^t(x_i), u)) \left(\mathbf{1}_i - \beta_u^{t-1}\right)$$

where $\mathbf{1}_i$ is a vector with a single non null coefficient at the ith position, equal to one.
6: **end for**

3 Sparse online relational SOM

Similarly to relational SOM, prototypes are written as convex combinations of the observations, but, in this case, they are restricted to the input data already fed to the algorithm and, more particularly, to the most significant of them. In order to guarantee the sparsity of the writing as well as similar properties with the original online relational SOM algorithm, several issues have to be verified.

1. Prototypes have to be initialized at random among the input data. Hence, the observations have to be randomly presented to the algorithm. The first U observations will be then used as initial values for the U prototypes.
2. The dissimilarity between a new input data and a prototype, written as a convex combination of the most significant past observations, has to be computed. This can be achieved using the following formula $\|x_k - p_u\|^2 = \sum_{j\in I(t)} \beta_{u,j}\delta(x_k, x_j) - \frac{1}{2}\sum_{i\in I(t)}\sum_{j\in I(t)} \beta_{u,i}\beta_{u,j}\delta(x_i, x_j)$, where $p_u = \sum_{j\in I(t)} \beta_{u,j}x_j$ and $I(t)$ contains the indices of the most significant inputs already fed to the algorithm before x_k is chosen.
3. Prototypes are sparse combinations of the input data. Hence, prototypes are periodically updated and the most coefficients only are selected. The updates may be performed throughout the iteration using either a deterministic design (the number of updates is fixed and updates are uniformly distributed during the learning of the map), or a random design (the updates are distributed according to some geometric distribution. The parameter of the geometric distribution may depend on the total number of iterations and on the size of the neighborhood). Sparsity could be achieved by selecting the first Q most important coefficients, where Q is a fixed integer. However, in order to allow for more flexibility in the expression and interpretability of the prototypes, the most significant coefficients are selected according to their value, by fixing a threshold: let $0 < \nu \leq 1$ be the selected threshold (if $\nu = 1$, the algorithm is no longer sparse, but the original one).
   For $u = 1, ..., U$, the coefficients are ordered in descending order for each prototype $\beta_{u,(1)}, ..., \beta_{u,(\sharp I(t))}$, where $\beta_{u,(1)} = \max_{i\in I(t)} \beta_{u,i}$ and $\beta_{u,(\sharp I(t))} = \min_{i\in I(t)} \beta_{u,i}$. Consider N_u such that $N_u = \arg\min_{n=1,...,\sharp I(t)} \left\{\sum_{i=1}^{n} \beta_{u,(i)} \geq \nu\right\}$. The most significant coefficients are updated as follows

$$\beta_{u,(i)} = \begin{cases} \frac{\beta_{u,(i)}}{\sum_{j=1}^{N_u} \beta_{u,(j)}} \text{ , if } (i) \leq N_u \\ 0 \text{ , if } (i) > N_u \end{cases}$$

The sparse online relational SOM algorithm is summarized in Algorithm 2.

4 The kernel version

In some cases, data may be described by a kernel, K, instead of a dissimilarity. We shall recall that a kernel is a symmetric similarity such that $K(x_i, x_i) = 0$ and which satisfies the following positive constraint: $\forall M > 0$, $\forall (x_i)_{i=1,...,M} \in \mathcal{G}$, $\forall (\alpha_i)_{i=1,...,M} \in \mathbb{R}$, $\sum_{i,j=1}^{M} \alpha_i\alpha_j K(x_i, x_j) \geq 0$. According to [12], there exists a Hilbert space $\mathcal{H}$, also called feature space, as well as a feature map $\psi : \mathcal{G} \to \mathcal{H}$, such that $K(x, x') = \langle\psi(x), \psi(x')\rangle_{\mathcal{H}}$. Similarly to the dissimilarity case, the prototypes are defined as convex combinations of (the images by ψ of) $(x_i)_i$. The distance between an input data x_k and some prototype p_u is then computed as the squared distance induced by the kernel

Algorithm 2 Sparse online RSOM

For all $u = 1, \dots, U$, initialize p_u^0 among the first U input data: $\beta_u^0 = 1_u^U$, where 1_u^U is a vector of length U with a single non-null coefficient on the u-th position, equal to 1. Initialize $I(0) = \{1, ..., U\}$.

for $t = 1, \dots, T$ **do**

Randomly choose an input x_k, $k \in \{1, ..., N\}$.

Assignment step: find the unit of the closest prototype

$$f^t(x_k) \leftarrow \arg\min_{u=1,\dots,U} \left[\sum_{j \in I(t-1)} \beta_{u,j}^{t-1} \delta(x_k, x_j) - \frac{1}{2} \beta_u^{t-1} \mathbf{\Delta}_{I(t-1)} \left(\beta_u^{t-1}\right)^T \right] ,$$

where $\mathbf{\Delta}_{I(t-1)} = (\delta(x_i, x_j))_{i,j \in I(t-1)}$.

Representation step: $\forall\, u = 1, \dots, U$

if $k \in I(t-1)$, **then**

$\beta_u^t \leftarrow \beta_u^{t-1} + \mu(t) H^t(d(f^t(x_k), u)) \left(\mathbf{1}_k - \beta_u^{t-1}\right)$

$I(t) = I(t-1)$

else if $k \notin I(t-1)$, **then**

$\beta_u^t \leftarrow \left[1 - \mu(t) H^t(d(f^t(x_k), u))\right] \left(\beta_u^{t-1}, 0\right) + \mu(t) H^t(d(f^t(x_k), u)) (\underbrace{0, ..., 0}_{\sharp I(t-1)}, 1)$

$I(t) = I(t-1) \cup \{k\}$.

end if

Sparse representation:

if t is an update instant (deterministic or random design) **then**

Sparsely update the prototypes: $\forall\, u = 1, \dots, U$,

$$\beta_{u,(1)}^t \geq ... \geq \beta_{u,\sharp I(t)}^t ,$$

$$N_{t,u} = \arg\min_{n=1,\dots,\sharp I(t)} \left\{ \sum_{i=1}^{n} \beta_{u,(i)}^t \geq \nu \right\}$$

$$\beta_{u,(i)}^t = \begin{cases} \frac{\beta_{u,(i)}^t}{\sum_{j=1}^{N_{t,u}} \beta_{u,(j)}^t} \text{ , if } (i) \leq N_{t,u} \\ 0 \text{ , if } (i) > N_{t,u} \end{cases}$$

end if

end for

$\|x_k - p_u\|^2 = K(x_k, x_k) - 2\sum_{i\in I(t)} \beta_{u,i} K(x_k, x_i) + \sum_{i,j\in I(t)} \beta_{u,i}\beta_{u,j} K(x_i, x_j)$. The sparse online relational SOM can thus be immediately adapted for kernels. Algorithm 2 has to be modified only in the assignment step which becomes

1: *Assignment step*: find the unit of the closest prototype

$$f^t(x_k) \leftarrow \arg\min_{u=1,\ldots,U} \left[\beta_u^{t-1} \mathbf{K}_{I(t-1)} \left(\beta_u^{t-1}\right)^T - 2 \sum_{j\in I(t-1)} \beta_{u,j}^{t-1} K(x_k, x_j) \right] ,$$

where $\mathbf{K}_{I(t-1)} = \left(K(x_i, x_j)\right)_{i,j\in I(t-1)}$.

5 Examples

The sparse version introduced in the present manuscript was compared to the online relational SOM on two real data sets. For the sparse version, several values were considered for the threshold ν. The sparse updates were performed either in a uniform deterministic design (fixed number of updates), or at random, according to a geometric distribution. The performances of the sparse RSOM and the online RSOM were then compared in terms of average computational time (in seconds), quantization and topographic errors and sparsity (number of non-zero coefficients). Scripts were all implemented under the free statistical software environement R.

nb. updates	ν	comp. time (s)	quantization err.	topographic err.	nb. coefs
50	0.80	2.04	0.00087	0.0339	5.87
50	0.85	2.13	0.00076	0.0157	7.65
50	0.90	2.37	**0.00067**	0.0077	12.07
50	0.95	2.91	**0.00064**	0.0067	23.45
50	0.99	4.14	**0.00067**	0.0055	46.80
25	0.80	2.76	**0.00067**	0.0167	12.58
25	0.85	3.48	**0.00065**	0.0139	17.13
25	0.90	3.17	**0.00065**	0.0128	22.99
25	0.95	3.61	**0.00064**	0.0107	34.99
25	0.99	4.69	0.00070	**0.0041**	53.75
10	0.80	7.04	**0.00066**	0.0079	40.09
10	0.85	6.96	**0.00065**	0.0087	43.08
10	0.90	7.55	**0.00067**	0.0075	47.93
10	0.95	7.87	**0.00065**	0.0055	57.55
10	0.99	8.52	0.00068	0.0054	68.15
Online RSOM		12.18	**0.00067**	**0.0051**	

Table 1. Average results for **Astraptes fulgerator** (100 random initializations). The first column contains the number of updates (deterministic design). The third column is the computational time (provided in seconds). The last column is the average number of non zero coefficients in the prototypes. The bolded values correspond to the results at least as good as the online RSOM.

Astraptes fulgerator. The first data set was introduced in [13]. In contains information on 465 Amazonian butterflies, each of them described by a sample of their DNA. Each input data is a DNA sequence of length 350. The Kimura distance for genetical sequences, as introduced in [14], was computed and the resulting distance matrix was used as input for relational and sparse relational SOM. For both algorithms, 100 different initializations with 2 500 iterations each were performed on a square grid of size 5×5. The results are summarized in Tables 1 and 2 for the deterministic and random designs respectively.

nb. updates	ν	comp. time (s)	quantization err.	topographic err.	nb. coefs
50	0.80	1.92	0.00093	0.0353	5.44
50	0.85	2.09	0.00078	0.0176	7.35
50	0.90	2.37	0.00069	0.0145	11.02
50	0.95	2.92	**0.00067**	0.0102	21.75
50	0.99	4.02	0.00068	0.0068	45.51
25	0.80	2.50	**0.00067**	0.0210	9.92
25	0.85	2.88	**0.00066**	0.0114	14.09
25	0.90	2.94	**0.00066**	0.0107	20.41
25	0.95	3.56	**0.00064**	0.0057	29.63
25	0.99	4.66	**0.00066**	0.0053	51.93
10	0.80	4.23	**0.00062**	0.0132	22.48
10	0.85	4.69	**0.00065**	0.0072	28.41
10	0.90	5.18	**0.00065**	0.0098	33.97
10	0.95	5.14	**0.00065**	**0.0051**	43.34
10	0.99	6.30	**0.00067**	**0.0033**	59.95
Online RSOM		12.18	**0.00067**	**0.0051**	

Table 2. Average results for **Astraptes fulgerator** (100 random initializations, updates were made with a random design).

Professional trajectories. The second example comes from [15]. It contains information about 2 000 people having graduated high-school in 1998 and monitored during 94 months afterwards. For each individual, a categorical sequence of length 94, giving his monthly professional status is available. In all, there are nine possible situations, from permanent contracts to unemployment. The dissimilarity used for these data is the optimal matching (OM) distance, as introduced in [16]. Here, 100 different initializations with 10 000 iterations each were performed on a square grid of size 10×10. The sparse version was compared to the standard online relational SOM (itself run from 100 different initializations and 10 000 iterations). The results for the deterministic design are summarized in Table 3 (due to the lack of space, we do not report here the results with a random design, which are quite similar).

It is interesting to note that the sparsity has a strong influence on the computational time: increasing the number of updates tends to decrease the computational time since the prototypes are regularly cleared from unnecessary coeffi-

cients. The computational time compared to the standard version is at least 10 times smaller in the sparse version for this large dataset. On the contrary, the performances, measured in terms of quantization and topographic errors, can be affected by a too large sparsity but the best ones remain close to those of the standard version.

nb. updates	ν	comp. time (s)	quantization err.	topographic err.	nb. coefs
100	0.80	111	29.5	0.384	**1.4**
100	0.85	130	27.8	0.348	1.8
100	0.90	147	25.5	0.277	2.9
100	0.95	215	21.8	0.112	11.3
100	0.99	480	**20.5**	0.084	40.4
50	0.80	157	25.6	0.247	2.6
50	0.85	174	23.8	0.177	4.4
50	0.90	223	22.1	0.109	9.8
50	0.95	307	21.0	0.086	23.3
50	0.99	672	**20.5**	0.080	52.9
25	0.80	247	22.6	0.124	7.3
25	0.85	278	21.6	0.102	12.2
25	0.90	339	21.0	0.089	20.1
25	0.95	470	**20.5**	0.090	34.0
25	0.99	800	**20.6**	0.078	60.9
Online RSOM		9126	**20.7**	**0.075**	

Table 3. Average results for "professional trajectories" (100 random initializations, updates were made with a deterministic design). Simulations were all performed on a server with OS Debian 8 Jessie, 8 processors AMD Opteron 8384 with 4 cores each and 256 Go RAM.

6 Conclusion and future work

A sparse version of the online relational SOM algorithm was proposed, by sequentially increasing the composition of the prototypes and sparsely updating them. The algorithm was compared with the online ROM on two real data sets and the sparse version appeared to achieve very similar performances as compared to the original algorithm, while improving computational time and prototype representation.

References

1. Kohonen, T.: Self-Organizing Maps, 3rd Edition. Volume 30. Springer, Berlin, Heidelberg, New York (2001)
2. Mac Donald, D., Fyfe, C.: The kernel self organising map. In: Proceedings of 4th International Conference on knowledge-based Intelligence Engineering Systems and Applied Technologies. (2000) 317–320

3. Boulet, R., Jouve, B., Rossi, F., Villa, N.: Batch kernel SOM and related Laplacian methods for social network analysis. Neurocomputing **71**(7-9) (2008) 1257–1273
4. Olteanu, M., Villa-Vialaneix, N.: On-line relational and multiple relational SOM. Neurocomputing **147** (2015) 15–30
5. Hammer, B., Hasenfuss, A.: Topographic mapping of large dissimilarity data sets. Neural Computation **22**(9) (September 2010) 2229–2284
6. Hofmann, D., Schleif, F., Paaß en, B., Hammer, B.: Learning interpretable kernelized prototype-based models. Neurocomputing **141** (2014) 84–96
7. Rossi, F., Hasenfuss, A., Hammer, B.: Accelerating relational clustering algorithms with sparse prototype representation. In: Proceedings of the 6th Workshop on Self-Organizing Maps (WSOM 07), Bielefield, Germany, Neuroinformatics Group, Bielefield University (2007)
8. Gisbrecht, A., Mokbel, B., Hammer, B.: The nystrom approximation for relational generative topographic mappings. NIPS workshop on challenges of Data Visualization (2010)
9. Mariette, J., Olteanu, M., Boelaert, J., Villa-Vialaneix, N.: Bagged kernel SOM. In Villmann, T., Schleif, F., Kaden, M., Lange, M., eds.: Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014). Volume 295 of Advances in Intelligent Systems and Computing., Mittweida, Germany, Springer Verlag, Berlin, Heidelberg (2014) 45–54
10. Goldfarb, L.: A unified approach to pattern recognition. Pattern Recognition **17**(5) (1984) 575–582
11. Rossi, F.: How many dissimilarity/kernel self organizing map variants do we need? In Villmann, T., Schleif, F., Kaden, M., Lange, M., eds.: Advances in Self-Organizing Maps and Learning Vector Quantization (Proceedings of WSOM 2014). Volume 295 of Advances in Intelligent Systems and Computing., Mittweida, Germany, Springer Verlag, Berlin, Heidelberg (2014) 3–23
12. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society **68**(3) (1950) 337–404
13. Hebert, P.D.N., Penton, E.H., Burns, J.M., Janzen, D.H., Hallwachs, W.: Ten species in one: DNA barcoding reveals cryptic species in the neotropical skipper butterfly astraptes fulgerator. Genetic Analysis (2004)
14. Kimura, M.: A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. Journal of molecular evolution **16**(2) (December 1980) 111–120
15. Rousset, P., Giret, J.F.: Classifying qualitative time series with som: The typology of career paths in france. In Sandoval, F., Prieto, A., Cabestany, J., Graña, M., eds.: Computational and Ambient Intelligence. Volume 4507 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2007) 757–764
16. Abbott, A., Forest, J.: Optimal matching methods for historical sequences. Journal of Interdisciplinary History **16**(3) (1986) 471–494