Pattern Recognition of mtDNA with Associative Models

María Elena Acevedo, Marco Antonio Acevedo, Federico Felipe and David Aquino

Instituto Politécnico Nacional, Escuela Superior de Ingeniería Mecánica y Eléctrica, Mexico City, Mexico

Abstract. In this paper we applied an associative memory for the pattern recognition of mtDNA that can be useful to identify bodies and human remains. In particular, we used both morphological hetroassociative memories: *max* and *min*. We process the problem of pattern recognition as a classification task. Our proposal showed a correct recall, we obtained the 100% of recalling of all the learned patterns. We simulated a corrupted sample of mtDNA by adding noise of two types: additive and subtractive. The memory showed a correct recall when we applied less or equal than 55% of both types of noise.

1 Introduction

The scientific and technologic advances in the tools used in genetic have allowed new applications in other disciplines. More frequent, there is a collaboration between forensic groups and other medical specialities. Nowadays, the applications of computational algorithms in several areas have become essential. In this work, we applied artificial intelligent tools to create a software that is able to recognize patterns of mtDNA as a forensic application.

Mitochondria [1] are structures within cells that convert the energy from food into a form that cells can use. Although most DNA is packaged in chromosomes within the nucleus, mitochondria also have a small amount of their own DNA. This genetic material is known as mitochondrial DNA or mtDNA. This type of DNA resists adverse conditions without being degraded.

Some approaches are applied to analyse and recognize DNA. Craven and Shavlik [2] applied artificial neural networks to recognize promoters in DNA. A genetic algorithm was applied to find variable length motifs (transcription factor binding sites) [3]. A Self-Organizing Neural Network [4] was used to identify motifs. A related work with motifs applied the RISO software and box-link structures [5]. In addition, voting and matching pattern algorithms [6] were used to identify motifs. Wells and Sperling [7] used the software PAUP 4.0 to analyse and identify mtDNA of the blow fly subfamily of Chrysomyinae to help a forensic-entomological analysis. In other paper, a neural network based multi-classifier system [8] for the identification of Escheri-chia coli promoter sequences in strings of DNA is presented. Pushpalatha and Mukunthan [9] applied a Neural-fuzzy Mapping to identify DNA Finger Printing. A genetic algorithm [10] was used as a preprocessing tool to Identifying significant genes from DNA microarray. Inbamalar and Sivakumar [11] applied the DWT Coiflet 5 to detect the protein coding regions in DNA sequences.

In this work, we applied the Morphological Associative Memories to identify people from the mtDNA. We use the two types of heteroassociative memories: *max* and *min*. The results from these memories were mapping to binary numbers and then, we applied the AND operation between the two binary results.

2 Basic concepts

2.1 Associative memory

Associative Memories (AM) [12] associate patterns **x** with **y**, which can represent any concept: faces, fingerprints, DNA sequences, animals, books, preferences, diseases, etc. We can extract particular features of these concepts to form patterns **x** and **y**.

There are two phases for designing an associative memory: Training and Recalling.

In the Training Phase, the process of associate patterns **x** with patterns **y** is performed. Now, we say that the memory is built.

The input and output patterns are represented by vectors. The task of association of these vectors is called Training Phase and the Recognizing Phase allows recovering patterns. The stimuli are the input patterns represented by the set $\mathbf{x} = \{x^1, x^2, x^3, \ldots, x^p\}$ where p is the number of associated patterns. The responses are the output patterns and are represented by $\mathbf{y} = \{y^1, y^2, y^3, \ldots, y^p\}$. Representation of vectors x^μ is $x^\mu = \{x_1^\mu, x_2^\mu, \ldots, x_n^\mu\}$ where n is the cardinality of x^μ. The cardinality of vectors y^μ is m, then $y^\mu = \{y_1^\mu, y_2^\mu, \ldots, y_m^\mu\}$. The set of associations of input and output patterns is called the fundamental set or training set and is represented as follows: $\{(\mathbf{x}^\mu, \mathbf{y}^\mu) \mid \mu = 1, 2, ..., p\}$

2.2 Morphological memories

The basic computations occurring in the proposed morphological network [13] are based on the algebraic lattice structure (**R**, ∨, ∧, +), where the symbols ∨ and ∧ denote the binary operations of maximum and minimum, respectively. Using the lattice structure (**R**, ∨, ∧, +), for an *m* x *n* matrix *A* and a *p* x *n* matrix *B* with entries from **R**, the matrix product $C = A \nabla B$, also called the *max* product of *A* and *B*, is defined by equation (1).

$$c_{ij} = \bigvee_{k=1}^{p} a_{ik} + b_{kj} = \left(a_{i1} + b_{1j}\right) \vee \ldots \vee \left(a_{ip} + b_{pj}\right) \quad (1)$$

The *min product* of A and B induced by the lattice structure is defined in a similar fashion. Specifically, the *i,j*th entry of $C = A \Delta B$ is given by equation (2).

$$c_{ij} = \bigwedge_{k=1}^{p} a_{ik} + b_{kj} = \left(a_{i1} + b_{1j}\right) \wedge \ldots \wedge \left(a_{ip} + b_{pj}\right) \quad (2)$$

Henceforth, let ($\mathbf{x}^1$, $\mathbf{y}^1$), ($\mathbf{x}^2$, $\mathbf{y}^2$), …, ($\mathbf{x}^p$, $\mathbf{y}^p$) be *p* vector pairs with $x^k = \left(x_1^k, x_2^k, \ldots, x_n^k\right)^t \in \mathbf{R}^n$ and $y^k = \left(y_1^k, y_2^k, \ldots, y_m^k\right)^t \in \mathbf{R}^m$ for k = 1, 2, …, *p*. For a given set of pattern associations $\{(\mathbf{x}^k, \mathbf{y}^k) \mid k = 1, 2, \ldots, p\}$ we define a pair of associated pattern matrices (**X**, **Y**), where $X=(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^p)$ and $Y=(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^p)$. Thus, **X** is of dimension *n* x *p* with *i,j*th entry x_i^j and *Y* is of dimension *m* x *p* with *i,j*th entry y_i^j. Since $\mathbf{y}^k \nabla (-\mathbf{x}^k)^t = \mathbf{y}^k \Delta (-\mathbf{x}^k)^t$, the notational burden is reduced by denoting these identical morphological outer vector products by $\mathbf{y}^k \times (-\mathbf{x}^k)^t$.

With these definitions, we present the algorithms for the training and recalling phase.

Training Phase

1. For each *p* association ($\mathbf{x}^\mu$, $\mathbf{y}^\mu$), the minimum product is used to build the matrix $\mathbf{y}^\mu \Delta (-\mathbf{x}^\mu)^t$ of dimensions *m* x *n*, where the input transposed negative pattern $\mathbf{x}^\mu$ is defined as $(-\mathbf{x}^\mu)^t = \left(-x_1^\mu, -x_2^\mu, \ldots, -x_n^\mu\right)$.
2. The maximum and minimum operators (∨ and ∧) are applied to the *p* matrices to obtain *M* and *W* memories as equations (3) and (4) show.

$$M = \bigvee_{k=1}^{p} \left(y^k \otimes \left(-x^k\right)^t\right) \quad (3)$$

$$W = \bigwedge_{k=1}^{p} \left(y^k \otimes \left(-x^k\right)^t\right) \quad (4)$$

Recalling phase

In this phase, the minimum and maximum product, Δ and ∇, are applied between memories **M** or **W** and input pattern $\mathbf{x}^\omega$, where $\omega \in \{1, 2, ..., p\}$, to obtain the column vector **y** of dimension *m* as equations (5) and (6) shows:

$$\mathbf{y} = M \oplus \mathbf{x}^\omega \quad (5)$$

$$\mathbf{y} = W \oplus \mathbf{x}^\omega \quad (6)$$

3 mtDNA recognition

In this section, we will describe the dataset that we used in this work, after, we will describe the algorithm to store and recognize the mtDNA.

3.1. mtDNA dataset

The mtDNA dataset [14] was obtained from National Center for Biotechnology Information (NCBI) that is part of the National Library of Medicine of United States. Besides, the NCBI offers some bioinformatics tools for the analisys of DNA sequences, RNA and proteins, and they are free and online.

The dataset has 276 patterns; each pattern has 60 characters such as adenine (a), cytosine (c), guanine (g) y thymine (t), which represent the four nitrogenous bases that are contained in DNA.

3.2 Algorithm

The data was stored in an Excel file. The data (characters) are converted to integers by the use of the ASCII code. Due to lower case characters start at the decimal number 97, we subtracted 97 to all the dataset, therefore, the numbers representing the characters are: 0-a, 2-c, 6-g and 19-t. For example, one of the sequences is,

gatcacaggt ctatcaccct attaaccact cacgggagct ctccatgcat ttggtatttt

then, the corresponding decimal numbers represented as a vector,

[6,0,19,2,0,2,0,6,6,19,2,19,0,19,2,0,2,2,2,19,0,19,19,0,0,2,2,0,2,19,2,0,2,6,6,6,0,6,2,19,2,19,2,2,0,19,6,2,0,19,19,19,6,6,19,0,19,19,19,19]

This vector corresponds to an input pattern x. Then, we have 276 input patterns with dimension of 97 and, we have 276 output patterns, y, with dimension of 276. When we use the max memory, the vectors y are built as follows,

$\mathbf{y}^1$ = [500 0 0 0 … $0_{(100)}$ $0_{(101)}$ … $0_{(275)}$ $0_{(276)}$]
$\mathbf{y}^2$ = [0 500 0 0 … $0_{(100)}$ $0_{(101)}$ … $0_{(275)}$ $0_{(276)}$]
$\mathbf{y}^3$ = [0 0 500 0 … $0_{(100)}$ $0_{(101)}$ … $0_{(275)}$ $0_{(276)}$]
…
$\mathbf{y}^{100}$ = [0 0 0 0 … $500_{(100)}$ $0_{(101)}$ … $0_{(275)}$ $0_{(276)}$]
$\mathbf{y}^{101}$ = [0 0 0 0 … $0_{(100)}$ $500_{(101)}$ … $0_{(275)}$ $0_{(276)}$]
…
$\mathbf{y}^{275}$ = [0 0 0 0 … $0_{(100)}$ $0_{(101)}$ … $500_{(275)}$ $0_{(276)}$]
$\mathbf{y}^{276}$ = [0 0 0 0 … $0_{(100)}$ $0_{(101)}$ … $0_{(275)}$ $500_{(276)}$]

The vectors to build a min memory are similar but we changed the value of 500 for -500. We used this value because we have observed that the value in the diagonal must be greater than the maximum value of the elements of the input vectors [15]. We want to highlight that we see the recognition problem as a classification task. Therefore, every pattern of mtDNA is seen as a class, then, we have 276 different classes. That is the reason for

which the output vectors have a number 500 or -500 in the index that indicates the number of the class.

Now, we applied the training phase to build both memories: *max* and *min* by the use of equations (3) and (4). Then, we present an input pattern to the memories to recall its corresponding output vector. The following example illustrates these operations. We used vectors with similar values but with lower dimension, and we just used *max* memory

$$\mathbf{x}^1 = [6\ 0\ 19\ 2\ 0] \rightarrow \mathbf{y}^1 = [500\ 0\ 0\ 0]$$
$$\mathbf{x}^2 = [6\ 6\ 19\ 2\ 19] \rightarrow \mathbf{y}^2 = [0\ 500\ 0\ 0]$$
$$\mathbf{x}^3 = [6\ 6\ 6\ 0\ 6] \rightarrow \mathbf{y}^3 = [0\ 0\ 500\ 0]$$
$$\mathbf{x}^4 = [0\ 0\ 2\ 2\ 0] \rightarrow \mathbf{y}^4 = [0\ 0\ 0\ 500]$$

Now, we calculate the first association as follows.

$$\mathbf{y}^1 \otimes (-\mathbf{x}^1)^t = \begin{bmatrix} 500 \\ 0 \\ 0 \\ 0 \end{bmatrix} \otimes -\begin{bmatrix} 6 & 0 & 19 & 2 & 0 \end{bmatrix}$$

$$\mathbf{y}^1 \otimes (-\mathbf{x}^1)^t = \begin{bmatrix} 500-6 & 500-0 & 500-19 & 500-2 & 500-0 \\ 0-6 & 0-0 & 0-19 & 0-2 & 0-0 \\ 0-6 & 0-0 & 0-19 & 0-2 & 0-0 \\ 0-6 & 0-0 & 0-19 & 0-2 & 0-0 \end{bmatrix}$$

$$\mathbf{y}^1 \otimes (-\mathbf{x}^1)^t = \begin{bmatrix} 594 & 500 & 481 & 498 & 500 \\ -6 & 0 & -19 & -2 & 0 \\ -6 & 0 & -19 & -2 & 0 \\ -6 & 0 & -19 & -2 & 0 \end{bmatrix}$$

We calculate the remain asociations and the results are,

$$\mathbf{y}^2 \otimes (-\mathbf{x}^2)^t = \begin{bmatrix} -6 & -6 & -19 & -2 & -19 \\ 494 & 494 & 481 & 498 & 481 \\ -6 & -6 & -19 & -2 & -19 \\ -6 & -6 & -19 & -2 & -19 \end{bmatrix}$$

$$\mathbf{y}^3 \otimes (-\mathbf{x}^3)^t = \begin{bmatrix} -6 & -6 & -6 & 0 & -6 \\ -6 & -6 & -6 & 0 & -6 \\ 494 & 494 & 494 & 500 & 494 \\ -6 & -6 & -6 & 0 & -6 \end{bmatrix}$$

$$\mathbf{y}^4 \otimes (-\mathbf{x}^4)^t = \begin{bmatrix} 0 & 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & -2 & 0 \\ 0 & 0 & -2 & -2 & 0 \\ 500 & 500 & 498 & 498 & 500 \end{bmatrix}$$

Now, we obtain the greatest number of each element of each association and we have as a result the max memory.

$$\mathbf{M} = \begin{bmatrix} 494 & 500 & 481 & 498 & 500 \\ 494 & 494 & 481 & 498 & 481 \\ 494 & 494 & 494 & 500 & 494 \\ 500 & 500 & 498 & 498 & 500 \end{bmatrix}$$

In the recalling phase (equations (5) and (6)), we present an input pattern to **M** to recall its corresponding output pattern.

$$\mathbf{M} \oplus \mathbf{x}^1 = \begin{bmatrix} 494 & 500 & 481 & 498 & 500 \\ 494 & 494 & 481 & 498 & 481 \\ 494 & 494 & 494 & 500 & 494 \\ 500 & 500 & 498 & 498 & 500 \end{bmatrix} \oplus \begin{bmatrix} 6 \\ 0 \\ 19 \\ 2 \\ 0 \end{bmatrix}$$

$$\mathbf{M} \oplus \mathbf{x}^1 = \begin{bmatrix} 494+6 \wedge 500+0 \ \wedge \ 481+19 \ \wedge 498+2 \wedge \ 500+0 \\ 494+6 \wedge 494+0 \ \wedge \ 481+19 \ \wedge 498+2 \wedge \ 481+0 \\ 494+6 \wedge 494+0 \ \wedge \ 494+19 \ \wedge 500+2 \wedge \ 494+0 \\ 500+6 \wedge 500+0 \ \wedge \ 498+19 \ \wedge 498+2 \wedge \ 500+0 \end{bmatrix}$$

$$\mathbf{M} \oplus \mathbf{x}^1 = \begin{bmatrix} 500 \wedge 500 \wedge \ 500 \ \wedge 500 \wedge \ 500 \\ 500 \wedge 494 \wedge \ 500 \ \wedge 500 \wedge \ 481 \\ 500 \wedge 494 \ \wedge \ 513 \ \wedge 502 \wedge \ 494 \\ 506 \wedge 500 \ \wedge \ 517 \ \wedge 500 \wedge \ 500 \end{bmatrix} = \begin{bmatrix} 500 \\ 481 \\ 494 \\ 500 \end{bmatrix}$$

All the recalled vectors are,

$$\mathbf{yrec}^1 = [500\ 481\ 494\ 500]$$
$$\mathbf{yrec}^2 = [500\ 500\ 500\ 500]$$
$$\mathbf{yrec}^3 = [487\ 487\ 500\ 498]$$
$$\mathbf{yrec}^4 = [483\ 481\ 494\ 500]$$

From these results, we can observe that only the two last patterns are recalled or well classified. We conclude that because the number 500 appeared in the correct index of the vector.

The recalled patterns when we use the *min* memory **W** are,

$$\mathbf{yprec}^1 = [-500\ -500\ -487\ -483]$$
$$\mathbf{yprec}^2 = [-481\ -500\ -487\ -481]$$
$$\mathbf{yprec}^3 = [-494\ -500\ -500\ -494]$$
$$\mathbf{yprec}^4 = [-500\ -500\ -498\ -500]$$

Then, we transform the recalled vectors by placing a number 1 where the value of the element is 500 or -500, and we write a 0 in the rest of the elements. The new vectors are the following.

$$\mathbf{yt}^1 = [1\ 0\ 0\ 1],\ \mathbf{ypt}^1 = [1\ 1\ 0\ 0]$$
$$\mathbf{yt}^2 = [1\ 1\ 1\ 1],\ \mathbf{ypt}^2 = [0\ 1\ 0\ 0]$$
$$\mathbf{yt}^3 = [0\ 0\ 1\ 0],\ \mathbf{ypt}^3 = [0\ 1\ 1\ 0]$$
$$\mathbf{yt}^4 = [0\ 0\ 0\ 1],\ \mathbf{ypt}^4 = [1\ 1\ 0\ 1]$$

The last step is to apply an and operation between **yt** and **ypt** vectors.

$$\mathbf{yt}^1 \mathit{and}\ \mathbf{ypt}^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \mathit{and} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The last result indicates that the first pattern was well classified, *i.e*, the pattern corresponds to the class 1, because the number 1 is located at the first position. Now, we do the same with the remain vectors.

$$\mathbf{yt}^2 and\ \mathbf{ypt}^2 = \begin{bmatrix}1\\1\\1\\1\end{bmatrix} and \begin{bmatrix}0\\1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

$$\mathbf{yt}^3 and\ \mathbf{ypt}^{31} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} and \begin{bmatrix}0\\1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

$$\mathbf{yt}^4 and\ \mathbf{ypt}^4 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix} and \begin{bmatrix}1\\1\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

We can observe that in all the cases the patterns were classified in the correct class.

4 Results

We developed a software in Matlab 2013, implemented in a laptop Dell® with an Intel® Core i7 processor.

We built two morphological associative memories: max and min with the 276 patterns of the dataset.

In the recalling phase we recovered all the patterns. This means that our memory does not have forgetting factor: everything is learned everything is recalled. Then we can say that our memory has a correct recall.

We simulated the problem when the sample of DNA is corrupted with noise (or incomplete).

We applied additive and subtractive noise to the patterns. In Table 1, we show the results when we apply a percentage of additive noise.

Table 1. Recalling of our memory with additive noise

% additive noise	Correct recall?
10	yes
20	yes
30	yes
40	yes
50	yes
51	yes
52	yes
53	yes
54	yes
55	yes
56	no
57	no
58	no
59	no
60	no

From Table 1, it can be observed that we have the correct recall when the percentage of additive noise is less or equal than 55.

We observed the same behaviour when we applied subtractive noise to the patterns: from the 56% of subtractive noise, there is not correct recall.

5 Conclusions

We applied an artificial intelligent tool to recognize patterns of mtDNA. We chose this kind of DNA because it is resistant to degradation.

The Morphological Associative Memories were a suitable algorithm for this application. This model has a low complexity due to their main operations: addition, subtraction and max and min operators.

We deal the pattern recognition problem as a classification task. This is the reason that we built the output vectors with value of 500 or -500 in the index that indicates the number of its corresponding class.

Our proposal recalled all the patterns that were trained, therefore, it has a correct recall and it showed a forgetting factor of zero.

If a sample of DNA is corrupted, there is the possibility that the sequence cannot be complete or some elements of the sequence can be change, even when this type of DNA is resistant. This problem were simulated by adding noise to the patterns. We applied additive and subtracting noise. We found that when the patterns contained more than the 55% of noise, the memory was not able to recognize the corresponding class.

Acknowledgments

The authors would like to thank the Instituto Politécnico Nacional (COFAA and SIP), and SNI for their economical support to develop this work.

References

1. Genetics Home Reference, A service of the U.S. National Library of Medicine, http://ghr.nlm.nih.gov/mitochondrial-dna
2. M. Craven and J. Shavlik. Proc. of the Tenth Int. Conf. on Machine Learning (1993).
3. S. Vijayvargiya and P. Shukla. Int. J. of Adv. In Tech. **2**(1), (2011).
4. Derong Liu, Xiaoxu Xiong and Bhaskar DasGupta. Networking, Sensing and Control. IEEE (2005).
5. Alexandra M. Carvalho, Ana T. Freitas, Arlindo L. Oliveira, and Marie-France Sago. IEEE/ACM Trans. Comp. Biol. Bioinfo. **3**, 2 (2006).
6. M. Abbass and H. Bahig. Math. Comput. Sci. **7** (2013)
7. J.D. Wells and Felix Sperling. Forensic Sci. Int. **120** (2001).
8. R. Ranawana and V. Palade. Neural Comput. & Applic. **14** (2005).
9. A. Pushpalatha and B. Mukunthan. Int. J. Comp. App. **13**(3), (2011).

10. K. Dheenathayalan and J. RamSingh. Intern. Conf. Intellig. Comput. App. (2014).
11. T. M. Inbamalar and R. Sivakumar. The Sci. World J. Article ID 786497 (2015).
12. C. Yáñez-Márquez, Associative Memories Based on Order Relations and Binary Operators (In Spanish). PhD Thesis. Centro de Investigación en Computación, Mexico (2002).
13. G. X. Ritter, P. Sussner and J. L. Diaz de León. IEEE Trans. on Neur. Net. **9** (1998).
14. http://www.ncbi.nlm.nih.gov/nuccore/AY195748
15. R. Navarro, E. Acevedo, M. Acevedo and F. Martínez. MCPR 2012, LNCS 7329, (2012).