Comparação entre os métodos *grid* e *quadtree* para compressão de dados em aplicações *online*

Rafael Fernandes Gonçalves da Silva
Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
rafaelfgs@ufmg.br

***Resumo*—Este trabalho consiste na utilização de métodos baseados em *grid* e *quadtree* para redução na amostragem e com algumas modificações de forma a tornar-se incremental. A aplicação de ambos os métodos gera uma superfície de densidade, sendo fixa para *grid* e variável para o *quadtree*. A avalição dos métodos é realizada através dos valores da acurácia e da economia de alocação de memória, esta representada pela quantidade de espaços utilizados na superfície. Em ambos os métodos é feita uma combinação entre a superfície criada no pacote anterior e no atual, utilizando um fator de esquecimento para dar um peso maior nos dados antigos ou nos atuais. Os testes são feitos principalmente na base de dados disponibilizada *SyntheticDatasetRodrigo*. Por fim, são utilizados outras duas base de dados: *sea* e *fourclass*, onde é possível observar as propriedades do *grid* e do *quadtree* através de outros métodos de entradas de dados.**

Palavras-chave*—Classificação *online*, *grid*, *quadtree

I. Introdução

Os métodos incrementais de classificação estão se tornando cada vez mais importantes nos dias atuais. O avanço tecnológico provocou um aumento gigantesco na criação e na utilização de informações envolvendo diversos campos distintos, como economia, telecomunicações, plataformas sociais, indústrias das mais diversas formas, entre outros. Em muitos desses campos, a informação necessita ser tratada em tempo real, já que o acesso à sua totalidade pode não estar disponível, com os dados sendo recebidos em formas de pacotes, e ainda podem ocorrer mudanças de comportamento dos dados ao longo do tempo.

Classificadores em tempo real são bastante comuns, sendo amplamente utilizados na área de visual computacional. Os trabalhos propostos em [1], [2] e [3] utilizam redes neurais convolucionais e algoritmos de forma *online* para classificar e reconhecer objetos à medida em que chegam os dados. Especificamente, conforme citado em [1], o reconhecimento de faces em tempo real torna-se mais interessante, onde uma nova face de entrada não acarreta uma total reclassificação dos dados antigos.

A classificação de forma *online* pode ser aplicada em dados relacionados à internet, como em [4] e *ibrahim2016internet*, onde são utilizados algoritmos de aprendizado de máquina para detectar e classificar tráfego de internet. Pode ser também aplicada no reconhecimentos de palavras em textos e/ou documentos para identificar e adaptar perfis de usuário em tempo real, conforme explicado em [5], ou para reconhecer situações de emergência em mensagens de usuários em mídias sociais, como explicado em [6].

Há também aplicações para a área financeira, onde [7] e [8] utiliza métodos de classificação em tempo real para detectar fraudes em transações financeiras e manipulações de relatórios para ocultações de perdas financeiras.

Os métodos de classificação incremental mais comuns necessitam de uma memória representativa dos dados, visto que o armazenamento de dados no seu formato original em um sistema contínuo por longos períodos de tempo torna-se inviável. Além disso, pode haver mudanças de conceito, onde o armazenamento de dados antigos passa a ser desnecessário após um tempo. Dessa forma, a compressão e representação de dados torna-se importante no contexto de classificação de grandes bases de dados e de métodos incrementais.

Há vários métodos para a realização da compressão de dados. Em [9], é realizada uma fragmentação de dados hidrográficos em subconjuntos na forma de *grids*, resultando em uma subamostragem. A redução do consumo de memória pode ser feita também através de grids hexagonais, como realizado em [10] utilizando diferentes dados de sinais e imagens. O método de *quadtree* é utilizado em [11] para a realização de representações espaciais de imagens tomadas por utilizando Veículos Aéreos Não-Tripulados durante um patrulhamento. Pode-se também realizar uma comparação entre os diferentes métodos, como é feito em [12], onde são realizados e comparados métodos de busca de hiper-parâmetros de um *Support Vector Machine* através de *gridsearch* e *quadtree*, ou como em [13], onde são comparadas as perdas de compressão dos métodos *block-encoding* e *octree* utilizando imagens de mapas como bases de dados.

Partindo dessas contextualizações, o presente trabalho tem por objetivo a comparação de dois métodos de compressão de dados: *grid* e *quadtree*, implementados de forma a funcionar para um fluxo de dados. São utilizadas três base de dados: *SyntheticDatasetRodrigo.csv*, *sea.data* e *fourclass*, sendo as duas primeiras disponibilizadas previamente e a última disponível no pacote *Evidential Clustering* (*evclust*) do repositório *CRAN* para a linguagem R. O código foi criado utilizando a base de dados *SyntheticDatasetRodrigo*, onde foi observado o seu comportamento para uma dada variação dos parâmetros de entrada.

II. METODOLOGIA

De uma forma bastante resumida, a metodologia utilizada consiste em comprimir uma base de dados para um formato de *grid* e um de *quadtree*, gerando uma probabilidade para cada intervalo dos formatos e partir disso criar um modelo de classificação.

O código implementado foi separado em quatro partes, descritas nas subseções seguintes.

A. Entradas do Algoritmo

A leitura da base de dados é o primeiro passo realizado pelo algoritmo. Para ficar de forma padronizada, a base de dados deve estar no formato matricial com três colunas e n linhas, onde as colunas representam a primeira variável, a segunda variável e os rótulos e as linhas representam as amostras. A base de dados *sea* possui três variáveis e teve de ser dividida em três bases de dados de forma a ser aplicada em pares de variáveis: 1 com 2, 1 com 3 e 2 com 3.

Em seguida são determinadas as constantes globais do algoritmo, sendo um total de quatro: *forg_factor* representa o fator de esquecimento, ligado à relação entre as antigas e novas representações dos dados; *batch_size* corresponde ao tamanho do pacote de entrada dos dados, nesse caso a base de dados foi dividida em pacotes de tamanhos iguais; *num_parts* está relacionado ao número de divisões do espaço, onde o espaço é divido em *num_parts* x *num_parts* para o *grid* e o número de levels do *quadtree* é dado por $\lfloor log_2(num\_parts) \rfloor + 1$; e *apply_filt* indica se será realizada ou não a filtragem para o método do *grid*, sendo esta uma constante binária.

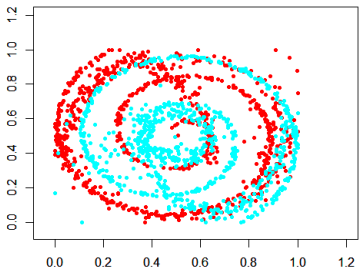

Figura 1. Amostras da base de dados *SyntheticDatasetRodrigo*.

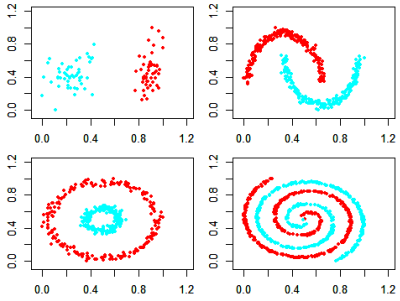

Figura 2. Amostras da base de dados disponibilizada dividida em cada conceito. A ordem temporal dos dados ocorre primeiramente acima da esquerda para a direita.

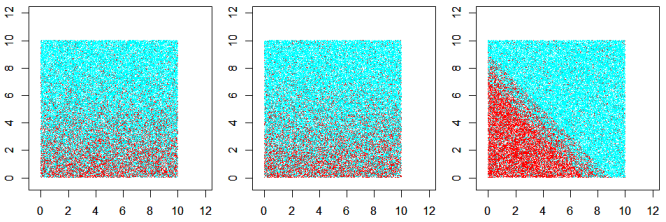

Figura 3. Amostras da base de dados *sea*. Da direita para a esquerda estão representadas as variáveis 1 e 2, 1 e 3, 2 e 3.

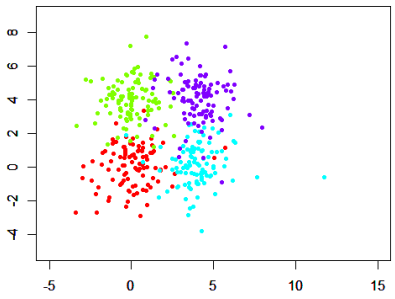

Figura 4. Amostras da base de dados *fourclass*.

As bases de dados utilizadas estão representadas nas Figuras 1, 3 e 4, sendo esta segunda dividida em pares de variáveis. A Figura 2 mostra as mudanças de conceito da base de dados, onde os primeiros 100 dados são gaussianas separáveis, do 101 ao 500 são duas parábolas invertidas, do 501 ao 900 são dois círculos concêntricos e do 901 ao 1700, duas espirais.

B. Funções Necessárias

Nessa parte do algoritmo encontram-se as funções necessárias para o funcionamento do código. São aqui trabalhadas duas variáveis principais: matrizes de probabilidade e modelo de classificação. A primeira representa as probabilidades de cada classe para um dado intervalo, apresentada na Tabela I. Possui um formato n x $(4+k)$, onde n é o número de intervalos necessários, as quatro primeiras colunas são os valores desses intervalos para as variáveis 1 e 2 e as k próximas colunas são as probabilidades de cada classe (note que n e k podem variar com o tempo). O modelo de classificação é similar à matriz de probabilidades, possui em formato n x 5, sendo que na quinta coluna estão os rótulos associado à cada intervalo, encontrados através da classe que possui a maior probabilidade naquele intervalo.

Tabela I
MATRIZ DE PROBABILIDADE PARA TRÊS CLASSES E QUATRO INTERVALOS.

$x1_{min}$	$x1_{max}$	$x2_{min}$	$x2_{max}$	$prob_1$	$prob_2$	$prob_3$
0	0.5	0	0.5	0	0.01	0.16
0	0.5	0.5	1	0.17	0.07	0.11
0.5	1	0	0.5	0.18	0.05	0
0.5	1	0.5	1	0.05	0.18	0.05

1) createGrid: Função utilizada para criar e combinar duas matrizes dados para o método *grid*. Possui sete parâmetros de entrada: os vetores das variáveis 1 e 2 das amostras da iteração atual, seus respectivos rótulos, as classes até então observadas, a matriz de probabilidade da iteração anterior e os intervalos desejados das variáveis 1 e 2. Inicialmente a matriz antiga de probabilidade é transformada em um grid do tamanho especificado. Em seguida é criado um *grid* com as probabilidades das amostras da iteração atual, essa probabilidade de um intervalo é calculada pela divisão entre do número de amostras do intervalo e número total de amostras. Em seguida é aplicado um filtro de forma a suavizar o *grid* de probabilidades (note que o filtro utilizado está normalizado de forma a não alterar a probabilidade total do *grid*. Em seguida é feita a combinação entre o *grid* antigo com o novo através do fator de esquecimento de forma a ponderar ambos os *grids* sem alterar a probabilidade total. Por fim, o *grid* resultante é transformado em uma matriz probabilidade, a qual representa o retorno da função.

2) combineTree: Função utilizada para combinar duas matrizes de probabilidade para o método de *quadtree*, visto que o intervalos presentes nas matrizes podem ser diferentes. Para isso, percorre-se as linhas de cada matriz e, quando é encontrado algum intervalo diferente, os valores de probabilidade da matriz com o intervalo maior são distribuídos entre os intervalos menores de forma balanceada. A combinação entre as matrizes agora equivalentes é feita através do fator de esquecimento de forma obter uma ponderação entre a representação antiga e a nova. A função retorna o resultado da combinação.

3) createTree: Esta função é utilizada para criar as matrizes de probailidade atual para o *quadtree*. Também possui sete parâmetros de entrada, similar a *createGrid*: os vetores das variáveis 1 e 2, os rótulos, as classes, a matriz de probabilidade e os intervalos. Para encontrar os intervalos do *quadtree*, é percorrido o limite máximo de divisões, mas a divisão somente é computada no intervalo em quatro condições: alcançar o limite máximo de divisões, houver apenas amostras de uma classe, não houver nenhuma amostra, ou houver uma classe em todo o espaço. De forma similar a probabilidade de um dado intervalo é o número de amostras nele dividido pelo número total de amostras. Ao final a função *combineTree* é chamada para combinar as matrizes de probabilidade antiga e nova.

4) createModel: Esta função recebe as matrizes de probabilidade (de qualquer um dos métodos) e, percorrendo cada linha da matriz, determina a classe correspondente ao intervalo através da maior probabilidade entre as classes. A função retorna o modelo de classificação, com os intervalos e a sua respectiva classe.

5) predModel: Função para determinar as classes das amostras de teste. Possui como entradas os vetores das amostras de teste (variáveis 1 e 2), os valores das classes já observadas e o modelo de classificação. Para determinar a classe de uma amostra, é observado seu respectivo intervalo no modelo de classificação e observado o índice correspondente. A função retorna as predições das amostras, de acordo com os índices e os valores das classes já observadas.

6) plotBase: Função com propriedades gráficas para exibir a base de dados atual na tela. Possui como entrada a base de dados e os limites das variáveis 1 e 2.

7) plotClassifier: Função com propriedades gráficas para exibir os resultados obtidos. Foi implementada de forma a exibir em tempo real o modelo do classificador, possuindo como entradas o modelo *grid*, o modelo *quadtree*, as classes e o tipo de exibição, onde valores verdadeiros exibem em tempo real (adicionando uma pausa e limpando o espaço de plotagem) e valores falsos exibem os modelos encontrados em todas as iterações. Para exibir os modelos, estes são transformados em formato de *grid* e mostrados como imagens.

C. Limites da base de dados

Nesta parte do algoritmo é encontrada a melhor forma de representação dos espaços, sendo calculados os limites máximos e mínimos de cada variável e, de acordo com a constante do número de divisões, as sequências dos intervalos do *grid* e do *quadtree*.

D. Classificador Online

Esta representa a parte principal do algoritmo, onde inicia-se as variáveis e chama-se as funções. Foi introduzido o conceito de quantidade de memória utilizada, sendo representada pelo número de intervalos necessários para cada método realizar a classificação. Para o *grid*, essa quantidade é sempre fixo, sendo igual ao quadrado da constante do número de divisões do espaço. Para o *quadtree*, essa quantidade pode variar de acordo com a posição dos dados de entrada.

Depois de iniciadas as variáveis da acurácia e de quantidade de memória, o código entra em um *loop* de acordo com o tamanho da base de dados e a constante global do tamanho dos pacotes de entrada. Primeiramente, são determinados os índices relacionados ao pacote atual, onde são selecionados 70% para treino do modelo e 30% para teste. Em seguida, são determinadas as variáveis das classes e das matrizes de probabilidades e, caso o número de classes se alterar, são adicionadas colunas nulas nas matrizes de probabilidades. Por fim, para cada um dos dois métodos, faz-se os seguintes passos: é atualizada a matriz de probabilidades de acordo com os dados de treino e a matriz da iteração anterior, é criado o modelo de classificação utilizando a matriz de probabilidades, são feitas as predições dos dados de teste com o modelo encontrado, é determinada a acurácia e a quantidade de memória utilizada pelo método. Ao final de cada iteração, pode-se exibir o resultado do modelo encontrado em tempo real. Depois de finalizado o *loop*, são exibidos alguns resultados finais.

III. Resultados e Discussão

Os testes principais para verificar o funcionamento e o desempenho do código foram realizados com a base de dados *SyntheticDatasetRodrigo*. Com essa base, foram realizados diferentes testes avaliando a resposta do código em função de diferentes valores de suas constantes globais. Ao final, o código também foi testado nas bases de dados *sea* e *fourclass*, porém com os parâmetros fixos.

Para a base de dados *SyntheticDatasetRodrigo*, foi estabelecido de forma empírica valores padrão das constantes globais, sendo *forg_factor* $= 2/3$, *batch_size* $= 32$, *num_parts* $= 16$ e *apply_filt* =VERDADEIRO. Os resultados obtidos com a variação desses parâmetros estão representados nas Figuras 5, 6, 7 e 8 e na Tabela II. Nessa tabela as constantes k_1, k_2, k_3 e k_4 são representadas pela mesma sequência citada acima.

A utilização de maiores pesos para os novos dados gera uma resposta mais rápida e mais instável tendo melhores resultados para classificações mais simples, porém piores para mais complexas. Enquanto que um peso maior para os dados antigos, reduz o tempo de resposta, porém atinge melhores resultados a longo prazo. Neste cenário, os resultados do *quadtree* foram pouco piores em relação ao *grid*, mas com uma redução de memória maior que 60%.

A redução do tamanho do pacote também gera uma instabilidade dos modelos, enquanto que pacotes muito grandes não são úteis em ambientes com frequentes mudanças de conceito. Nesse caso o *quadtree* tem uma maior instabilidade e uma maior economia de memória em pacotes pequenos, e, em pacotes maiores, quase se iguala ao *grid*, com uma redução de mais de 50% de memória.

O aumento do número de divisões do espaço provoca uma melhora dos resultados para o *grid*, juntamente com um aumento exponencial de memória. O *quadtree* mostrou-se indiferente para as variações realizadas, sendo melhor que o *grid* em 8 divisões (com redução de 30% de memória) e pior em 32 divisões (com redução de mais de 90%). A Figura 9 também demonstra essa redução no gasto de memória.

A aplicação do filtro no *grid* mostrou resultados melhores em classificações mais simples, enquanto que sua retirada provoca resultados melhores a longo prazo. Isso pode ser visto através das Figuras 10 e 11, onde o modelo perde desempenho para classes muito próximas. Esse fato pode também ser observado na Tabela III, onde o *grid* sem o filtro obteve resultados similares ao *quadtree*.

Foi também observada a capacidade reativa dos modelos. As Figuras 12 e 13 demonstram a adaptação do modelo no momento em que ocorre uma mudança de conceito. Pode-se observar que para o *grid* a mudança ocorre de uma forma mais suave e é possível verificar o formato dos dados, enquanto que no *quadtree*, as mudanças ocorrem somente onde é necessário (intervalos onde ocorrem duplicidade de classes diferentes), tornando-se mais econômico porém com uma pior aparência.

Por fim, utilizando as outras bases de dados, através da Tabela IV e da Figura 14, pode-se observar a uma grande oscilação (não crescente) no valor da acurácia e uma baixa economia de memória do *quadtree*, demostrando a complexidade de classificação da base *sea* (resultados gerados com as constantes iguais à $2/3$, 128, 32 e 1). Já para a base de dados *fourclass*, onde cada pacote representa uma classe diferente, o *quadtree* obteve um erro quase completo da seguanda classe. Esse fato deve-se à não ocorrência de sobreposições dentro de cada pacote, visto que não houve divisões do espaço fazendo com que os pontos da terceira classe fosse preditos para a primeira classe (resultados gerados para à $2/3$, 100, 8 e 1).

IV. Conclusão

Através dos resultados obtidos, pode-se concluir que os métodos utilizados podem ser utilizados em um contexto *online* de forma incremental, retendo informação de entradas anteriores e se adaptando a novas distribuições das amostras. Foi possível observar uma grande redução na utilização de memória, principalmente para o método *quadtree*, visto que sua divisão do espaço se adpta ao formato dos dados reduzindo o gasto em regiões de mesma ou com nenhuma classe.

De forma a aprimorar os resultados, são citados algumas possíveis atualizações para o código. Primeiramente, pode-se utilizar um espaço que possa variar e se adaptar de forma *online*, já que atualmente este é dado como fixo durante a execução do programa. A utilização de uma nova forma de *grid* também seria interessante para reduzir o crescente uso de memória com o aumento do número de intervalos. Pode-se ainda testar outras condições para o aumento dos níveis do *quadtree* e alguma forma de balanceamento do resultado. Por fim, como o código é restrito à duas variáveis, seria de grande vantagem alterações para o funcionamento para mais variáveis, ou inclusive para dados de imagens como entrada.

Referências

[1] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 233–248.

[2] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 761–769.

[3] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *International conference on machine learning*, 2015, pp. 597–606.

[4] A. B. Mohammed and S. Nor, "Near real time online flow-based internet traffic classification using machine learning (c4.5)," *International Journal of Engineering*, vol. 3, no. 4, pp. 370–379, 2009.

[5] A. Bouchachia, A. Lena, and C. Vanaret, "Online and interactive self-adaptive learning of user profile using incremental evolutionary algorithms," *Evolving Systems*, vol. 5, no. 3, pp. 143–157, 2014.

[6] J. Yin, S. Karimi, A. Lampert, M. Cameron, B. Robinson, and R. Power, "Using social media to enhance emergency situation awareness," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[7] S. Cao, X. Yang, C. Chen, J. Zhou, X. Li, and Y. Qi, "Titant: Online real-time transaction fraud detection in ant financial," *arXiv preprint arXiv:1906.07407*, 2019.

[8] H. Patel, S. Parikh, A. Patel, and A. Parikh, "An application of ensemble random forest classifier for detecting financial statement manipulation of indian listed companies," in *Recent Developments in Machine Learning and Data Analytics*. Springer, 2019, pp. 349–360.

[9] M. Wlodarczyk-Sielicka and A. Stateczny, "Fragmentation of hydrographic big data into subsets during reduction process," in *2017 Baltic Geodetic Congress (BGC Geomatics)*. IEEE, 2017, pp. 193–198.

[10] M. Knaup, S. Steckmann, O. Bockenbach, and M. Kachelrieß, "Ct image reconstruction using hexagonal grids," in *2007 IEEE Nuclear Science Symposium Conference Record*, vol. 4. IEEE, 2007, pp. 3074–3076.

[11] N. Basilico and S. Carpin, "Online patrolling using hierarchical spatial representations," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2163–2169.

[12] M. Beltrami and A. C. L. da Silva, "Grid-quadtree algorithm for support vector classification parameters selection," *Appl. Math. Sci*, vol. 9, pp. 75–82, 2015.

[13] P. P. Wai, S. S. Hlaing, K. L. Mon, M. M. Tin, and M. M. Khin, "Comparison between block-encoding and quadtree compression methods for raster maps," in *International Conference on Big Data Analysis and Deep Learning Applications*. Springer, 2018, pp. 258–263.

Tabela II
VALORES DE ACURÁCIA E MEMÓRIA PARA A BASE DE DADOS *SyntheticDatasetRodrigo* COM A VARIAÇÃO DOS PARÂMETROS.

Constantes Globais				grid		quadtree	
k_1	k_2	k_3	k_4	Acurácia(%)	Memória	Acurácia(%)	Memória
2/3	32	16	S	88.49±1.09	256	85.09±2.61	97±5
1/4	32	16	S	86.26±0.70	256	82.75±1.40	92±4
9/10	32	16	S	83.06±0.77	256	80.23±3.30	93±5
2/3	3	16	S	82.93±1.48	256	59.82±1.10	43±5
2/3	128	16	S	87.57±0.98	256	87.33±0.98	114±7
2/3	32	8	S	78.34±2.48	64	83.96±2.17	45±2
2/3	32	32	S	94.30±0.67	1024	84.94±1.18	97±2
2/3	32	16	N	88.04±1.92	256	-	-

Tabela III
VALORES DE ACURÁCIA E MEMÓRIA PARA A BASE DE DADOS *SyntheticDatasetRodrigo* NO MÉTODO OFFLINE.

grid com filtro		*grid* sem filtro		*quadtree*	
Acurácia(%)	Memória	Acurácia(%)	Memória	Acurácia(%)	Memória
74.99±0.73	256	77.92±1.02	256	77.88±1.09	162±4

Tabela IV
VALORES DE ACURÁCIA E MEMÓRIA PARA A BASE DE DADOS *sea*.

	grid		*quadtree*	
Variáveis	Acurácia(%)	Memória	Acurácia(%)	Memória
[1,2]	66.43±0.22	1024	65.82±0.41	776±5
[1,3]	66.06±0.40	1024	65.43±0.45	783±5
[2,3]	85.09±0.22	1024	83.10±0.18	773±7

Tabela V
VALORES DE ACURÁCIA E MEMÓRIA PARA A BASE DE DADOS *fourclass*.

grid		*quadtree*	
Acurácia(%)	Memória	Acurácia(%)	Memória
94.50±1.73	64	74.00±0.91	4

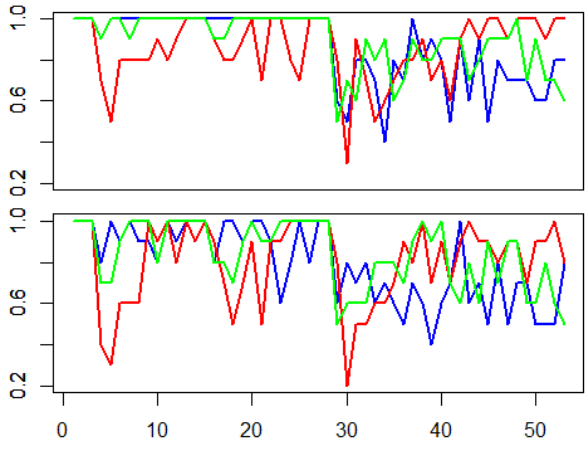

Figura 5. Acurácia encontrada variando-se o fator de esquecimento. Acima os resultados do *grid* e abaixo do *quadtree*. A linha azul corresponde a um fator de 1/4, verde de 2/3 e vermelho de 9/10.

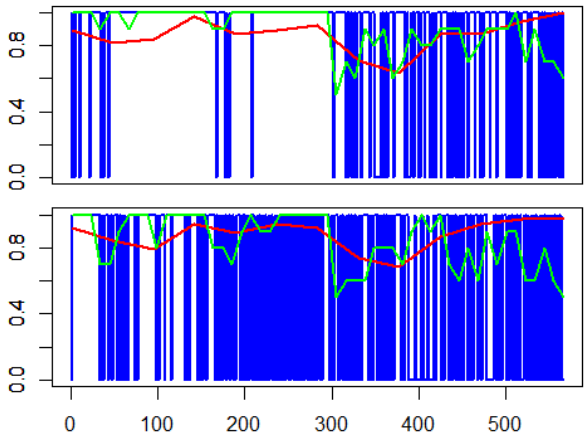

Figura 6. Acurácia encontrada variando-se o tamanho dos pacotes. Acima os resultados do *grid* e abaixo do *quadtree*. A linha azul corresponde a um tamanho de 3, verde de 32 e vermelho de 128.

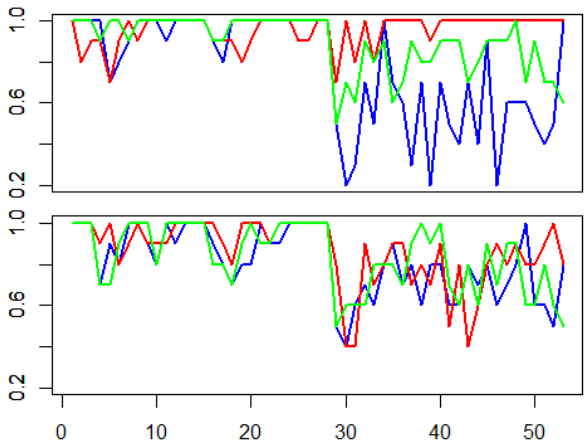

Figura 7. Acurácia encontrada variando-se o número de divisões do espaço. Acima os resultados do *grid* e abaixo do *quadtree*. A linha azul corresponde a um grid e a um tamanho de árvore respectivamente de 8 e 4, verde de 16 e 5 e vermelho de 32 e 6.

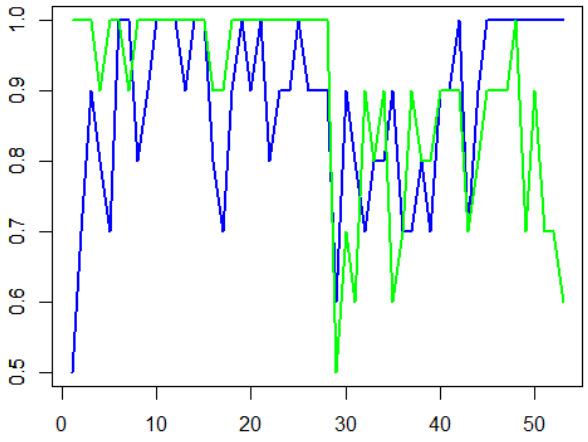

Figura 8. Acurácia encontrada utilizando e não utilizando a filtragem no método do *grid*. A linha em azul corresponde ao resultado sem a filtragem e em verde utilizando a filtragem.

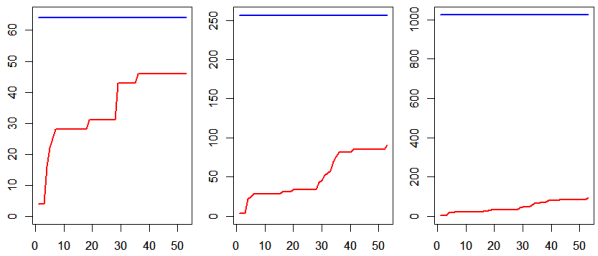

Figura 9. Quantidade de elementos utilizados pelos métodos variando-se o número de divisões do espaço. Em azul para o *grid* (correspondente ao tamanho máximo) e vermelho para o *quadtree*. Da esquerda para a direita os valores utilizados de grid e tamanho de árvore foram respectivamente iguais a 8 e 4, 16 e 5 e 32 e 6.

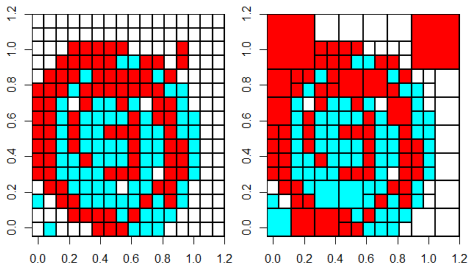

Figura 10. Diferença observada entre o método do *grid* sem a realização da filtragem (à esquerda) e do *quadtree* (à direita).

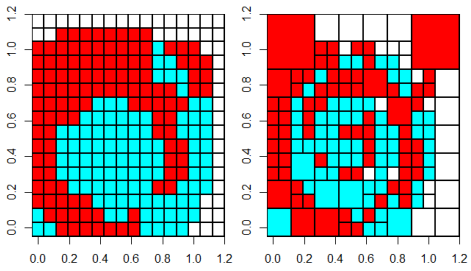

Figura 11. Diferença observada entre o método do *grid* com a realização da filtragem (à esquerda) e do *quadtree* (à direita).

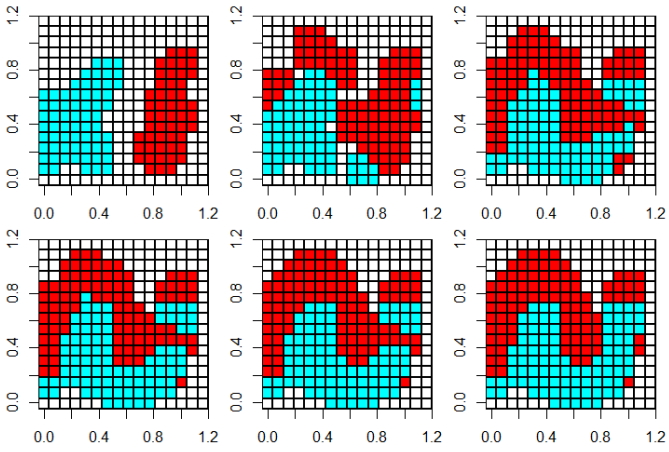

Figura 12. Variação do modelo do *grid* para a ocorrência de uma mudança de conceito. A ordem temporal é primeiro da esquerda para a direta e em seguida de cima para baixo.

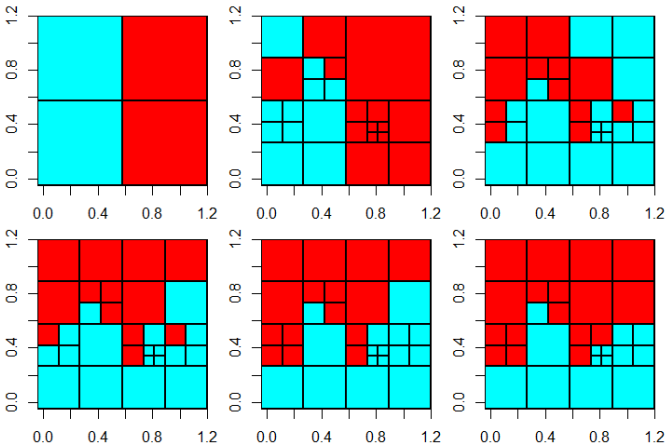

Figura 13. Variação do modelo do *quadtree* para a ocorrência de uma mudança de conceito. A ordem temporal é primeiro da esquerda para a direta e em seguida de cima para baixo.

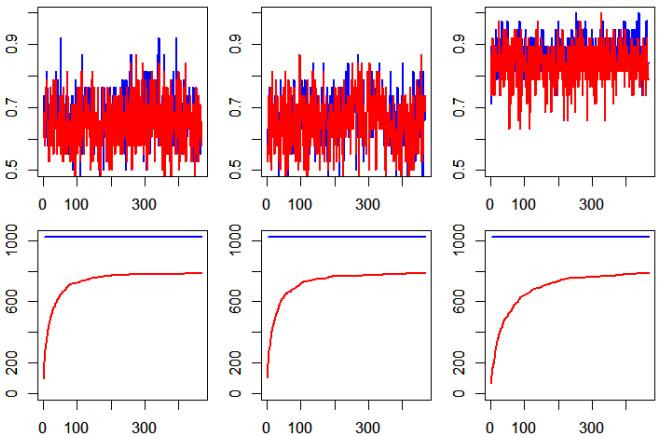

Figura 14. Acima está representada a acurácia encontrada e abaixo a quantidade de elementos utilizados pelos métodos na base de dados *fourclass*. Em azul para o *grid* e vermelho para o *quadtree*. Da direita para a esquerda estão representadas as variáveis 1 e 2, 1 e 3, 2 e 3.

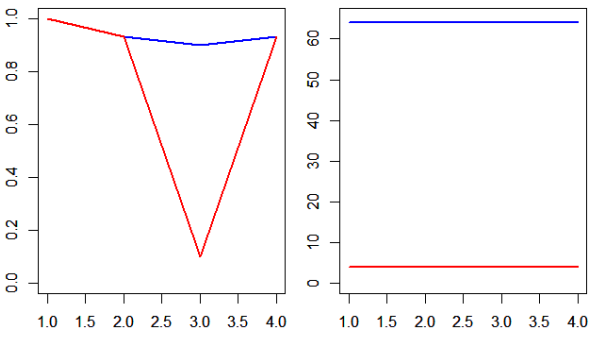

Figura 15. Acurácia encontrada (acima) e quantidade de elementos utilizados (abaixo) pelos métodos na base de dados *fourclass*. Em azul para o *grid* e vermelho para o *quadtree*.