Where Do Patterns To Be Recognized Come From? (CDT-22)

Preprint · February 2020

DOI: 10.13140/RG.2.2.20866.63681

1 author:

Luciano da F. Costa
University of São Paulo
734 PUBLICATIONS **13,156** CITATIONS

SEE PROFILE

Where Do Patterns To Be Recognized Come From? (CDT-22)

Luciano da Fontoura Costa
luciano@ifsc.usp.br

São Carlos Institute of Physics – DFCM/USP

v.2 3rd March 2020
(v.1 29th Feb. 2020)

Abstract

As a consequence of its close relationship with human cognition and intelligence, pattern recognition has motivated great interest in science and technology. Though emphasis is typically placed on feature selection and classification methods, the patterns to be categorized need to be generated first. In this work we discuss how better understanding about how the patterns to be recognized were originally generated can provide valuable insights about both feature selection and pattern classification. Indeed, the relationship between pattern generation and pattern recognition indicates that the latter is intrinsically related to scientific modeling, i.e. understanding how data is produced.

'...facendo a similitudine dello specchio, il quale si tramuta in tanti colori, quanti sono quelli delle cose che gli si pongono dinanzi...'

Leonardo da Vinci.

1 Introduction

Much of human cognition and intelligence is closely related to the ubiquitous task of pattern recognition (e.g. [1, 2, 3]). Simplistically speaking, this task consists in, given some entities represented in terms of respective measurements or features, assigning new or previously defined categories. In the case of new categories, we have *unsupervised classification* or *clustering*, and in the case of predefined categories, *supervised classification*.

Typically, supervised classification relies on predefined categories, which were originally determined through some unsupervised approach (e.g. [4]). Because both these types of classifications are often a challenge, many efforts have been made toward obtaining improved respective methods. However, before a set of specific patterns can be recognized, they need to be *generated* first, as illustrated in Figure 1, therefore defining three subsequent related tasks: (i) *pattern generation*; (ii) *feature selection/extraction*; and (iii) *pattern classification*.

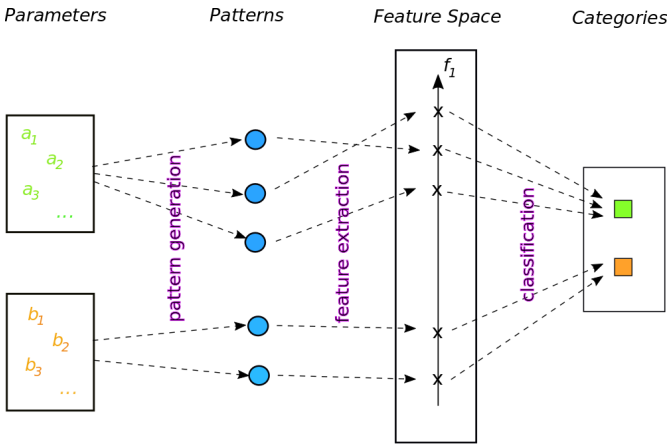


Figure 1: Patterns to be recognized need to be generated first. In this diagram we show the traditional two main stages in pattern recognition, namely feature extraction and classification, as well as the prior stage respective to the generation of the patterns.

In the present work, we address the interesting question of whether and how knowledge about how patterns to be recognized were generated can help important tasks such as feature selection and the identification of effective classification methods.

Though patterns can be generated in a virtually infinite number of ways, in this work we consider patterns produced by probabilistic automata (e.g. [5, 6]), namely graphs whose links correspond to transition probabilities, while a symbol is assigned to each node. As we will see, patterns can be obtained by having an hypothetical agent to perform a random walk (e.g. [7]) along this type of graph. We will show that this simple approach can yield several types of patterns in one or more dimensions. Then, we will approach the problem of recognizing these patterns that we are able to generate. In particular, how can the knowledge about the generation of the patterns help in choosing effective features and classification methods?

It should be kept in mind that we by no means believe that all possible pattern can be understood as coming from a probabilistic automata. Many other types of pattern generation could be considered, including fractals (e.g. [8]), Lissajous curves, cellular automata (e.g. [9]), reaction-diffusion systems (e.g. [10]), to name but a few possibilities. Real-world patterns are generated by even more diverse, complex and sophisticated systems, each with their specific properties. These facts, even if taken isolatedly, suggest that pattern generation is a issue well worth to be studied when approaching pattern recognition.

2 Deterministic Automata

Figure 2 illustrate a simple deterministic automaton, containing 4 nodes with associated symbols (blue for 0, yellow for 1) plus a termination node (orange). An *agent* starts at the leftmost node and proceeds along the arrows, until reaching the orange node, where the agent stops. As each node is visited, the corresponding symbol is output, so we have the pattern '$1, 0, 1, 1$' as result. The 1 above each link means that the respective link is taken with unit probability (.e. certainty), meaning that the same pattern is obtained always.

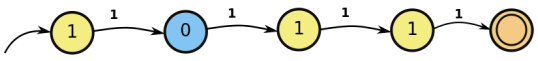


Figure 2: A determininstic automaton capable of generating the pattern '$1, 0, 1, 1$' as a hypothetical agent moves along the links, after starting at the leftmost node.

The obtained pattern can be represented graphically in several manners, some of which are illustrated in Figure 3.

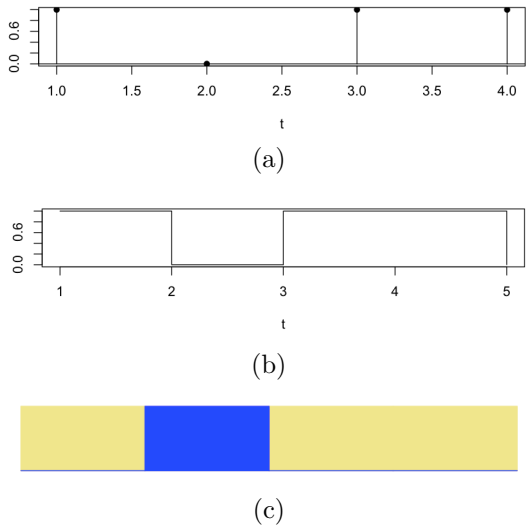


Figure 3: Three possible ways of representing the pattern '$1, 0, 1, 1$' graphically: (a) as a stemplot; (b) as a square wave; and (c) as a continuous bar plot.

Figure 4 illustrates two other deterministic automata that, though simple, are capable of: (a) generating an infinite sequence of 1s; and (b) producing a perfectly square wave '$0, 1, 0, 1, \ldots$'.

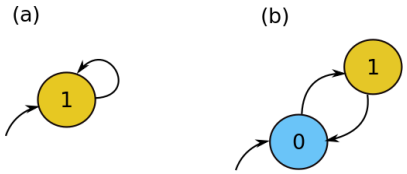


Figure 4: Two additional examples of deterministic automata: (a) an automaton capable of generating an infinite sequence of 1s; and (b) an automaton that generates a perfectly square wave '$0, 1, 0, 1, \ldots$'.

3 Probabilistic Automata

By allowing the transition probabilities to take values smaller than 1, it is possible to obtain probabilistic automata, allowing different patterns to be obtained at each respective execution, as the hypothetical agent performs a *random walk*. Figure 5 depicts a possible probabilistic automaton.

The agent starts at node 0 and performs a random walk so that, at each node, the outgoing links are taken with the respective probabilities. As the agent moves, the automaton outputs an infinite string of 0s and 1s, with pattern 0 having 9 times more chance of occurring than pattern 1. Examples of patterns that can be produced

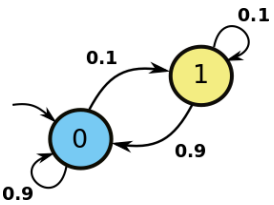


Figure 5: An example of probabilistic automaton, characterized by including transition probabilities smaller or equal to 1. This automaton is capable of producing infinite patterns consisting of successive groups of of 0s and 1s, starting with 0, and with pattern 0 having 9 times more chance of occurring than pattern 1.

by this automaton include but are by no means limited to '0, 0, 0, 1, 1, 0, 0, 1, 0, 1, . . .'; '0, 1, 1, 0, 0, 0, 0, 0, 0, 1, . . .'; and '0, 0, 1, 0, 1, 1, 0, 0, 0, 0, . . .'.

Other examples of probabilistic automata are provided in Figure 6, including the previous automaton in (a).

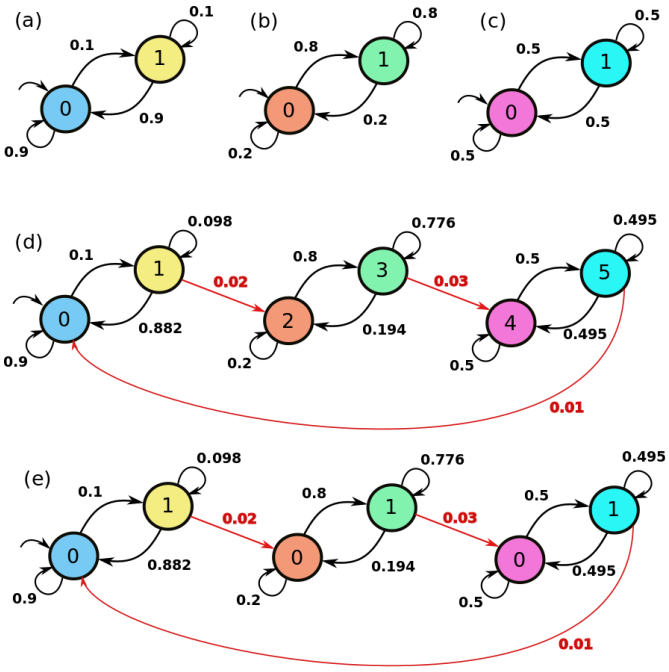


Figure 6: Diverse examples of probabilistic automata, including the automaton from Figure 5 in (a).

Figure 11 shows 4 examples truncated to the 200 first symbols for each of the respective probabilistic automata in Figures 6(a), (b), and (c).

The automaton in Figure 5, as well as those in the other examples, produces infinite patterns of 0s and 1s. It is possible to obtain finite patterns by either stopping that automata after a given number of agent steps, or by using a termination node, as in the automaton shown in Figure 8, which generates patterns with the same probabilistic structure as the automaton in Figure 5, but with 4 symbols only.

Probabilistic automata provide a means for generating

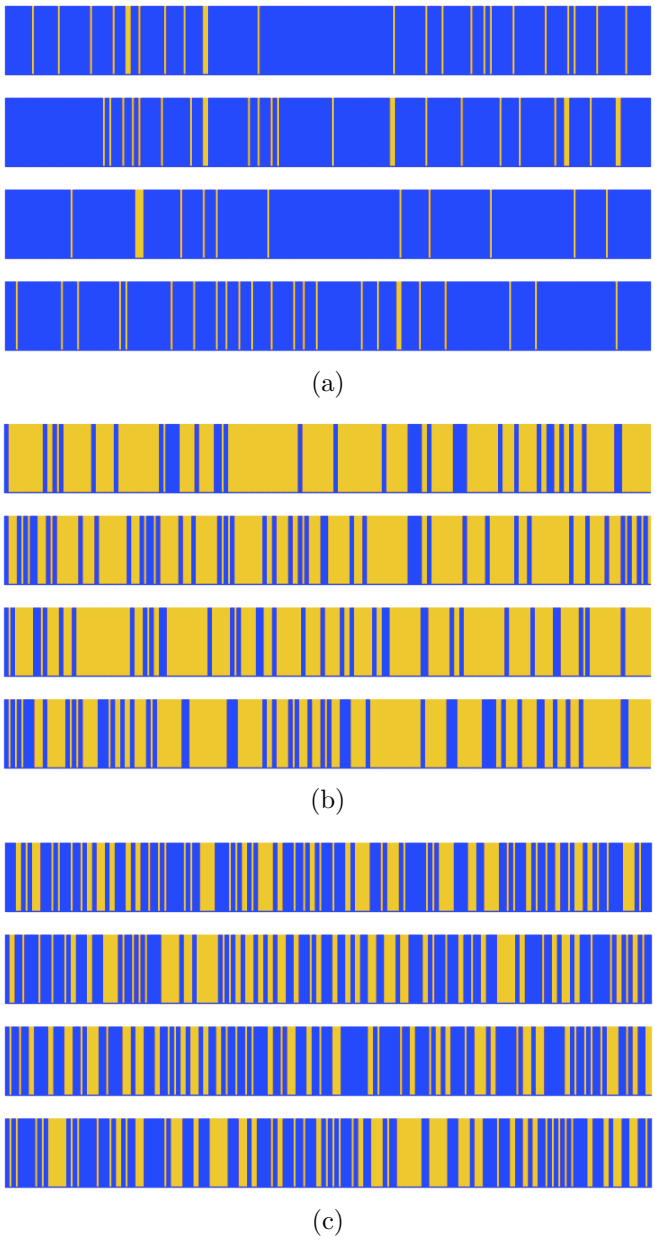


Figure 7: Sets of 4 patterns, truncated to the first 200 symbols, produced by the automata in Figures 6(a), (b), and (c). The symbols 0 and 1 are represented in blue and yellow, respectively.

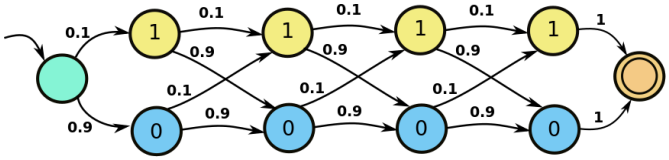


Figure 8: This automaton generates patterns with the same structure than the patterns generated by the automaton in Figure 5, but with fixed length of 4 symbols.

a variety of patterns that can be used for studying respective pattern recognition, as will be further developed in this work. However, before proceeding, it is interest-

ing to consider the related problem of implementing this type of automata computationally, as well as considering probabilistic pattern generation in a more mathematical manner. These two issues are addressed in the following sections, respectively.

4 Computational Implementation of Probabilistic Automata

Given a probabilistic automaton described by its respective transition probability matrix A, we can simulate the operation of this automaton through a random walk in which the nodes are visited successively, in random fashion following the transition probabilities, along the automaton structure. Computationally speaking, we need a means to choose one of the outgoing links whenever the agent is in a given node. This situation is illustrated graphically in Figure 9, in which the agent, currently at the node i shown in magenta, needs to choose among one of the four possible outgoing links, each of which having its respective transition probability, here represented as $p_{1,i}$, $p_{2,i}$, $p_{3,i}$, and $p_{4,i}$.

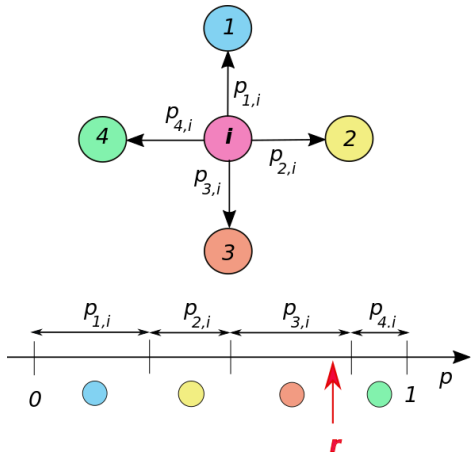


Figure 9: A moving agent, currently at node i (magenta) needs to choose one of the outgoing links to go through into one of the other 4 nodes. We can use a Monte Carlo-based approach in which the outgoing transition probabilities are stacked along a probability axis, p, adding necessarily to 1. Then, a random number r is drawn uniformly within the interval $[0, 1)$, and the link associated to the interval where r falls is chosen as the outgoing passage.

The choice of the outgoing link to be taken can be performed computationally by using a Monte Carlo-based approach. First, all the outgoing probabilities are stacked sequentially, side-by-side, along a probability axis p, which necessarily adds to 1 as a consequence of the matrix A being stochastic. Then, a random number r is drawn uniformly within the interval $[1, 0)$. The link corresponding to where r falls is taken as the outgoing passage.

The above methodology can be implemented computationally by using the following pseudo-code, where the vector *prbs* contains the probabilities $p_{1,i}$, $p_{2,i}$, $\ldots p_{M,i}$, where M is the total number of outgoing links, as components:

Algorithm 1 *NextStep*(*prbs*)

1. $i = 1$
2. $p = prbs[1]$
3. $r = rand(1)$; *draws r uniformly within* $[0, 1)$
4. While $r > p$ do
   (a) $i = i + 1$
   (b) $p = p + prbs[i]$
5. Output i

Observe that in this specific implementation we assumed that the possible destination nodes were numbered sequentially as $i = 1, 2, \ldots N$. The code will need to be slightly modified in case the nodes have different identifiers.

5 A Little Bit of Mathematics

The probabilities of the agent being at any of the nodes of a probabilistic automata can be estimated by using mathematical principles. Let's start by representing the automaton in Figure 5 in terms of its respective probability transition matrix:

$$A = \begin{bmatrix} p_{0,0} & p_{0,1} \\ p_{1,0} & p_{1,1} \end{bmatrix}$$

where $p_{i,j}$ represents the probability of the agent moving from node j to node i. In the specific configuration in Figure 5, we have the following transition matrix A:

$$A = \begin{bmatrix} 0.9 & 0.9 \\ \underline{0.1} & \underline{0.1} \\ 1 & 1 \end{bmatrix}$$

Observe that the sums of every column is 1, which means that this matrix is a *stochastic matrix*, which is a requisite for a well-specified probabilistic automaton (meaning that the outgoing probabilities are properly normalized so that they add to 1).

Let's say that the probabilities of being at each node are initially given by the following state probability vector $\vec{P}^{[t]}$

$$\vec{P}^{[t]} = \begin{bmatrix} p_0 \\ p_1 \end{bmatrix}$$

where p_0 and p_1 are the initial probabilities of the agent being at node 0 and 1, respectively.

By using basic probability principles, it follows that the probabilities of being at the each of the two nodes at a subsequent time instant $t + \Delta t$ can be obtained as:

$$\vec{P}^{[t+\Delta t]} = \begin{bmatrix} 0.9 & 0.9 \\ 0.1 & 0.1 \end{bmatrix} \vec{P}^{[t]}$$

Let's consider a simple example, with initial state probabilities $p_0 = 1$ and $p_1 = 0$. After one time step Δt, we have the new state probability given as

$$\begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.9 & 0.9 \\ 0.1 & 0.1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and, at the next instant we have:

$$\begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix} = \begin{bmatrix} 0.9 & 0.9 \\ 0.1 & 0.1 \end{bmatrix} \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}$$

which will remain unchanged henceforth, therefore corresponding to the *equilibrium* state probability for this specific automaton. Observe that such equilibrium states correspond to the eigenvector $\vec{v}$ of matrix A respective to eigenvalue $\lambda = 1$. In other words,

$$A\vec{v} = \lambda\vec{v}. \tag{1}$$

This provides a useful method to estimate the equilibrium occupancy probability of each of the states (corresponding to each of the nodes) in a probabilistic automaton.

As an exercise, let's confirm that 1 is indeed one of the eigenvalues of the considered automaton. We start by writing the respective *secular equation* of A as

$$det\,(A - \lambda I) = \begin{bmatrix} 0.9 - \lambda & 0.9 \\ 0.1 & 0.1 - \lambda \end{bmatrix} = 0$$

by developing the determinant, we get

$$det\,(A - \lambda I) = \frac{1}{10}\left[(9 - 10\lambda)(1 - 10\lambda) - (9)(1)\right] =$$
$$= 9 - 90\lambda - 10\lambda + 100\lambda^2 - 9 = 100(\lambda)(\lambda - 1) = 0 \tag{2}$$

which yields two eigenvalues: $\lambda_1 = 0$ and, as it could be expected, $\lambda_2 = 1$.

6 Sequential Patterns

It is possible to have much more general probabilistic automata than those in Figure 6(a-c). One of the possibilities for obtaining these automata consists in merging smaller sub-automata. For instance, Figure 6(d) shows an automaton obtained by connecting the automaton in Figure 6(a) to that in Figure 6(b), then connecting the latter to the automaton in Figure 6(c), and then connecting the latter to the initial automata, as indicated by the links shown in red. Observe that the nodes of the obtained, combined automaton have been assigned sequential symbols, from 0 to 5. The obtained patterns can be understood as a sequence of sub-patterns produced in turn by each of the involved sub-automata.

When adding a new link with transition probability p to an existing node, the other outgoing probabilities need to be renormalized proportionally in order that all outgoing links add to 1. For instance, consider the addition of the new link with probability 0.02 to the leftmost automaton in Figure 6(a). There were, originally, two outgoing links, with respective probabilities 0.1 and 0.9. First, we subtract 0.02 from 1, yielding 0.98. This remaining probability is then split proportionally to the two original probabilities, yielding the renormalized values of $p_{1,1} = (0.1)(0.98) = 0.098$ and $p_{0,1} = (0.9)(0.98) = 0.882$. Observe that we have $p_{1,1} + p_{1,0} + p_{1,2} = 0.098 + 0.882 + 0.02 = 1$, as could be expected. An analogue procedure applies when removing a link.

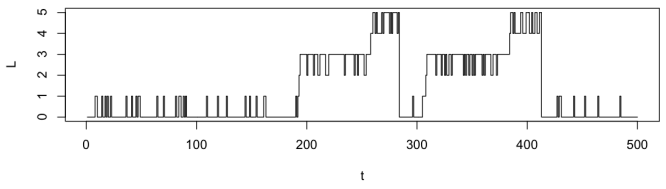
The so-obtained automaton will produce patterns consisting of sequentially appending patterns generated by the automaton in Figure 6(a), followed by patterns produced by the automaton in Figure 6(b), followed by patterns produced by the automaton in Figure 6(c), and then repeating this sequence. Figure 10(a) illustrates one of the patterns that can be produced by the combined automaton.

7 Multidimensional Patterns

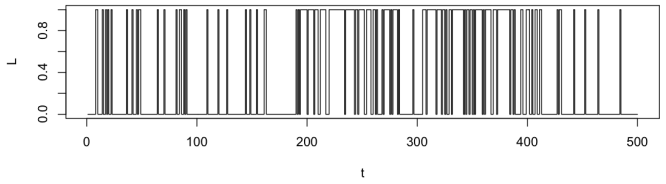
One-dimensional patterns obtained from probabilistic automaton such as those seen in this work can be used as the basis for generating higher dimensional patterns, such as images. This can be done in a virtually unlimited number of ways. For instance, we can start with a blank $N \times N$ image and add vertical bars at the positions marked with the symbol 1 along a one-dimensional pattern L having length N. A similar scheme can be used to add horizontal bars, considering the same pattern L, or a pattern from another realization of the automaton, or a patterns coming from a distinct automaton.

Figure 11 illustrates this types of patterns, considering a same specific color scheme (diverse color patterns can be used).

Another related approach to produce two-dimensional patterns would be to add a given shape at the crossing points defined by the previous patterns. Figure 12 illus-



(a)



(b)

Figure 10: An example of sequential pattern generated by the automaton in Figure 6 (d) is shown in (a). Observe that the parts of the pattern generated by each of the three sub-automata can be readily identified by observing the respective symbol values. These parts are much harder to be recognized in the same pattern as produced by the automaton in Figure 6(e), shown in (b), as this pattern is composed only by 0s and 1s.

trates one such pattern obtained by placing a circle with radius 15 at each of the crossing points defined in the pattern in Figure 11(a). Interestingly, the essential information required for constructing this image includes only the original one-dimensional pattern and the information about the repetition of the circle with radius 15 pixels, so that all the remainder information could be understood as being *redundant*.

8 Recognizing Generated Patterns

So far, we have seen how several types of one and multidimensional patterns can be generated by using probabilistic automata. Now, we proceed to the issue of recognizing these patterns. Given that we know precisely how they were generated, can this information be of any help while selecting features or choosing classification methods?

Let's first consider the patterns generated by the three automata in Figure 6(a-c), of which same examples are illustrated respectively in Figure 11. We know that all these three types of patterns have the same qualitative structure of subsequent groups of 0s and 1s, one of the differences between the three types corresponding to the probabilities in which the symbols 0 and 1 appear in each case. Thus, a possible feature to be selected for subsequent pattern classification would be the relative number of 1s, i.e. the total number of 1s divided by the length N of the pattern, yielding $f = (\text{number of 1s})/N$.

Figure 13 shows the density of f obtained for pattern lengths varying as $500, 750, \ldots, 2000$ for 1000 realizations of each of the three automata in Figure 6(a-c). The densities correspond to normal approximations considering the respectively estimated averages and standard deviations obtained in each case.

As it could be expected, the relative frequency of ones obtained for several executions of the same configuration of each automata yielded a dispersion, as indicated by the normal widths, that tends to decrease with the length N of the patterns. Importantly, virtually no overlap can be observed between the densities corresponding to the three types of patterns considered in this experiment, which would favor respective recognition without any significative misclassifications.

It is interesting to consider also the case in which the three generating automata had more similar parameters. Figure 14 depicts the densities obtained for three automata with transition probabilities: $0.55 : 0.45$ (orange); $0.4 : 0.6$ (green); and $0.5 : 0.5$ (blue).

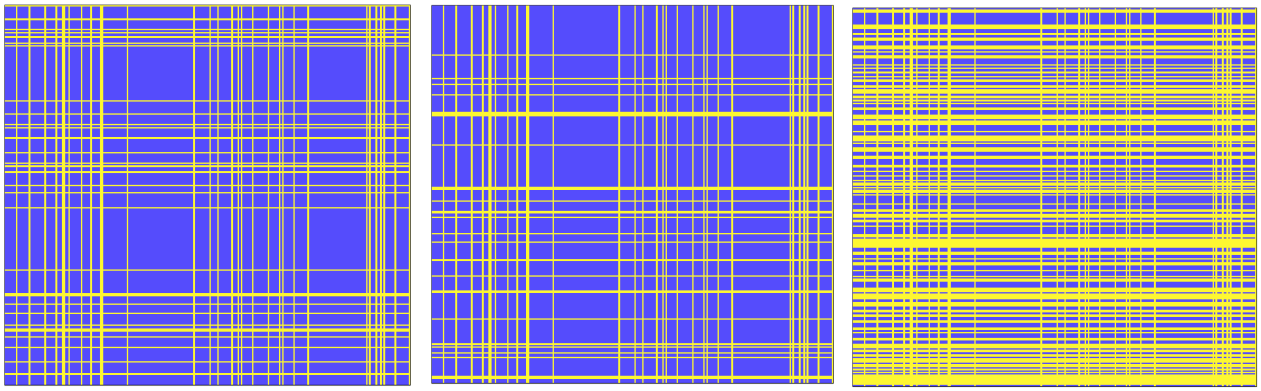
Substantial overlap can now be observed, especially between the orange and blue densities. This case indicates that, even when we know how patterns are generated, we can by no means guarantee fully precise classification, as a consequence of intrinsic variability of the properties of patterns generated with similar parameters.

The situation is even more intricate. For instance, let's now consider two automata, such as that in Figure 6 and another that produces patterns with one group of 0s followed by another group of 1s, with the same probabilities as the former automata. It immediately follows that the relative frequency of ones will no longer be an effective feature to be adopted for recognizing these patterns.

This interesting experiment has some important implications. First, it makes us realize that the relative frequency of ones does not provide a complete (bijective) representation of the patterns, implying that more than one pattern can be mapped into the same feature. Indeed, it was only possible to avoid this problem in the previous experiments because we had a *restricted* domain of patterns that ensured bijective mapping from them into the adopted relative frequency feature.

We can also conclude that patterns with the same relative frequency of symbols, but with distinct structure, will require additional features to be taken into account, such as statistics of the distance between the same type of symbol along the patterns.

As an exercise, the reader is encouraged to think about possible effective features to be adopted for the recognition of the sequential and multidimensional patterns presented in Sections 6 and 7.



(a) (b) (c)

Figure 11: Three examples of two-dimensional patterns (images) obtained by: (a) bars placed according to the same one-dimensional pattern generated by the automaton in Fig. 5; (b) bars placed according to two distinct one-dimensional pattern generated by the automaton in Fig. 5; and (c) bars placed according to one one-dimensional pattern generated by the automaton in Fig. 5 (vertical lines) and one one-dimensional pattern generated by the automaton in Figure 6(c) (horizontal lines).

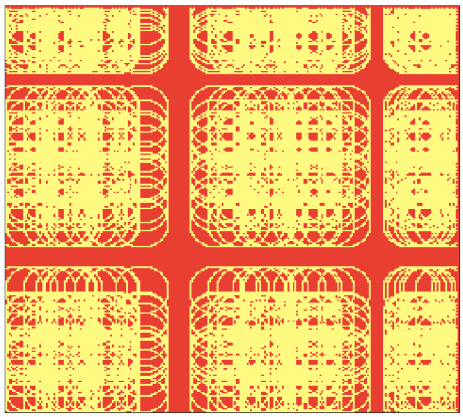


Figure 12: A pattern obtained by placing a circle with radius 15 at each of the crossing points defined in the pattern in Figure 11(a).

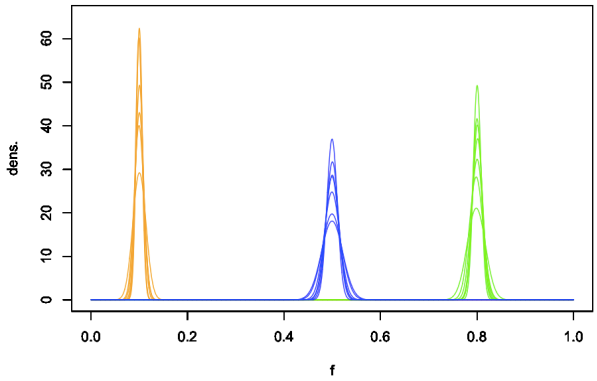


Figure 13: Densities of the relative number of 1s (f) obtained for 200 executions of the automata in Figure 6(a), shown in orange; (b), shown in green; and (c), shown in blue. Observe that the dispersion of f tends to decrease with the length N of the patterns. In this case, no overlap can be observed between the densities, which would favor recognition without misclassifications.

9 Concluding Remarks

As a consequence of its close relationship with human cognition and intelligence, pattern recognition will continue to motivate much interest and efforts in science and technology. Though the entities to be recognized are often understood to be given, it is important to take into account the fact that they needed to be somehow generated by some natural or human-mediated manner. The current work briefly addressed the key issue whether and how knowledge about the generation of the patterns can contribute to improved pattern recognition, by providing subsidies for better identification of suitable features and more effective classification methods.

We started by presenting how deterministic and stochastic patterns can be generated by using automata, more specifically random walks in graphs or networks. Several examples were provided and discussed, including considerations about the computational implementation of the probabilistic automata and some mathematical background related to the spectral analysis of the stochastic transition matrices. The potential of using probabilistic automata for pattern generation was illustrated with respect to separated, sequential and multidimensional patterns.

The problem of recognizing some of the described generated patterns was then addressed. In particular, we considered finite-length patterns obtained from three independent probabilistic automata yielding sequences of 0s and 1s, but with varying symbol probabilities. The knowledge about how the patterns were generated allowed the identification of a feature, namely the relative frequency of 1s, that provides effective subsidies for respective clas-

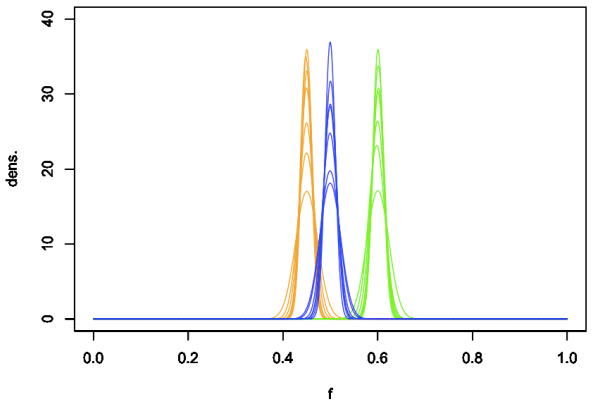


Figure 14: Densities of f obtained for three automata with transition probabilities: 0.55 : 0.45 (orange) ; 0.4 : 0.6 (green) and 0.5 : 0.5 (blue). Significant ovrlap is now observed between the blue/orange and blue/green densities, implying in possible misclassifications.

sification of patterns generated with relatively different parameter configurations. However, as a consequence of the dispersion of the values of this feature observed especially when the patterns are shorter, we also concluded that it will be impossible to ensure error-free classification when the patterns were generated with relatively similar parameter configurations. We also discussed how the consideration of restricted pattern domains favor correct classification, and why additional features may be needed when the generated patterns share some properties. All in all, we can conclude that:

> Even when we know *how patterns are generated*, it *does not* necessarily follow that we can recognize those patterns without possible *misclassifications*, or will necessarily be able to know the *best features* for their recognition, or even the *best classification* approach, though we will generally be in a *better position* for addressing those aspects.

It is hoped that the framework developed in this work has provided sound subsidies for discussing how pattern generation is intrinsically important for pattern recognition. Further considerations would be interesting regarding the relationships of the presented material and deep learning (e.g. [11]) and scientific modeling (e.g. [12]).

Acknowledgments.

Luciano da F. Costa thanks CNPq (grant no. 307085/2018-0) for sponsorship. This work has benefited from FAPESP grant 15/22308-2.

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2000.

[2] K. Koutrombas and S. Theodoridis. *Pattern Recognition*. Academic Press, 2008.

[3] L. da F. Costa. Pattern cognition, pattern recognition. Researchgate, Dec 2019. `https://www.researchgate.net/publication/338168835_Pattern_Cognition_Pattern_Recognition_CDT-19`. Online; accessed 29-Feb-2020.

[4] L. da F. Costa. On features and categories. Researchgate, Feb 2020. `https://www.researchgate.net/publication/339271541_On_Features_and_Categories_CDT-20`. Online; accessed 29-Feb-2020.

[5] J. R. Norris. *Markov Chains*. Cambridge University Press, 1998.

[6] P. Linz. *An Introduction to Formal Languages and Automata*. Jones & Bartlett Learning, 2016.

[7] J. Zinn-Justing. *From random walks to random matrices*. Oxford University Press, 2019.

[8] H.-O. Peitgen, H. Jürgens, and D. Saupe. *Chaos and Fractals: New Frontiers of Science*. Springer, 2004.

[9] S. Wolfram. *A new kind of Science*. Wolfram Media, 2002.

[10] A. Benatti and L. da F. Costa. Pattern formation: Modeling space-time dynamics. Researchgate, Feb 2020. `https://www.researchgate.net/publication/339415849_Pattern_Formation_Modeling_Space-Time_Dynamics_CDT-21`. Online; accessed 29-Feb-2020.

[11] H. F. de Arruda, A. Benatti, C. H. Comin, and L. da F. Costa. Learning deep learning. Researchgate, 2019. `https://www.researchgate.net/publication/335798012_Learning_Deep_Learning_CDT-15`. Online; accessed 22-Dec-2019.

[12] L. da F. Costa. Modeling: The human approach to science. Researchgate, 2019. `https://www.researchgate.net/publication/333389500_Modeling_The_Human_Approach_to_Science_CDT-8`. Online; accessed 03-June-2019.

Costa's Didactic Texts – CDTs

CDTs intend to be a halfway point between a formal scientific article and a dissemination text in the sense that they: (i) explain and illustrate concepts in a more informal, graphical and accessible way than the typical scientific article; and (ii) provide more in-depth mathematical developments than a more traditional dissemination work.

It is hoped that CDTs can also incorporate new insights and analogies concerning the reported concepts and methods. We hope these characteristics will contribute to making CDTs interesting both to beginners as well as to more senior researchers.

Each CDT focuses on a limited set of interrelated concepts. Though attempting to be relatively self-contained, CDTs also aim at being relatively short. Links to related material are provided in order to complement the covered subjects.

Observe that CDTs, which come with absolutely no warranty, are non distributable and for non-commercial use only.

The complete set of CDTs can be found at: `https://www.researchgate.net/project/Costas-Didactic-Texts-CDTs`.