CLASSIFICAÇÃO DE COMPLEXIDADE DE AUTÔMATOS CELULARES POR MEIO DE ANÁLISE DE VETORES DE TEXTURA

Arthur Moreira Passos e Eurico Luiz Prospero Ruivo (Orientador)

Apoio: PIBIC Mackpesquisa

RESUMO

Autômatos Celulares (ACs) são sistemas definidos a partir de regras simples baseadas em células, os estados dessas e a relação entre elas. Contudo, essas definições básicas podem gerar comportamentos extremamente complexos e até imprevisíveis antes de serem propriamente simulados. A partir disso, estudos tem tentado classificá-los a partir dessas propriedades de complexidade de comportamento. Uma das classificações que surgiu foi a de Wolfram (1984), que se baseou nos padrões na evolução temporal dos ACs, para definir quatro classes de complexidade: Classe I, Classe II, Classe III e Classe IV. Baseado nessa classificação, conjectura-se (WOLFRAM, 2002) que regras da Classe IV apresentam computabilidade universal.

É possível classificar os ACs manualmente em espaços de regras mais básicas, porém, ao tratar de espaços maiores e mais complexos, a classificação manual se torna inviável, surgindo assim a necessidade de desenvolver modelos de classificação automática.

Nesse projeto é implementado um método de classificação a partir da análise de características de um vetor de texturas extraídas da evolução temporal da execução das regras dos ACs unidimensionais binários com configurações iniciais aleatórias. Para automatizar e testar a análise das características, são utilizados os algoritmos k-NN (k-Nearest Neighbors) e o percéptrons de múltiplas camadas (PMCs).

Nos resultados obtidos o algoritmo desenvolvido apresentou um mínimo de 67% de precisão, porém, esse método ainda pode ser explorado mais profundamente.

Palavras-chave: Autômatos Celulares. Classificação de Complexidade. Vetor de Textura.

ABSTRACT

Cellular Automata (CA) are systems defined by simple rules based on cells, their states, and the relationship between cells. However, those definitions can generate highly complex behavior and even unpredictable before properly simulated. Therefore, studies have tried to classify these systems based on those complexity properties. One of the classifications that

emerged was the Wolfram (1984)'s. It was based on the patterns of the temporal evolution of the CAs, defining four complexity classes: Class I, Class II, Class III and Class IV. Based on this classes, Wolfram (2002) conjectures that rules classified as Class IV have the property of universal computability.

It is possible to classify the CA manually when dealing with the most basic rules, however, when dealing with wider and more complex spaces, the manual classification becomes impracticable, emerging the need to develop automized models of classification.

This project implements a method of classification unidimensional binary ACs with random initial configuration by analyzing characteristics of a texture vector extracted from the temporal evolution. To automate and test the analysis of the characteristics, the algorithms k-NN (k-Nearest Neighbors) and multilayer perceptron (MLP) are used.

In the results collected the algorithm presented a minimum of 67.5% of precision, although this method could be explored further.

Keywords: Cellular Automata. Complexity Classfication. Texture vector.

1 INTRODUÇÃO

Autômatos celulares (ACs) são sistemas dinâmicos totalmente discretos (no tempo, no espaço e nas variáveis de estado) de comportamento localmente definido. Por mais essas definições sejam em termos de regras locais simples, analisando o comportamento das células entre si é possível perceber uma evolução complexa, e em alguns casos, a complexidade de algumas regras pode apresentar computabili8dade universal, como a "Regra 110" (COOK, 2004) do espaço elementar.

Tal complexidade cresce de forma exponencial ao se tratar de espaços além do elementar, inviabilizando classificações manuais no cenário desses espaços.

Classificar as regras de ACs quanto ao comportamento e complexidade é um tema significativo de estudos na área (MARTINEZ, 2013), com o principal objetivo de relacionar os comportamentos globais das regras com a configuração local das células.

Nesse campo, conjectura-se que ACs com poder computacional localizam-se em uma região entre regras com comportamento ordenado e periódico e regras com comportamento caótico (LANG-TON, 1990). Esses comportamentos já estão precisamente definidos para o espaço dos ACs elementares.

Contudo, criar um método de classificação automatizado e consistente é essencial para que seja possível expandir tais definições para espaços mais amplos. Tal generalização

deve ser capaz de tratar a alta variabilidade que pode surgir a partir de uma mesma regra com configurações iniciais diferentes.

Neste projeto serão utilizados os dados coletados de vetores de textura de evoluções temporais de autômatos celulares unidimensionais binários. Como descrito por (NOGUEIRA, 2019), é possível utilizar vetores de textura extraídos da evolução temporal das execuções das regras para representá-las, na tentativa de desenvolver um sistema autorreferente de classificação dinâmica resiliente essa alta variabilidade para que possa ser generalizado para espaços maiores.

2 REFERENCIAL TEÓRICO

Neumann (1966) desenvolveu um dos trabalhos que tornou possível o desenvolvimento da área de autômatos celulares. Nesse ele desenvolve a teoria de uma máquina capaz de se autorreplicar com base na interação local entre suas partes. Desde então estudos sobre suas propriedades dinâmicas bem como aplicações de sua estrutura dinâmica localmente definida à modelagem de sistemas complexos reais vêm sendo desenvolvidas.

No espaço dos Autômatos Celulares Elementares (ACEs), constituído por regras que atuam sobre triplas binárias apenas, existe uma regra que apresenta de computabilidade universal (COOK, 2004). Tal regra é conhecida como Regra 110, nome que recebe a partir de seu número no esquema de numeração estabelecido por Wolfram.

Duas das questões colocadas em (WOLFRAM, 1985) sobre a dinâmica típica dos ACs são as seguintes, em tradução livre: “Que classificação geral sobre o comportamento dos autômatos celulares pode ser feita?” e “Como estão distribuídos os diferentes comportamentos ao longo de um espaço de regras de autômatos celulares?”. Wolfram (1985) respondeu parcialmente às duas questões fornecendo um esquema de classificação geral e explorando como a quantidade de regras em cada uma dessas classes varia de acordo com o espaço de regras considerado.

A classificação de complexidade dada por Wolfram (1984) toma como base observações experimentais das evoluções temporais típicas de uma regra e particiona os ACs em quatro classes, a saber:

- Classe I (C I): configurações pertencentes a evoluções temporais de regras desta classe tipicamente tornam-se homogêneas após um certo número de iterações. Em outras palavras, todas as células de uma configuração chegam a um mesmo estado;

- Classe II (C II): evoluções temporais típicas de regras desta classe atingem um ponto-fixo ou um regime periódico após um transiente;
- Classe III (C III): padrões não periódicos e aparentemente caóticos surgem ao longo das evoluções temporais típicas de regras desta classe;
- Classe IV (C IV): regras pertencentes a esta classe geram configurações nas quais surgem tanto regiões periódicas com comportamento bem-definido quanto regiões com comportamento caótico, havendo interação entre tais estruturas.

Conjectura-se (WOLFRAM, 2002) que regras da Classe IV apresentam computabilidade universal.

No trabalho Li e Packard (1990), foi desenvolvida uma partição mais fina dessas regras, subdividindo a Classe II em duas classes.

Langton (1990) conjecturou que regras que apresentam capacidade de armazenamento e processamento complexo de informações encontram-se numa região denominada “beira do caos”, localizada numa zona intermediária entre as regras da Classe II de Wolfram e as regras da Classe III de Wolfram. Tendo essas regras capacidade de realizar computações de maneira emergente, fica clara a relevância de se estabelecer a topologia dos espaços de regras de ACs de acordo com a classificação de suas regras.

Dada a falta de praticidade, ou até impossibilidade, de classificar manualmente espaços maiores de regras de ACs, a necessidade de uma classificação automática leva à busca de técnicas consolidadas para detecção e classificação de padrões (WITTEN; FRANK, 2002; CASTRO; FERRARI, 2016).

A fim de testar a correlação entre os dados selecionados de forma automatizada, serão utilizados algoritmos clássicos e consistentes de classificação. Nesse contexto, foram selecionados o k-NN (k-Nearest Neighbors) e o percéptrons de múltiplas camadas (PMCs), ambos algoritmos do contexto de Aprendizagem de Máquina, necessitando de treinamento.

O algoritmo k-NN trata de classificar os elementos tentando minimizar a diferença relativa as características dos elementos da classe em questão e as características do elemento a ser classificado.

O percéptrons de múltiplas camadas trata de somar as diferentes características com diferentes pesos, repetidamente em várias camadas. No treinamento são estimadas as características mais significativas de cada classe e suas respectivas correlações. Assim, ao aplicar as características de um novo elemento às fórmulas resultantes, é possível realizar uma predição de sua classe.

3 METODOLOGIA

Os experimentos do projeto foram majoritariamente implementados e testados usando o Wolfram Mathematica, exceto para os testes de classificação do algoritmo KNN, que foi implementado em Python.

Para representar a evolução temporal dos autômatos celulares unidimensionais binários foi utilizada uma matriz binária (apenas 1 e 0 presentes). E dessa forma é possível associar quadrados brancos para 0 e pretos para 1 (Figura 1).

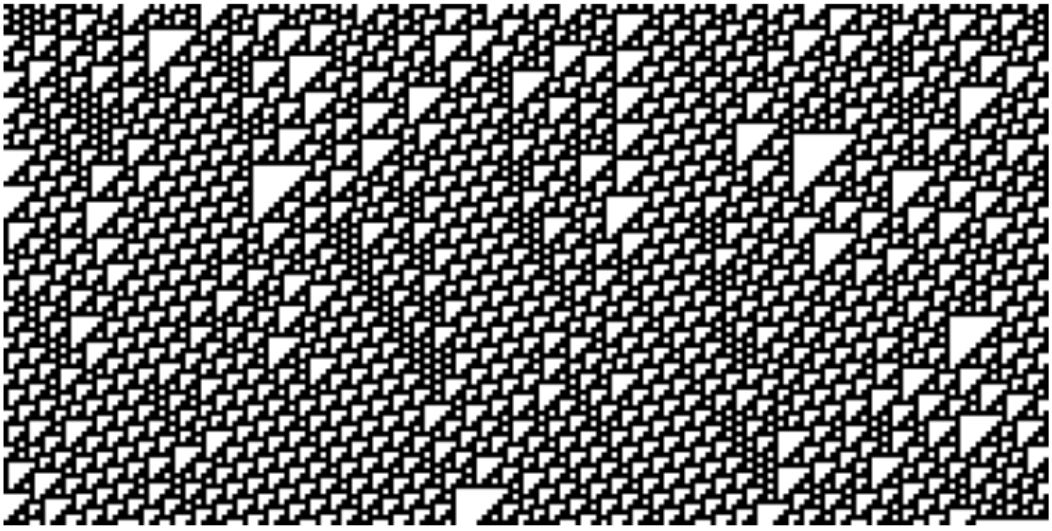

Figura 1. Exemplo de execução da regra 101.Cada linha representa uma execução da regra, sendo a primeira linha a configuração inicial.

Primeiramente foi implementada uma função parar gerar dos exemplos de janelas de execução de regra dos ACs. Para os testes realizados nesse projeto a função simula 45 execuções da regra selecionada a execução da regra a partir de um vetor de 25 posições binarias de valor randômico. Em seguida seleciona as últimas 25 execuções e retorna uma matriz 25 por 25, ou seja, são ignoradas as 30 primeiras execuções da simulação. Esse processo é feito para minimizar o impacto do trecho transiente das execuções (Figura 2), no qual a real complexidade da regra possa ainda não ser clara.

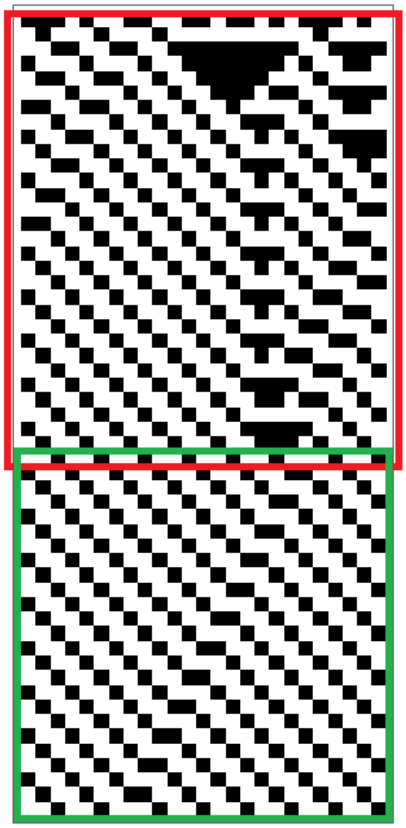

Figura 2. Exemplo de execução da regra 145 (Classe 2 de Wolfram) com entradas aleatórias, em vermelho o trecho transiente e em verde o trecho retornado.

Em seguida, definindo a extração das texturas, é adotado um parâmetro d que dita o tamanho textura, d elementos de altura e d elementos de largura. E a partir desse quadrado, é extraído um valor binário, que é então convertido para decimal, como é exemplificado na Figura 3, utilizando d igual a 3.

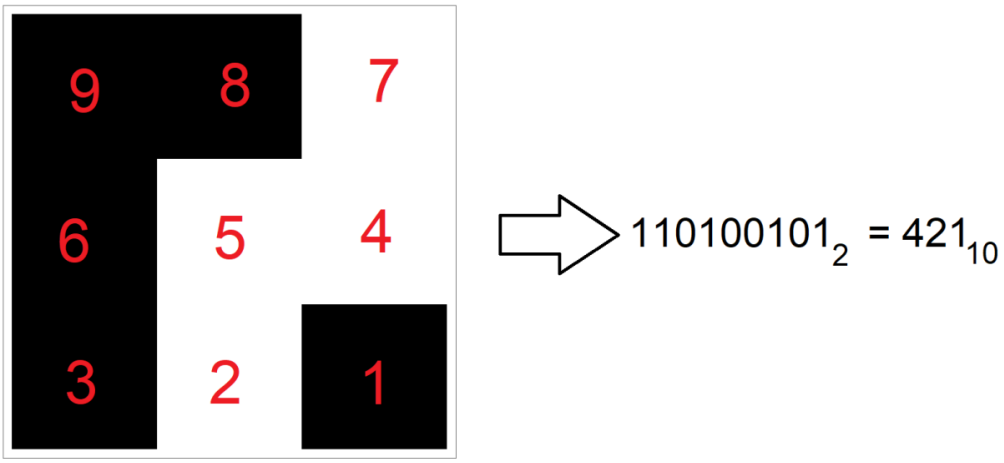

Figura 3. Exemplo de extração de textura com tamanho 3 por 3.

Dado esse modelo, foi desenvolvida a função responsável por coletar essas texturas se deslocando um elemento por vez até o fim da linha (utilizando a primeira e segunda coluna para os últimos elementos da linha) e retornar um vetor com a contagem de cada uma das *2^(d^d)* texturas, totalizando 512 nesse caso.

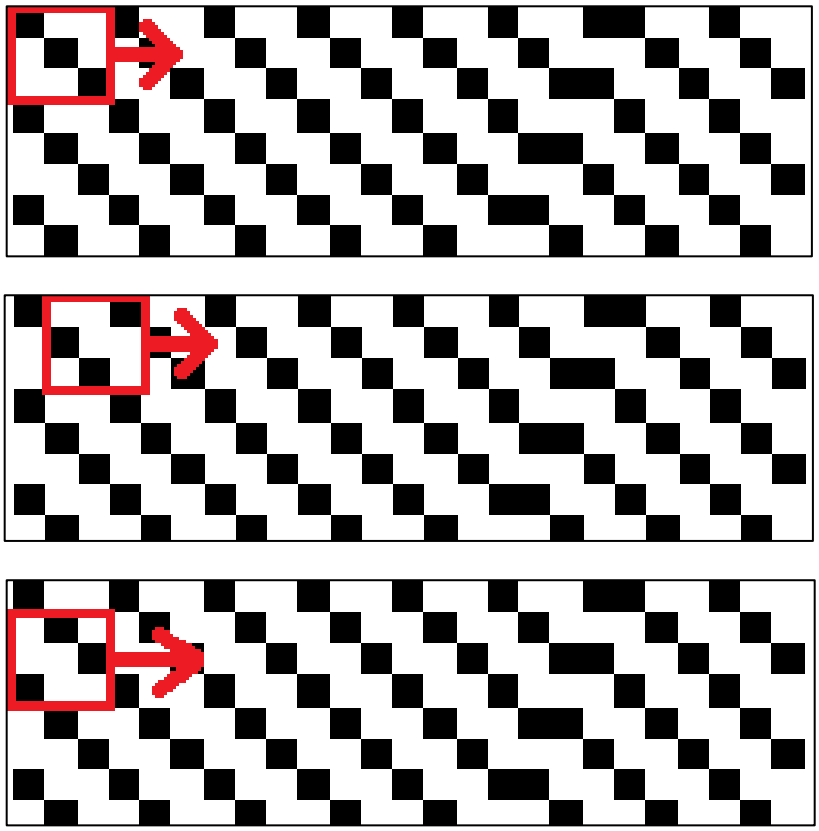

Figura 4. Exemplo da progressão de coleta das texturas.

Essas texturas são contabilizadas em um vetor, de maneira que o vetor é inicializado com 512 posições com zeros. Para cada ocorrência de uma textura, é utilizado seu valor para incrementar 1 a posição desse valor. Por exemplo: ao ler-se a textura 421 pela primeira vez naquela extração, o 421º elemento do vetor deixa de 0 e se torna 1. Caso passe por outra ocorrência da textura 421, o valor deixa de ser 1 e se torna 2.

Simulando uma extração de texturas com d igual a 2, tendo apenas 16 texturas diferentes possíveis, como pode ser visto na Figura 5.

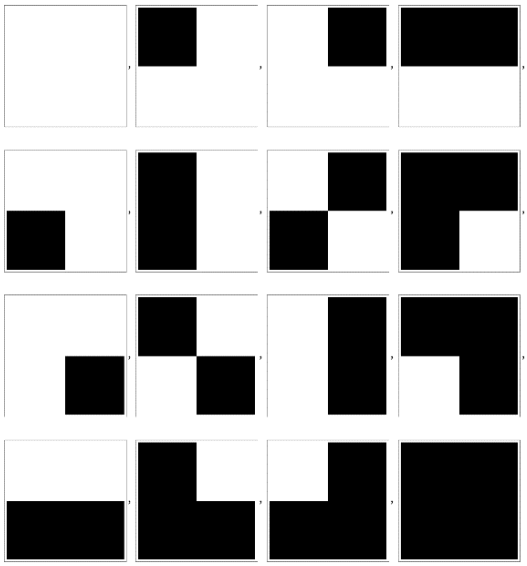

Figura 6. Todas as 16 texturas possíveis de d igual a 2.

Assim, partindo para uma execução da regra 110 apresentada na Figura 6.

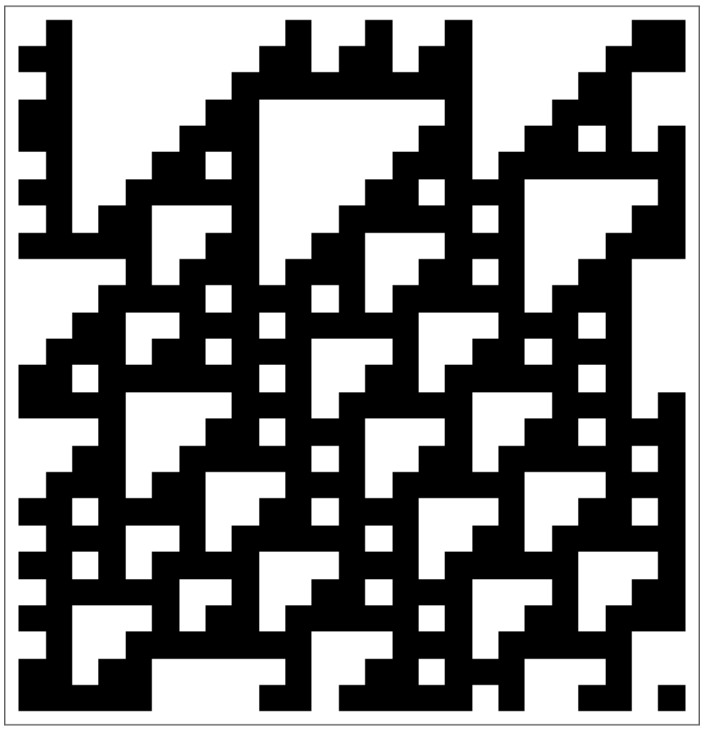

Figura 6. Evolução temporal de tamanho 25 por 25 da regra 110 com configurações iniciais aleatórias.

Ao associar cada textura a um valor binário, é possível armazenar a contagem das ocorrências em um vetor de 16 posições, da forma apresentada na Figura 7.

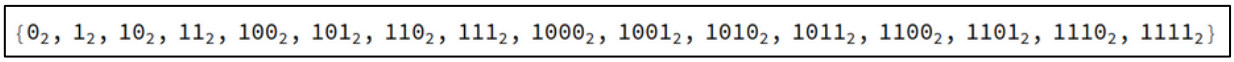

Figura 7. Posições do vetor correspondentes a cada textura de tamanho 2 por 2.

Terminando a extração e contagem das texturas nesse exemplo, temos o vetor resultante na Figura 8.

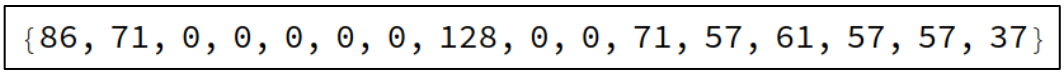

Figura 8. Vetor de contagem das texturas resultante da simulação.

Nos processos do projeto foram utilizadas texturas adotando d igual 3.

O procedimento é realizado com todas as execuções das regras e são obtidos esses respectivos vetores com a contagem das texturas. A partir desses vetores, foram selecionadas 14 características para representá-los:

- Maior frequência,
- Menor frequência,
- Maior diferença entre frequências (diferentes de zero),
- Diferença entre a maior frequência e a segunda maior,
- Média desconsiderando zero,
- Mediana desconsiderando zero,
- Maior valor da Lista de Moda (diferente de zero),
- Desvio Padrão,
- Quantidade de frequências diferentes,
- Quantidade de frequências diferentes de zero,
- Maior valor do vetor normalizado,
- Menor valor do vetor normalizado diferente de zero,
- Maior valor da moda (diferente de zero) vetor normalizado,
- Menor valor da moda vetor normalizado.

Em seguida foram implementados os algoritmos Percéptrons de Múltiplas Camadas (MLP) no Wolfram Mathematica e K-Nearest Neighbor (KNN) em Python utilizando a biblioteca scikit-learn. Também foi implementado o Ensemble, combinando os dois algoritmos e selecionando o algoritmo com maior correlação para cada exemplo em específico.

Para treinar os algoritmos foram classificadas todas as 256 regras com base na definição de Wolfram (1984) a partir das classes definidas para cada regra na ferramenta Wolfram Alpha. Foram utilizados três exemplos com configurações iniciais aleatórias de cada regra para treinar os algoritmos, a fim conseguir utilizar mais utilizar mais variações de cada regra para treino

Para testar os algoritmos, após o treino, foram utilizados 20 exemplos diferentes de cada regra com outras configurações iniciais aleatórias.

Em seguida, separadamente, foram feitos testes utilizando três vezes mais exemplos de regras de Classe 4, já que sua baixa quantidade em relação as outras regras (10 regras de Classe 4 em 256 regras) podem dificultar o treino dos algoritmos. O valor específico de três vezes foi escolhido pois o erro mais comum do algoritmo é regras de Classe 4 são erroneamente classificadas como com regras de Classe 3 pelo algoritmo, e como temos 26 regras (do total de 256) de Classe 3, foram usados esses 30 exemplos de Classe 4 (10 vezes três) para balancear.

4 RESULTADO E DISCUSSÃO

Os resultados dos testes foram apresentados por meio de três matrizes de confusão por cada execução, tendo no eixo vertical as classificações do algoritmo e no horizontal a classificação correta (baseada na classificação obtida no Wolfram Alpha). São elas

- Uma matriz com a soma dos valores de cada classificação,
- Uma matriz normalizada, adotando a soma de todas as classificações como 1,
- Uma matriz normalizada, adotando a soma de todas as classificações como o total de regras únicas classificadas.

Essas tabelas são apresentadas para cada algoritmo de Machine Learning utilizado, Percéptrons de Múltiplas Camadas (MLP), K-Nearest Neighbor (KNN) e Ensemble.

Assim, os resultados do primeiro teste, utilizando 3 exemplos de cada regra no treino dos algoritmos, estão exibidos nas Tabela 1 à Tabela 9. Sendo os resultados das tabelas 1 a 3 resultantes do algoritmo MLP, das tabelas 4 a 6 resultantes do algoritmo KNN, e das tabelas 7 a 9 resultantes do algoritmo Ensemble.

	1	2	3	4	Total
1	479	1	0	0	480
2	43	3777	50	50	3920
3	0	92	369	59	520
4	0	86	51	63	200
Total	522	3956	470	172	5120

Tabela 1. Primeiro resultado do MLP do primeiro teste.

	1	2	3	4	Total
1	0.0935547	0.000195313	0.	0.	0.09375
2	0.00839844	0.737695	0.00976563	0.00976563	0.765625
3	0.	0.0179688	0.0720703	0.0115234	0.101563
4	0.	0.0167969	0.00996094	0.0123047	0.0390625
Total	0.101953	0.772656	0.0917969	0.0335938	1.

Tabela 2. Segundo resultado do MLP do primeiro teste.

	1	2	3	4	Total
1	23.95	0.05	0.	0.	24.
2	2.15	188.85	2.5	2.5	196.
3	0.	4.6	18.45	2.95	26.
4	0.	4.3	2.55	3.15	10.
Total	26.1	197.8	23.5	8.6	256.

Tabela 3. Terceiro resultado do MLP do primeiro teste.

	1	2	3	4	Total
1	479	1	0	0	480
2	31	3814	64	11	3920
3	1	86	423	10	520
4	0	79	110	11	200
Total	511	3980	597	32	5120

Tabela 4. Primeiro resultado do KNN do primeiro teste.

	1	2	3	4	Total
1	0.0935547	0.000195313	0.	0.	0.09375
2	0.00605469	0.744922	0.0125	0.00214844	0.765625
3	0.000195313	0.0167969	0.0826172	0.00195313	0.101563
4	0.	0.0154297	0.0214844	0.00214844	0.0390625
Total	0.0998047	0.777344	0.116602	0.00625	1.

Tabela 5. Segundo resultado do KNN do primeiro teste.

	1	2	3	4	Total
1	23.95	0.05	0.	0.	24.
2	1.55	190.7	3.2	0.55	196.
3	0.05	4.3	21.15	0.5	26.
4	0.	3.95	5.5	0.55	10.
Total	25.55	199.	29.85	1.6	256.

Tabela 6. Terceiro resultado do KNN do primeiro teste.

	1	2	3	4	Total
1	479	1	0	0	480
2	31	3814	64	11	3920
3	1	86	423	10	520
4	0	79	110	11	200
Total	511	3980	597	32	5120

Tabela 7. Primeiro resultado do Ensemble do primeiro teste.

	1	2	3	4	Total
1	0.0935547	0.000195313	0.	0.	0.09375
2	0.00605469	0.744922	0.0125	0.00214844	0.765625
3	0.000195313	0.0167969	0.0826172	0.00195313	0.101563
4	0.	0.0154297	0.0214844	0.00214844	0.0390625
Total	0.0998047	0.777344	0.116602	0.00625	1.

Tabela 8. Segundo resultado do Ensemble do primeiro teste.

	1	2	3	4	Total
1	23.95	0.05	0.	0.	24.
2	1.55	190.7	3.2	0.55	196.
3	0.05	4.3	21.15	0.5	26.
4	0.	3.95	5.5	0.55	10.
Total	25.55	199.	29.85	1.6	256.

Tabela 9. Terceiro resultado do Ensemble do primeiro teste.

A partir disso também é possível extrair a taxa de acerto dos algoritmos nesse caso, como é possível ver na Tabela 10.

	C 1	C 2	C 3	C 4
MLP	99.79	96.35	70.96	31.50
KNN	99.79	97.30	81.35	5.500
Ensemble	99.79	97.30	81.35	5.500

Tabela 10. Taxa de acerto do primeiro teste em porcentagem.

Dados os resultados dessa primeira execução, é possível verificar que todos os algoritmos apresentam uma baixa taxa de acerto ao classificar regras de Classe 4. Nesse caso o MLP foi o que apresentou a maior taxa de acerto, 31.5%.

Além disso, o algoritmo utilizando Ensemble exibiu o mesmo comportamento que o KNN.

A fim de aprimorar o treino para classificação de regras de Classe 4, como mencionado na Metodologia, foram adicionados mais exemplos de execuções dessas regras, totalizando três vezes o número de execuções anterior.

Da mesma forma, os resultados desse segundo teste estão exibidos nas Tabela 11 a Tabela 19. Sendo os resultados das tabelas 11 a 13 resultantes do algoritmo MLP, das tabelas 14 a 16 resultantes do algoritmo KNN, e das tabelas 17 a 19 resultantes do algoritmo Ensemble.

	1	2	3	4	Total
1	480	0	0	0	480
2	36	3771	33	80	3920
3	2	59	355	104	520
4	0	53	142	405	600
Total	518	3883	530	589	5520

Tabela 11. Primeiro resultado do MLP do segundo teste.

	1	2	3	4	Total
1	0.0869565	0.	0.	0.	0.0869565
2	0.00652174	0.683152	0.00597826	0.0144928	0.710145
3	0.000362319	0.0106884	0.0643116	0.0188406	0.0942029
4	0.	0.00960145	0.0257246	0.0733696	0.108696
Total	0.0938406	0.703442	0.0960145	0.106703	1.

Tabela 12. Segundo resultado do MLP do segundo teste.

	1	2	3	4	Total
1	24.	0.	0.	0.	24.
2	1.8	188.55	1.65	4.	196.
3	0.1	2.95	17.75	5.2	26.
4	0.	2.65	7.1	20.25	30.
Total	25.9	194.15	26.5	29.45	276.

Tabela 13. Terceiro resultado do MLP do segundo teste.

	1	2	3	4	Total
1	480	0	0	0	480
2	29	3751	19	121	3920
3	1	68	292	159	520
4	0	20	74	506	600
Total	510	3839	385	786	5520

Tabela 14. Primeiro resultado do KNN do segundo teste.

	1	2	3	4	Total
1	0.0869565	0.	0.	0.	0.0869565
2	0.00525362	0.679529	0.00344203	0.0219203	0.710145
3	0.000181159	0.0123188	0.0528986	0.0288043	0.0942029
4	0.	0.00362319	0.0134058	0.0916667	0.108696
Total	0.0923913	0.695471	0.0697464	0.142391	1.

Tabela 15. Segundo resultado do KNN do segundo teste.

	1	2	3	4	Total
1	24.	0.	0.	0.	24.
2	1.45	187.55	0.95	6.05	196.
3	0.05	3.4	14.6	7.95	26.
4	0.	1.	3.7	25.3	30.
Total	25.5	191.95	19.25	39.3	276.

Tabela 16. Terceiro resultado do KNN do segundo teste.

	1	2	3	4	Total
1	480	0	0	0	480
2	29	3751	19	121	3920
3	1	68	292	159	520
4	0	20	74	506	600
Total	510	3839	385	786	5520

Tabela 17. Primeiro resultado do Ensemble do segundo teste.

	1	2	3	4	Total
1	0.0869565	0.	0.	0.	0.0869565
2	0.00525362	0.679529	0.00344203	0.0219203	0.710145
3	0.000181159	0.0123188	0.0528986	0.0288043	0.0942029
4	0.	0.00362319	0.0134058	0.0916667	0.108696
Total	0.0923913	0.695471	0.0697464	0.142391	1.

Tabela 18. Segundo resultado do Ensemble do segundo teste.

	1	2	3	4	Total
1	24.	0.	0.	0.	24.
2	1.45	187.55	0.95	6.05	196.
3	0.05	3.4	14.6	7.95	26.
4	0.	1.	3.7	25.3	30.
Total	25.5	191.95	19.25	39.3	276.

Tabela 19. Terceiro resultado do Ensemble do segundo teste.

Da mesma forma, na Tabela 20 temos a taxa de acerto desse segundo teste.

	C 1	C 2	C 3	C 4
MLP	100.0	96.20	68.27	67.50
KNN	100.0	95.69	56.15	84.33
Ensemble	100.0	95.69	56.15	84.33

Tabela 20. Taxa de acerto do primeiro teste em porcentagem.

Dado os resultados desse segundo teste, foi possível aumentar significativamente a precisão do algoritmo no caso da classificação da Classe 4, os algoritmos do KNN e do Ensemble apresentaram uma taxa de acerto das regras de Classe 4 de aproximadamente 84%. Contudo diminuiu os acertos para regras da Classe 3. Considerando o MLP que exibiu a taxa mínima de acerto mais alta, temos a taxa de acerto mínima como 67.5%.

5 CONSIDERAÇÕES FINAIS

Dado o exposto, é possível verificar que existe uma forte correlação entre a quantidade de texturas extraídas da evolução temporal dos ACs binários elementares e a as classificações das regras nas 4 classes de Wolfram.

Vale ressaltar que as evoluções temporais de uma regra de Classe 3 ou Classe 4, pode exibir o mesmo comportamento que uma regra de Classe 1 ou Classe 2, dependendo do das configurações iniciais. Fazendo com que o método jamais seja 100% eficaz.

Contudo, ainda é possível explorar essa técnica com tamanhos de texturas maiores, por exemplo, adotando o tamanho da textura 4 por 4.

Além disso, há espaço para otimizar as configurações dos algoritmos de Aprendizagem de Máquina utilizados, podendo ser feita uma rede especializada em diferenciar as classes C3 e C4, uma vez que são elas as mais problemáticas, ou até testar a efetividade de outros algoritmos.

6 AGRADECIMENTOS

Agradeço a atenção e auxílio do ao professor orientador Eurico Ruivo, que me introduziu no tema pacientemente e me auxiliou durante todo o projeto.

Também agradeço aos colegas que trabalharam ao meu lado e me auxiliaram durante a pesquisa.

Agradeço ao PIBIC MackPesquisa, que me proporcionou apoio por meio da bolsa para Iniciação Cientifica.

E agradecemos à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo projeto STIC-AmSud (CoDANet) nº 88881.197456/2018-01.

7 REFERÊNCIAS

CASTRO, L. N. d.; FERRARI, D. G. Introdução à mineração de dados: conceitos básicos, algoritmos e aplicações. São Paulo: Saraiva, 2016.

CHUA, L. O.; YOON, S.; DOGARY, R. A nonlinear dynamics perspective of Wolfram's New Kind of Science Part 1: Threshold of Complexity. International Journal of Bifurcation and Chaos, v. 12, n. 12, p. 2655–2766, 2002.

COOK, M. Universality in elementary cellular automata. Complex Systems, v. 15, n. 1, p. 1–40, 2004.

HOEKSTRA, A.; KROC, J.; SLOOT, P. Introduction to modeling of complex systems using cellular automata. Simulating Complex Systems by Cellular Automata, Springer, p. 1–16, 2010.

KARI, J. Theory of cellular automata: A survey. Theoretical Computer Science, Elsevier, v. 334, n. 1–3, p. 3–33, 2005.

LANGTON, C. Computation at the edge of chaos: Phase transitions and emergent computation. Physica D: Nonlinear Phenomena, Elsevier, v. 42, n. 1-3, p. 12–37, 1990.

LI, W. Phenomenology of nonlocal cellular automata. Journal of Statistical Physics, v. 68, n. 5-6, p. 829–882, 1992.

LI, W.; PACKARD, N. The structure of the elementary cellular automata rule space. Structure, v. 4, p. 281–297, 1990.

MARTINEZ, G. J. A note on elementary cellular automata classification. arXiv preprint – arXiv:1306.5577, 2013.

MITCHELL, M. Computation in cellular automata: A selected review lattice. NonStandard

Computation, p. 1–42, 1998.

NEUMANN, J. V. Theory of self–reproducing automata. IEEE Transactions on Neural Networks, v. 21, n. 1, p. 3–14, 1966.

NOGUEIRA, M. A. Classificação automática do comportamento dinâmico de autômatos celulares binários. Tese (Doutorado) — Universidade Presbiteriana Mackenzie, 2019.

WITTEN, I. H.; FRANK, E. Data mining: practical machine learning tools and techniques with java implementations. ACM Sigmod Record, ACM New York, NY, USA, v. 31, n. 1, p. 76–77, 2002.

WOLFRAM, S. Computation theory of cellular automata. Communications in Mathematical Physics, v. 96, n. 1, p. 15–57, 1984.

WOLFRAM, S. Twenty problems in the theory of cellular automata. Physica Scripta, Institute of Physics Publishing, T9, n. 3, p. 170–183, 1985.

WOLFRAM, S. A New Kind of Science. [S.l.]: Wolfram Media, 2002.

Contatos: arthurmoreirap@gmail.com e eurico.ruivo@mackenzie.br