Relatório de Trabalho Mineração de Dados

Renan Souza de Freitas* e Rodrigo A. de Freitas Vieira†
Programa de Pós-Graduação em Sistemas de Informação
Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo (USP)
*renan.freitas@usp.br
†rodrigo.vieira@usp.br

I. Introdução

Este trabalho tem como objetivo demonstrar o processo de implementação, testes e análises de resultados de algoritmos de Mineração de Dados sobre as tarefas de Agrupamento, Classificação e Associação.

O relatório é organizado da seguinte forma: A Seção II é apresentado o ambiente de programação na qual os algoritmos foram implementados e testados; na Seção III as bases de dados estudadas são apresentadas; na Seção IV descreve-se a implementação dos algoritmos implementados pelo grupo; A Seção V decorre sobre os experimentos da tarefa de classificação; A Seção VI expõe os experimentos da tarefa de agrupamento; A Seção VII trata dos experimentos sobre regras de associação.

II. Ambiente de programação

Dois ambiente de programação foram escolhidos para o desenvolvimento do trabalho, o MATLAB (Matrix Laboratory) R2015a e o Python 2.7.13. O grupo decidiu implementar os algoritmos usados para a tarefa de classificação e agrupamento em MATLAB e utilizar códigos de terceiros em Python para a tarefa de regras de associação.

Para a tarefa de classificação foi implementado pelo grupo algumas variações de Redes Neurais Artificias Multilayer Perceptron e o método de ensemble AdaBoost.

Para o Agrupamento foi implementado o algoritmo Self Organizing Map e a U-Matrix para visualização.

Para a tarefa de Regras de Associação usou-se de códigos disponíveis online para o método Apriori. O código foi desenvolvido por asaini, zkan e arimbr e está disponível online pelo github [1].

Nenhuma biblioteca foi utilizada para realizar as tarefas de mineração.

III. Bases de Dados

A. Classificação

Para todas bases de dados, foi feita a separação dos dados das instâncias e suas classes. Além disso, todas as bases de dados foram normalizadas por meio de Z-score.

[1]https://github.com/asaini/Apriori

1) Spambase: A base de dados Spambase abrange instâncias de e-mails classificados como spam e e-mails comuns, com 4601 ao todo. Possui 57 atributos originalmente, todos contínuos. No presente trabalho, houve utilização de Análise de Componentes Principais (*Principal Component Analysis*, PCA) para reduzir a dimensionalidade das instâncias do conjunto de dados, visando melhorar a performance dos algoritmos de classificação em termos de tempo. Foram utilizados os componentes que, em conjunto, explicavam variância relativa de, pelo menos, 95% da total. Assim, 48 dimensões foram mantidas no conjunto de dados. Na seção de Tarefa de classificação é descrito como o *k-fold cross-validation* é utilizado nesse conjunto como estratégia de teste.

2) Wine Quality: O artigo original foi seguido, mantendo as bases red e white wine separadas. Os autores do trabalho optaram por utilizar somente o conjunto de dados dos vinhos tinto, por ter um número menor de instâncias. Na seção de Tarefa de classificação é descrito como a estratégia de teste *Holdout* é utilizada.

B. Agrupamento

As bases de dados utilizadas para a tarefa de agrupamento foram a t4.8k[1] e a Path-based2:spiral[2]. Ambos são conjuntos artificiais de apenas 2 dimensões, criados para representar formatos difíceis de clusterizar com métodos usuais como o k-means.

O t4.8k, representado na figura 1(a), apresenta 6 formas geométricas interposto mas sem conexões, além de ruído por todo o espaço. O Path-based2:spiral, representado na figura 1(b) apresenta três grupos de dados organizados em caminhos espirais. Essa visualização e compreensão dos dados originais será importante para a análise dos resultados do SOM.

Os datasets foram obtidos em formato de texto pelo site Clustering Benchmark Datasets. Os dados foram carregados para o Matlab e armazenados em matrizes "X" para as entradas e "Y" para os rótulos, sendo salvo em arquivos ".mat". Assim como na tarefa de classificação, os dados de agrupamento são normalizados com o Z-score. Porém, os dados não são armazenados de forma normalizada, requerendo a normalização antes da execução do algoritmo.

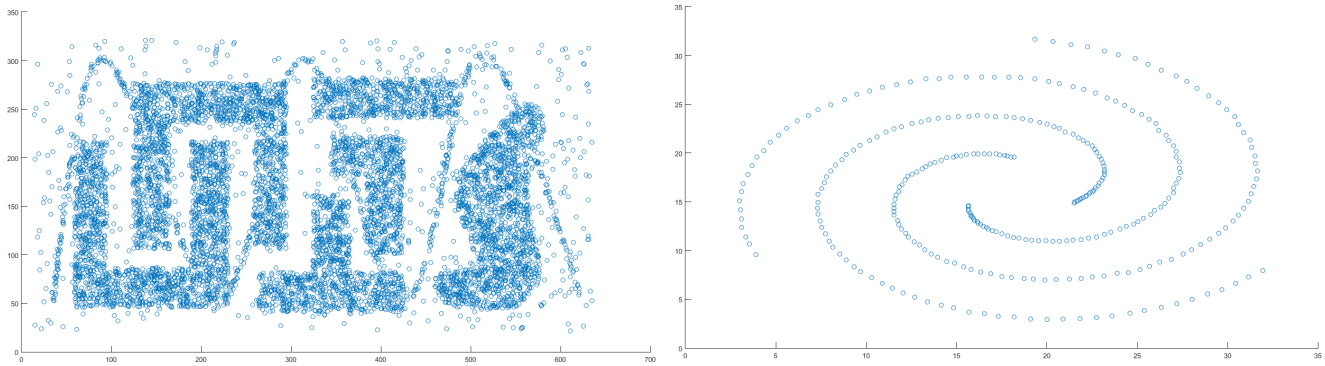

Figura 1: Conjunto de dados para a tarefa de agrupamento: (a) Base de dados t4.8k (b) Base de dados Path-based2:Spiral

C. Associação

Os conjuntos de dados selecionados para a tarefa de associação foram 1984 United States Congressional Voting Records Database e Contraceptive Method Choice.

O conjunto 1984 United States Congressional Voting Records Database apresenta os votos de cada representante do congresso americano sobre 16 importantes votações. O conjunto é composto por uma entrada para cada representante, classificado por sua filiação (republicano ou democrata) e com seus 16 votos como atributos. Esses votos podem apresentar os seguintes valores: y para posições favoráveis, n para posições contrárias e ? para posições indefinidas.

Para a utilização desses dados no algoritmo Apriori, acrescentou-se no inicio do valor de cada atributo o índice do atributo (com excessão da filiação), para que fossem identificadas as diferentes posições em cada uma das 16 votações. Alguns exemplos de entrada seriam:

Transaction 1: $republican, 2n, 3y, 4n, 5y, 6y, 7y, 8n, 9n, 10n, 11y, 12?, 13y, 14y, 15y, 16n, 17y$

Transaction 2: $republican, 2n, 3y, 4n, 5y, 6y, 7y, 8n, 9n, 10n, 11n, 12n, 13y, 14y, 15y, 16n, 17?$

Transaction 3: $democrat, 2?, 3y, 4y, 5?, 6y, 7y, 8n, 9n, 10n, 11n, 12y, 13n, 14y, 15y, 16n, 17n$

O conjunto Contraceptive Method Choice apresenta uma pesquisa realizada em 1987 na Indonésia com mulheres casadas que não estavam grávidas ou não sabiam se estavam na época da entrevista.

Os dados apresentam 10 atributos, dentre eles: 4 categóricos (Wife's education, Husband's education, Husband's occupation e Standard-of-living index), 3 binários (Wife's religion, Wife's now working? e Media exposure), 2 numéricos (Wife's age e Number of children ever born), e um atributo de classe identificando o método de contraceptivo utilizado, podendo admitir três valores distintos.

Para a geração de regras de associação com esses dados, os valores numéricos tiveram de ser discretizados. Wife's age foi dividido em 4 intervalos: até 20 anos; entre 20 e 30 anos; entre 30 e 40 anos; e maiores de 40 anos. Da mesma forma, o atributo Number of children ever born foi discretizado em 4 intervalos: nenhum filho, até 4 filhos, até 8 filhos e mais do que 9 filhos.

Aos outros atributos foi adicionado aos valores o nome do atributo referente para transformá-lo em itens distinto. Alguns exemplos de entrada seriam:

Transaction1: $[20 < age <= 30], WifeEduc = 2,$
$HusbEduc = 3, [0 < children <= 4], Religion = 1,$
$WifeWorking = 1, HusbandOccupation = 2,$
$Standard - of - living = 3,$
$MediaExposure = 0, Contraceptive = 1$

IV. Implementação

A. Self Organizing Maps

O primeiro passo para a implementação da Rede SOM foi a definição de sua estrutura. O tipo da lattice selecionado foi a quadrática pela sua simplicidade; já a função de vizinhança implementada foi a gaussiana; enquanto o número de neurônios na camada de entrada é definido pelo conjunto de dados, o número de neurônios na camada de saída e a dimensão são definidos por um parâmetro de duas posições indicando o tamanho do mapa.

A criação da rede é realizada por uma função SOM parametrizada com os seguintes valores:

- dim - dimensão do mapa de saída, definido como um vetor de duas posições
- $alphaIni$ - Taxa de aprendizado inicial
- $radiusIni$ - Raio de vizinhança inicial
- $lambda$ - Taxa de decaimento da taxa de aprendizado
- tau - Taxa de decaimento do raio
- $fDist$ - Métrica de distancia, podendo assumir os valores 'e' para distância euclidiana ou 'm' para manhattan
- $itMax$ - Número máximo de épocas de execução

O algortimo então, se desenvolve como descrito a seguir.

Os pesos dos neurônios da rede no espaço vetorial são representados por uma matriz W com Ns linhas (número total de neurnônios na saída) e D colunas (dimensão dos dados no conjunto de entrada). A inicialização dessa matriz é realizada randomicamente dentro do intervalo de valores mínimo e máximo do conjunto de entrada.

Para organização dos neurônios no grid e controle da vizinhança topológica, uma estrutura auxiliar chamada

$neuronsGrid$ foi criada. Essa estrutura é uma matrix com uma entrada para cada neurônio da camada de saída (Ns linhas) e com duas colunas representando as posições i e j no grid de saída.

Com os valores iniciais dos neurônios definidos, o processo iterativo do SOM se inicia.

A cada época as instâncias do conjunto de entrada são permutadas fazendo com que a apresentação para a rede padrão-a-padrão seja aleatória. Então, para cada instância p, calcula-se a distância do ponto para todos os neurônios W da rede. O neurônio com menor distância é selecionado como BMU.

Uma função $neighborhood$ é chamada para calcular θ, um vetor com a influência da atualização sobre cada neurônio a partir da vizinhança do BMU. A função de vizinhança é calcula com uma gaussiana

$$\theta = exp\left(- \frac{D}{2 * radius^2} \right) \quad (1)$$

onde D é a distância topológica de cada neurônio ao BMU, calculada como a distância manhattan com o $neuronsGrid$.

O processo de adaptação sináptica é realizado de acordo com a taxa de aprendizado α e o retorno da função de vizinhança θ para cada neurônio em W da seguinte forma

$$W_c = W_c + \alpha \times \theta_c \times (p - W_c), \quad c = 1, 2, ..., Ns. \quad (2)$$

Após a execução para todas as instâncias de entrada, ocorre o decaimento da taxa de aprendizado e do raio com

$$\alpha = \alpha_{inicial} \times \exp\left(\frac{-it}{\lambda}\right) \quad (3)$$

$$radius = radius_{inicial} \times \exp\left(\frac{-it}{\tau}\right) \quad (4)$$

Sendo it a época atual e λ e τ as taxas de decaimento da taxa de aprendizado e do raio, respectivamente.

O procedimento é realizado repetidamente até se atingir o número total de épocas.

A cada iteração também são armazenados quatro valores para análise do comportamento da rede durante o aprendizado. O Erro de Quantização apresenta uma medida de avaliação do SOM sobre a distância dos dados ao BMU referente. O Erro Topológico apresenta outra medida da qualidade da rede identificando distorções topológicas. O Delta Médio apresenta a média do valor de atualização absoluto sobre todos os neurônios da rede para cada dimensão vetorial. Já o Delta Máximo apresenta o valor absoluto máximo de variação sobre um neurônio da rede. Todas essas medidas servem para visualização da convergência do algoritmo e auxílio na parametrização.

O Erro de Quantização é calculado somando-se a distância de cada instância de entrada para o seu BMU e ao fim da época computa-se a média. Para o cálculo do Erro Topológico o segundo BMU de cada entrada é resgatado e verifica-se se o primeiro e o segundo BMU são vizinhos diretos, isto é, possuem distância topológica 1, caso não sejam soma-se 1 ao erro e ao fim da época computa-se a média. Para o Delta médio em cada instância calcula-se a média da atualização absoluta dos neurônios e ao fim da época computa-se a média de todos os dados. Já o Delta máximo armazena-se o valor absoluto máximo de atualização de um neurônio em cada época.

Para gerar a evolução do grid durante o processo de aprendizado são armazenados os valores de W no inicio e no fim da execução e após 33% e 66% das iterações.

A função então retorna os seguintes valores:

- W - Pesos finais dos neurônios da rede
- $WHist$ - Pesos dos neurônios no início, em 33% e 66% de iteracoes e no fim
- $QuantErrorHist$ - Erro de Quantização a cada iteração
- $TopolErrorHist$ - Erro Topologico a cada iteração
- $DeltaMeanHist$ - Delta Médio a cada iteração
- $DeltaMaxHist$ - Delta Máximo a cada iteração
- $TimeTraining$ - Tempo de Treinamento

Para visualização da rede gerada, três formas de visualização foram implementadas: o Grid dos neurônios no espaço vetorial, uma ilustação de quantização (influência) dos neurônios sobre os dados e a U-Matrix.

O Grid é gerado no espaço vetorial original dos dados, logo pode ser criado apenas para conjuntos com duas ou três dimensões. Nele podem ser vistos os pontos no espaço em que os neurônios foram organizados e suas respectivas vizinhanças topológica. São apresentadas duas imagens, uma apenas com o grid final para analise do posicionamento ao fim do algoritmo, e um composição de small multiples com a evolução do grid com os valores em $WHist$.

A ilustação da quantização também é representada no espaço vetorial do conjunto de entrada e conecta cada ponto ao BMU referente. A função implementada para gerar essa ilustração é parametrizada pela influência mínima desejada, no qual são plotados apenas os neurônios que sejam BMU de mais dados que o valor selecionado.

A Matriz-U foi a parte mais complexa de implementar do SOM, enquanto os outros algoritmos foram desenvolvidos de forma orgânica, foram necessárias três tentativas para a efetiva criação da U-matrix.

A primeira implementação, foi apenas para obter uma ideia geral do código e de como representar uma matriz de distâncias por cores. Foi criada uma matriz com as mesmas dimensões do mapa de saída, com valores das médias das distâncias de cada neurônio para seus vizinhos. Como esperado, nenhuma informação era obtida com essa visualização, porém, conseguimos produzir a matiz em forma visual colorida de acordo com as distâncias.

Antes de realizar a segunda tentativa de implementação, foram investigadas diferentes lógicas para a criação da matriz, porém, foram descartadas por não serem consistentes.

A segunda tentativa, mais próxima de uma solução plausível, foi uma tentativa de criar uma matriz como na figura 2. A matriz teria um valor para cada neurônio (média dos neurônios vizinhos) e entre eles a distância seria calculada. Os ”buracos” faltantes seria preenchidos com a média das distâncias ao redor. Essa solução apesar de não ser a melhor, funciona. Porém, ao calcular a média das distâncias verticais uma distração no código fazia com que as se calculasse as

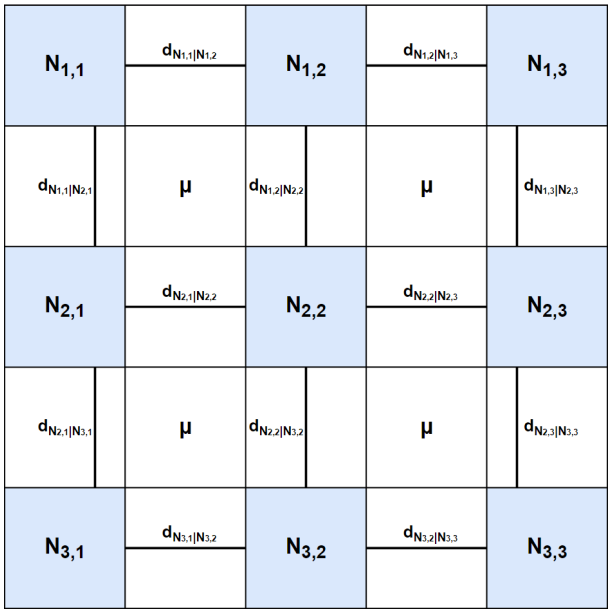

Figura 2: Segunda tentativa de implementação da Matriz-U

distâncias horizontais. Esse erro passou despercebido levando-nos a crer que o erro estava na lógica de criação da matriz.

Assim, desenvolveu-se a terceira e definitiva implementação. Essa abordagem é bastante semelhante à anterior e é baseada na tese de doutorado do José Alfredo Ferreira Costa. O formato do mapa é igual ao da segunda implementação apenas mudando a forma de calcular alguns valores. As distâncias verticais e horizontais se mantêm. Já os neurônios são representados pela média da posição de todos os neurônios que o envolvem, enquanto os "buracos"entre as distâncias (referidos como μ na figura 2) é a média dos neurônios diagonalmente opostos, calculados com a seguinte equação

$$\mu = \frac{1}{2}\left(\frac{\|N_{i,j} - N_{i+1,j+1}\|}{\sqrt{2}} + \frac{\|N_{i,j+1} - N_{i+1,j}\|}{\sqrt{2}}\right) \quad (5)$$

onde i e j são os indices do neurônio no grid e $N_{i,j}$ refere-se ao neurônio superior esquerdo ao μ em questão.

Essa implementação possibilitou a identificação e correção do erro na implementação anterior. Comparando-se as matrizes resultantes das duas implementações, percebe-se que esta última produz uma visualização mais informativa, facilitando a identificação dos grupos.

Por fim, para a realização dos testes o código gera um diretório com os resultados incluindo os gráficos de atualização média dos pesos, atualização máxima dos pesos, erro de quantização e erro topológico; as visualizações do conjunto de dados origianl, o Grid final, a evolução do Grid, a quantização com influencia mínima de 0 e 5 e a Matiz-U com vizualização 2D e 3D; um arquivo de texto com o log do teste registrando as informações sobre a base de dados, os parâmetros da rede e os resultados; e dois arquivos do .mat com os parâmetros e os resultados do teste.

B. Comitê de máquinas

Conforme a especificação, os componentes do comitê de máquinas deveriam ser Redes Neurais Artificiais (RNA) Multilayer Perceptron (MLP). Foi realizada a implementação tanto dos componentes do comitê, a MLP, como também o comitê em si, utilizando-se do algoritmo Adaboost.

A implementação de uma rede MLP, classe de redes de arquitetura *feed-forward*, possui uma gama de decisões que merecem atenção como a quantidade de camadas ocultas, quantidade de neurônios nas camadas ocultas, funções de ativação, estratégia de treinamento e abordagem da valoração da taxa de aprendizado. Durante a implementação da MLP, um dos focos foi facilitar a parametrização do treinamento a fim de possibilitar a criação de diferentes casos de teste e avaliação. Nos casos onde mais de uma abordagem foi realizada, parâmetros correspondentes foram criados. A seguir, as decisões tomadas durante a implementação.

a) Quantidade de camadas ocultas: Feita uma implementação preliminar de uma rede MLP com suporte a número variável de camadas ocultas, os autores optaram pela abordagem mais simples, criando implementações de redes MLP com uma única camada. A justificativa consiste no fato de que redes com mais camadas ocultas possuem um custo computacional mais elevado para seu treinamento. Além disso, redes com uma camada oculta são o suficiente para a grande maioria das aplicações.

b) Quantidade de neurônios nas camadas ocultas: A implementação suporta um número variável de neurônios nas camadas ocultas.

c) Função de ativação: A função de ativação usada tanto na camada oculta quanto na camada de saída foi a função logística.

d) Estratégia de treinamento: Os autores optaram pela implementação da MLP com treinamento em lote.

e) Taxa de aprendizado: O tratamento dado durante o treinamento da rede MLP à taxa de aprendizado foi abordado de algumas maneiras durante a implementação, todas considerando o parâmetro α_0, a taxa de aprendizado inicial:

- Taxa fixa
  α_0 é mantida como taxa de aprendizado ao longo de todo o treinamento;
- Taxa com decaimento
  a taxa de aprendizado α, inicializada com α_0, tem seu valor reduzido pela metade a cada 5 épocas;
- Taxa adaptativa
  a taxa de aprendizado α, inicializada com α_0, pode ter seu valor incrementado ou reduzido de acordo com o comportamento da medida de erro da época anterior em comparação com a atual, considerando o conjunto de validação, conforme o breve algoritmo 1.
- Taxa calculada por bisseção
  nessa abordagem, nenhum valor inicial necessita ser fornecido, de modo que α é calculado automaticamente por uma implementação do método de busca linear definido pelo algoritmo de bisseção

f) Gradiente: Foram utilizados gradiente descendente e gradiente conjugado.

Algorithm 1 Taxa de aprendizado adaptativa

if $MSE_{train}(i-1) < MSE_{train}(i)$ **then**
  $A = A + \delta_A$
  $B = B + \delta_B$
  $\alpha = 1.1 * \alpha$
  $i = i + 1$
else
  $\alpha = .9 * \alpha$
end if

g) Término do treinamento: o término do treinamento da rede MLP foi tratado por meio da combinação das seguintes abordagens:

- Limite de épocas
  A implementação aceita um valor máximo de épocas no treinamento;
- Limiar de métrica de erro
  Caso o erro da rede sobre o conjunto de treinamento esteja abaixo de um limiar tido como aceitável, a implementação interrompe o treinamento;
- *Early Stoppping*
  Dado que depois de certo número de épocas, a rede tende a se sobreajustar ao conjunto de treinamento, é interessante evitar a perda de generalização interrompendo o treinamento ao detectar-se o sobreajuste. A implementação considera que caso o erro sobre o conjunto de validação comporte-se como uma função monotonicamente crescente nas últimas 20 épocas, existe sobreajuste. Nesse caso, a rede tem seu treinamento interrompido e os pesos antes dessas épocas são retornados. A figura 3 exemplifica um caso onde a parada prematura poderia ter sido útil.

h) Neurônios na camada de saída da rede: Para cada conjunto de dados, foi feita uma abordagem diferente.

- Spambase
  Para o problema de classificação binária da base de dados de spam, a saída da rede conta com apenas um neurônio. Para esse caso, é necessário definir um limiar de decisão em $\left[0, 1\right]$ para atribuir uma instância a uma classe ou outra. Esse limiar foi definido arbitrariamente como 0.5, mas poderia ter sido feito uso de análise da curva ROC, alterando-se gradualmente o limiar de decisão para encontrar aquele de melhor qualidade. A implementação do processo descrito de análise de curva ROC foi feito, mas não utilizado nos experimentos realizados.
- Qualidade de vinho tinto
  Para o problema de classificação multiclasse, dado o conjunto *C* de classes, a camada de saída conta com $|C|$ neurônios. Um neurônio *i* nesse camada corresponde à confiança da rede de que uma dada instância pertence à classe $c_i \in C$.

A função de treinamento da rede MLP chama-se *MLPtreina* e retorna os pesos da camada oculta, os pesos da camada de saída, o erro ao longo das épocas no conjunto de treinamento, o erro ao longo das épocas no conjunto de validação e a quantidade de épocas executadas.

Além disso, como forma de tentar melhorar a performance de uma dada rede MLP sobre o conjunto de dados multiclasse de vinho tinto, algumas implementações alternativas de MLP foram feitas. São descritas a seguir, apesar de não ter sido obtido com elas o resultado esperado.

- *Classificador baseado em MLP para classes ordenadas*
  Implementação de classificador baseada no artigo de [3]. classificador, para cada $c_i \in C$, o conjunto de classes, cria $|C| - 1$ redes MLP para o problema de classificação binário das instâncias do conjunto de dados pertencerem a $c_j \in C \mid j > i$. Em outras palavras, as saídas das $|C|-1$ redes no classificador, fornecem a probabilidade $P(y > c_i)$. A vantagem teórica dessa abordagem é utilizar-se da ordem existente nos rótulos de um conjunto como o de vinho tinto.
- *MLP para regressão*
  Implementação que converte o problema de classificação em um problema de regressão antes de submetê-lo à rede.

O algoritmo Adaboost foi escolhido como base para a implementação do ensemble. A existência de um conjunto de dados com múltiplas classes, implicou na implementação do Adaboost.M2, extensão do Adaboost criada pelos mesmos autores do algoritmo original especializada em problemas de classificação desse tipo.O algoritmo Adaboost.M2 é detalhadamente descrito em [4]. Nessa variação do Adaboost, o classificador deve ser modelado de modo que sua saída indique sua confiança na pertinência de uma dada instâncias às classes possíveis. Dado que no presente trabalho a técnica de classificação consiste em redes MLP, a camada de saída das mesmas foi definida para que possuísse quantidade de neurônios igual ao número de classes do conjunto de dados.

Inicialmente, uma primeira dificuldade foi encontrada para entender o próprio Adaboost.M2, posto que alguns termos não foram considerados exatamente claros pelos autores. Dados N o número de amostras no conjunto de dados e nc o número de classes do conjunto, pode-se destacar o termo D, matriz $(N, nc-1)$ que representa a confiança do classificador nas classes erradas. A matriz D é manipulada ao longo das rodadas de execução do Adaboost.M2, influenciando as reamostragens que são executadas antes de cada nova rodada. Houve dificuldade em realizar essa manipulação em um primeiro momento. Na primeira implementação do algoritmo, o conjunto de dados utilizado na rodada anterior para treinamento era reamostrado para ser utilizado na rodada, considerando uma distribuição de probabilidades gerada a partir de D, de modo que eram realizadas reamostragens sobre um conjunto já reamostrado. Percebido o erro, o código foi modificado para que reamostrasse a partir do conjunto de dados original. A reamostragem, feita com reposição, é realizada criando-se uma distribuição de probabilidade P onde para uma instância i do conjunto de treinamento a probabilidade P_i é calculada com base na equação, na rodada t do Adaboost.M2

$$P_i(t) = \frac{\sum\limits_{y \neq y_i} D_i(y)}{Z} \tag{6}$$

onde Z é uma constante definida afim de que os valores em P sejam mantidos em $[0, 1]$.

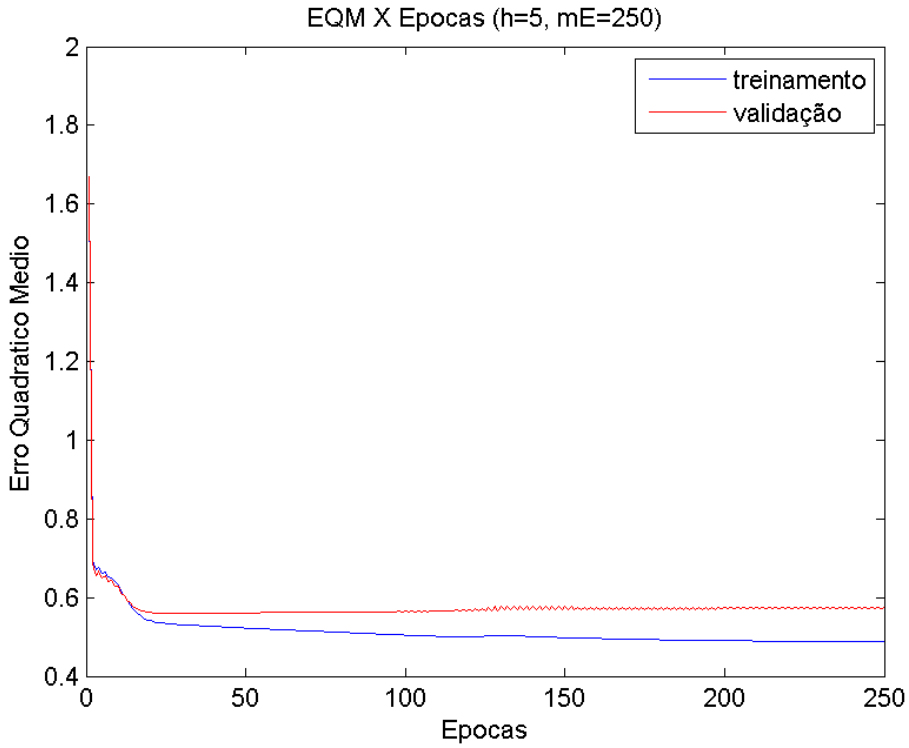

Figura 3: Evolução do erro quadrático médio (EQM) sobre o conjunto de treinamento e sobre o de validação durante a execução de uma MLP sobre o conjunto de dados de vinho tinto, com $h = 5$ neurônios na camada oculta e $m = 250$ épocas, sem parada prematura. Pode-se observar um ponto de inflexão da na curva do EQM sobre o conjunto de validação enquanto a curva sobre o de treinamento continua em queda, indicando sobreajuste do classificador.

No algoritmo Adaboost a reamostragem tem uma elevada importância, pois, a grosso modo, promove a maior presença no conjunto de treinamento de uma dada rodada das amostras que na rodada anterior foram erroneamente classificadas ou que possuem valores de confiança mais altos que outras para as classes diferentes do seu rótulo. Nesse cenário, a distribuição de instâncias entre classes é alterada, sendo possível que, por exemplo, uma classe majoritária deixe de sê-lo em detrimento de outra em alguma rodada. Nesse exemplo, tal comportamento poderia ser explicado por confiança demasiado alta nas classes incorretas por parte dos classificadores das rodadas anteriores. Contudo, as classes do conjunto de dados de vinhos são altamente desbalanceadas. Mesmo muitas execuções do algoritmo, apesar de dobrarem ou triplicarem o número de instâncias em classes menores, não implicam em grandes alterações na distribuição de instâncias entre classes no conjunto de dados, pois as classes com mais instâncias são algumas ordens de grandeza maiores que as demais.

Consequentemente, algumas estratégias de reamostragem foram tentadas e incorporadas ao adaboost:

- Reamostragem com probabilidades ponderadas
  probabilidades ponderadas em função do tamanho da classe em relação ao conjunto de dados, sendo a ponderação inversa: probabilidades de instâncias de classes maiores são ponderadas por pesos menores; contudo, os resultados não foram significativamente melhores, de modo que não serão detalhados;
- Reamostragem mantendo a proporção de classes
  forçou-se manter a proporção original das classes no conjunto de dados, afim de confirmar que as modificações na distribuição das amostras consequentes a reamostragem ”padrão”do Adaboost são favoráveis à performance do ensemble; por questão de espaço os resultados não serão apresentados, mas pode-se perceber que o uso dessa reamostragem tornou o ensemble pior que as redes MLP, de modo que os resultados não serão detalhados;

Apesar de tentadas essas estratégias alternativas de reamostragem, os autores optaram por manter a estratégia de reamostragem inicial.

Notavelmente, é interessante promover a diversidade entre os componentes em um ensemble. A reamostragem é uma estratégia possível, mas não a única. Uma outra abordagem é a promoção da diversidade estrutural. No caso de redes MLP como componentes desse trabalho, optou-se por a cada rodada do Adaboost.M2 alterar o número de neurônios na camada oculta. Desse modo, dada uma quantidade h_0 de neurônios inicial, a cada rodada uma nova quantidade é definida em na faixa de valores $\left[h_0 - \delta_h, h_0 + \delta_h\right]$, seguindo uma distribuição uniforme de probabilidades para definir-se δ_h. Na implementação, o valor máximo para δ_h é 5.

Ademais, também em relação ao Adaboost houve modificações com a esperança de obtenção de melhorias. De acordo com um algoritmo guloso de seleção de componentes, terminadas todas as rodadas de execução do Adaboost os

componentes geradas foram submetidos a esse algoritmo e os componentes selecionados, utilizados como um segundo ensemble para classificar o conjunto de dados de teste. A comparação entre o ensemble original e o de componentes selecionados foi realizada através da acurácia. Contudo, o desempenho do ensemble com componentes selecionados não apresentou-se satisfatório, de modo que que a abordagem foi descartada.

Em linhas gerais, o algoritmo consistia em ordenar cada componente em função de seu erro quadrático médio sobre o seu conjunto de treinamento. Feita a ordenação, definia-se um ensemble com o melhor desses componentes. Em seguida, cada componente da lista era adicionado ao ensemble e caso o desempenho do novo ensemble em termos de erro quadrático médio fosse melhor que o desempenho anterior, o componente adicionado era mantido. O processo é repetido até que toda a lista ordenada seja percorrida.

zUm algoritmo similar foi implementado para seleção de atributos, mas foi descartado pelo seu custo computacional ter sido considerado impeditivo.

V. Tarefa de Classificação

Nos experimentos realizados com os dois conjuntos de dados, houve exploração da parametrização de redes MLP visando encontrar redes com bom desempenho para compará-las com o ensemble. Os componentes do ensemble, por sua vez, possuem parametrização similar a das redes MLP tidas com bom desempenho. A abordagem utilizada pode ser enumerada nas seguintes etapas:

1) *Estratégia de teste*
   Define-se uma abordagem para teste, de acordo com o conjunto de dados, sendo disponíveis *holdout* e *k-fold cross-validation*;
2) *Grid de parâmetros nas redes MLP*
   Inicialmente foram executados experimentos para definir uma região interessante em termos de acurácia para os parâmetros número de neurônios na camada oculta e número máximo de épocas;
3) *Ensemble*
   Nessa etapa, valores promissores para neurônios na camada oculta obtidos na etapa 1 são utilizados para a execução do ensemble;
4) *Comparação entre melhores*
   A melhor parametrização de MLP encontrada é comparada ao ensemble em termos de acurácia, matriz de confusão e medidas derivadas dela.

A. Conjunto de dados sobre vinho tinto

1) ***Estratégia de teste - Holdout****:* A estratégia *holdout* foi escolhida pelo tamanho do conjunto de dados ter sido considerado pequenos demais para o uso de *k-fold cross-validation*. Todas as etapas descritas foram executadas utilizando o holdout como estratégia de teste. Após a normalização, o conjunto de dados original é reamostrado sem reposição de modo que $\frac{2}{3}$ dos dados componham o conjunto de treinamento e o restante o conjunto de teste. O conjunto de dados de treinamento é dividido novamente de modo que $\frac{2}{3}$ componham o conjunto de treinamento final e as demais instâncias componham o conjunto de validação, utilizado, por exemplo, para a parada prematura. Dada essa separação entre os dados, os conjuntos de dados resultantes são persistidos em um arquivo em disco de modo que execuções de algoritmos diferentes ou algoritmos com parametrizações diferentes utilizem a mesma divisão de dados. Seguindo o processo de holdout descrito, foram criadas 5 divisões sendo que cada uma é submetida 20 vezes a cada parametrização diferente ou a cada algoritmo diferente.

Nesse conjunto de dados, uma etapa foi adicionada a sequência descrita anteriormente, entre as etapas *Grid de parâmetros* e *Ensemble*:

- *Variação de abordagens de taxas de aprendizado e gradiente*
  Os melhores resultados do passo anterior foram utilizados para testar as diferentes abordagens de taxas de aprendizado e de gradiente.

Essa adição é uma consequência da investigação realizada durante a busca de melhorias dos classificadores geradaos para esse conjunto de dados.

2) ***Grid de parâmetros nas redes MLP****:* Nessa etapa de experimentos, foi realizado o desenho fatorial dos parâmetros quantidade de neurônios na camada oculta e número máximo de épocas em uma rede MLP com gradiente descendente e taxa de aprendizado calculada pelo método de bisseção com parada prematura. Nesse experimento, a quantidade de neurônios *h* é definida como $h \in \{3, 5, 7, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ e o máximo de épocas *m* é definido como $m \in \{200, 500, 1000\}$. Utilizando a estratégia holdout para teste como mencionado acima, pode-se observar na figura 4 que o valor para *h* entre cerca de 5 e 20 neurônios é o que atinge melhor desempenho em termos de acurácia da rede. Além disso, pode-se observar na figura 5 que nessa faixa de valores o número de épocas executadas difere de *m* pois a execução da rede com parada prematura permitiu a mesma que detecta-se o sobreajuste e interrompesse a própria execução afim de maximizar sua generalização. Como consequência, pode-se inferir que para a rede MLP não é interessante a execução com $m > 1000$ visto que com parada prematura a execução seria interrompida antes do máximo e sem parada prematura ocorreria sobreajuste, nessa faixa de valores para *h*. Dessa etapa, obtém-se que valores interessantes para *h* são $\{h \mid 5 \leq h \leq 10\}$ e para *m* são $\{m \mid m \leq 1000\}$.

3) ***Variação de abordagens de taxas de aprendizado e gradiente****:* Nessa etapa foram realizados experimentos alterando-se a estratégia para a taxa de aprendizado e para o gradiente. Todas as implementações descritas na subseção sobre implementação foram testadas, seguindo a mesma estratégia de holdout previamente descrita. Nas implementações que necessitam obrigatoriamente de uma taxa de aprendizado (taxa coom decaimento e taxa fixa), foram fornecidos os valores $\alpha_0 \in \{0.01, 0.1, 0.2, 0.5, 0.9\}$. A quantidade $h = 20$ de neurônios na camada oculta e o número máximo de épocas $m = 1000$ foram definidos a partir dos resultados da etapa anterior, onde percebeu-se que essa parametrização atingia os melhores valores de acurácia.

Podem ser observados os resultados de cada parametrização em termos de acurácia na figura 6. Através da análise do

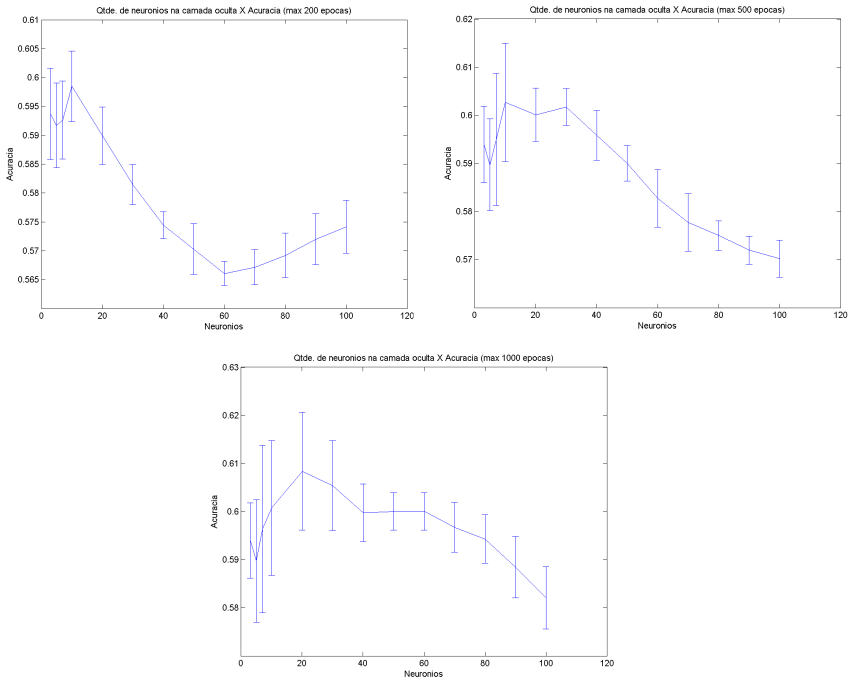

Figura 4: Evolução da acurácia média, com desvio padrão, da MLP conforme aumenta-se o número de neurônios na camada oculta. (a) Acurácia da rede com 200 épocas como máximo. (b) Acurácia da rede com 500 épocas como máximo. (c) Acurácia da rede com 1000 épocas como máximo.

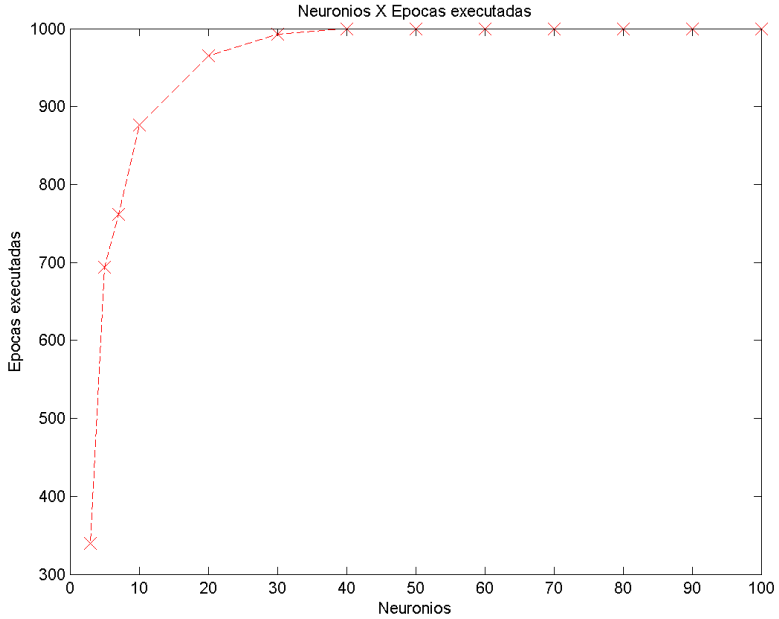

Figura 5: Evolução do número de épocas executadas em função da quantidade de neurônios na camada oculta quando o número máximo de épocas $m = 1000$. As épocas executadas diferem de m como consequência da execução com parada prematura.

gráfico, pode-se notar que algumas parametrizações se destacam com acurácia próxima de 0.6: taxa de aprendizado fixa com $\alpha_0 = 0.01$, método de bisseção com gradiente descendente, método de bisseção com gradiente conjugado e taxa de aprendizado adaptativa.

Como todas as execuções foram feitas com parada prematura, também é interessante observar a figura 7, onde são mostradas por parametrização o número médio de épocas executadas. Na figura 7, algumas das parametrizações apresentam apenas a linha vermelha que representa média do valor, pois não houve variação no número de épocas executadas, ou seja, nessas parametrizações o número de épocas executadas foi o mesmo que o máximo definido. Dentres as parametrizações que são destaque na figura 6 por sua acurácia, pode ser observado que entre o uso do método de bisseção com gradiente descendente e o uso do método de bisseção com gradiente conjugado, esse último possui um número médio de épocas executadas menor: cerca de 400 épocas contra 1000 do primeiro. Pode-se atribuir essa diferença ao uso do gradiente conjugado que, usando informações de segunda ordem sobre o gradiente, promove uma exploração mais veloz da superfície de erro pela rede em direção a um mínimo. Também deve ser observado que as redes com valores de $\alpha_0 = 0.01$, que possuem acurácia mais elevada que outras que possuem valor parametrizável para α_0. Uma explicação possível é que o valor inicial da taxa de aprendizado é tão baixo nesses casos que promove uma exploração bastante suave da superfície de erro, de modo que ao ser encontrado um mínimo dificilmente a rede sairá dele, visto que o valor da taxa de aprendizado indica o tamanho do passo a ser dado a cada época. Contudo, como já foi observado, o conjunto de dados de vinho tinto é altamente desbalanceado. Mesmo uma abordagem simples como a fixação da taxa de aprendizado é capaz de aproximar-se de abordagens mais rebuscadas em termos de acurácia pois basta que classifique corretamente as classes com maior número de instâncias.

Dessa forma, faz-se interesssante analisar as matrizes de confusão das redes com esses parâmetros. A figura 8 apresenta as matrizes de confusão. Pode-se observar que as redes que partem de ou permanecem em $\alpha = 0.01$, a taxa de aprendizado, classificam as instâncias nas classes com maior número de instâncias. As redes que utilizam de método de bisseção, por sua vez, apresentam mais predições corretas em classes com C_5 e C_2 ainda que em menor número nessa última.

Consequentemente, pode-se entender que as redes fazendo uso do método de bisseção apresentam melhor desempenho. Dentre essas, a rede com gradiente conjugado é a que melhor equilibra esse desempenho com o número em média baixo de épocas que executa.

4) ***Ensemble****:* Da mesma forma que com as redes MLP, o ensemble foi testado utilizando-se de holdout, como descrito previamente. Os componentes do ensemble são redes MLP com uma única camada oculta, taxa de aprendizado calculada pelo método de bisseção, gradiente conjugado e sem parada prematura. Como descrito na implementação, cada uma das redes tem um número de neurônios na camada oculta variável, mas próximo de $h_0 = 20$. O número de componentes foi definido com 35 em um um primeiro momento. Foram feitas execuções com valores distintos para o parâmetro *m*, o máximo de épocas de cada componente. Nos testes, $m \in \{1000, 5000, 10000, 15000\}$.

Na figura 9 pode-se observar a matriz de confusão para uma execução do Adaboost.M2 nas configurações descritas acima. O comportamento observado nessa matriz pode ser estendido às demais execuções, por esse ser o caso médio. Nota-se que o ensemble foi capaz de classificar corretamente não só as instâncias de teste pertences às classes intermediárias, mas também algumas instâncias de vinhos considerados ruins e ótimos.

Além disso, pode-se observar na figura 10 o efeito do aumento do número máximo de épocas dos componentes na acurácia do ensemble. Notavelmente, o comportamento da acurácia aparenta ser crescente com o número de épocas máximo permitido aos componentes. Ao passo que o aumento do número de épocas contribui para o sobreajuste em redes MLP, no ensemble o poder de generalização aparenta aumentar. Uma possível explicação para isso é que mesmo que ocorra o sobreajuste de um componente sobre o conjunto de dados com o qual ele foi treinado, dois fatos contribuem para o comportamento percebido. Há o fato desse componente gerar uma hipótese distinta dos demais e também o fato de que no Adaboost.M2 os votos de cada componente são ponderados pelo seu desempenho.

5) ***Comparação entre melhores****:* Considerando-se o domínio do problema, classificação da qualidade de vinhos, é interessante não só identificar os vinhos de qualidade média, mas também os vinhos de qualidade superior e inferior. Assim, considerando a quantidade maior de classes com classificações corretas e a acurácia média maior, pode-se concluir que o ensemble com $m = 10000$ épocas máximas é o melhor classificador encontrado para esse conjunto de dados.

B. Conjunto de dados sobre spam em e-mail

1) ***Estratégia de teste - k-fold cross validation****:* As etapas descritas no início da seção fora executadas para o conjunto de dados de spam utilizando como estratégia de teste o k-fold cross-validation, com $k = 5$. Feita a normalização e a aplicação de PCA para redução de dimensionalidade, o conjunto de dados é divido em 5 folds de tamanhos iguais. Cada fold, por sua vez é reamostrado sem reposição para que seja feita uma divisão tal que $\frac{2}{3}$ dos dados pertençam ao conjunto de treinamento e os demais ao conjunto de teste. O conjunto de de treinamento ainda é dividido novamente, de modo que $\frac{2}{3}$ de suas instâncias estejam no conjunto de treinamento final, ao passo que as outras estarão no conjunto de dados de validação, útil para parada prematura, por exemplo.

2) ***Grid de parâmetros nas redes MLP****:* Durante essa etapa, realizou-se a combinação de alguns valores para os parâmetros *h* e *m*, os neurônios na camada oculta e o máximo de épocas, respectivamente. Enquanto para o parâmetro *h* os valores foram definidos como $5, 10, 20, 50, 100$, para o parâmetro *m* os valores foram $100, 200, 500$.

Na figura 11 é possível observar a evolução da acurácia média e desvio padrão para uma rede utilizando gradiente descendente e método de bisseção, para o cálculo da taxa

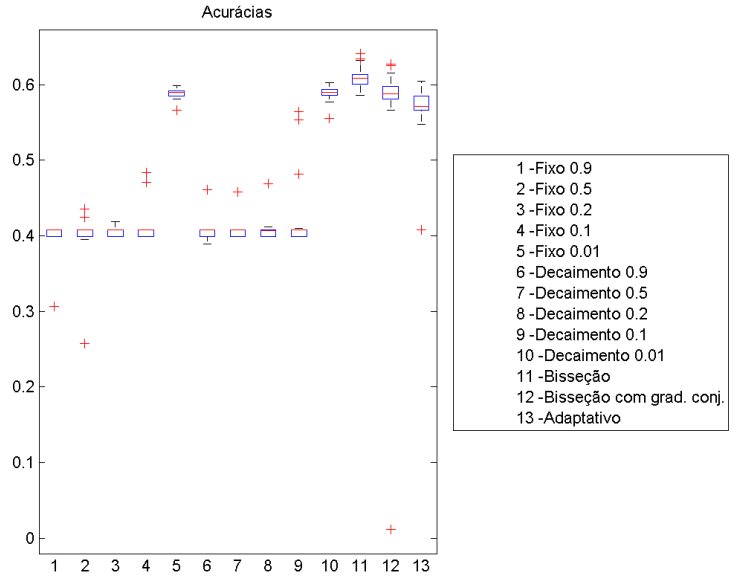

Figura 6: Acurácia média de redes MLP com diferentes abordagens para taxa de aprendizado e gradiente, para o dataset de vinho tinto. Todas as execuções foram feitas com $h = 20$ neurônios na camada oculta, $m = 1000$ como máximo de épocas e parada prematura. A legenda na figura indica, nos casos de taxa fixa e com decaimento, os valores de α_0.

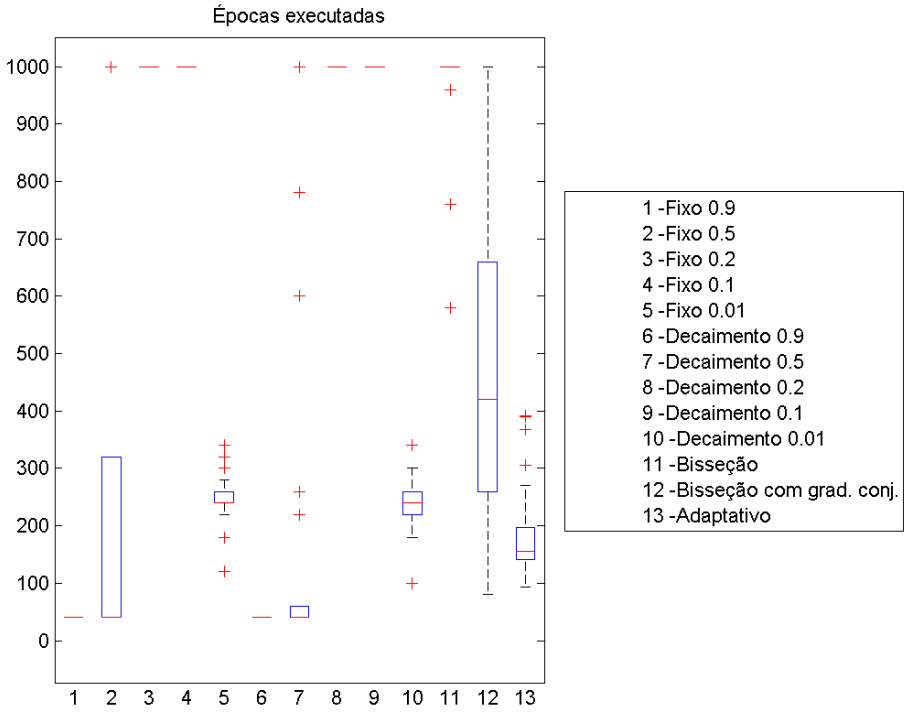

Figura 7: Quantidade média de épocas executadas em redes MLP com diferentes abordagens para taxa de aprendizado e gradiente, para o dataset de vinho tinto. Todas as execuções foram feitas com $h = 20$ neurônios na camada oculta, $m = 1000$ como máximo de épocas e parada prematura. A legenda na figura indica, nos casos de taxa fixa e com decaimento, os valores de α_0.

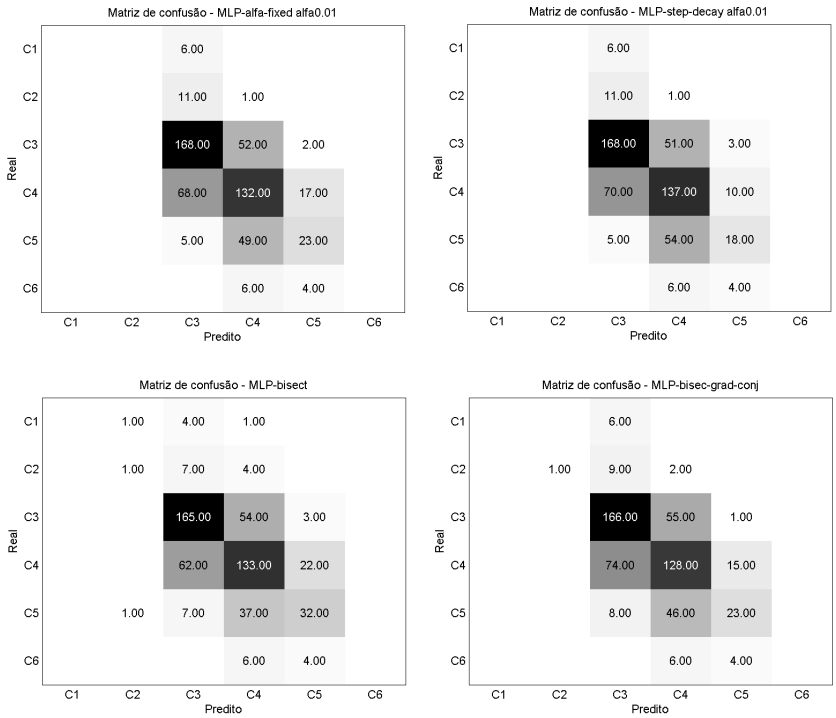

Figura 8: Matrizes de confusão de parametrizações de destaque, para o dataset de vinho tinto. Todas as execuções foram feitas com $h = 20$ neurônios na camada oculta, $m = 1000$ como máximo de épocas e parada prematura. (a) Taxa fixa em 0.01 (b) Taxa com decaimento a partir de 0.01 (c) Método de bisseção com gradiente descendente (d) Método de bisseção com gradiente conjugado

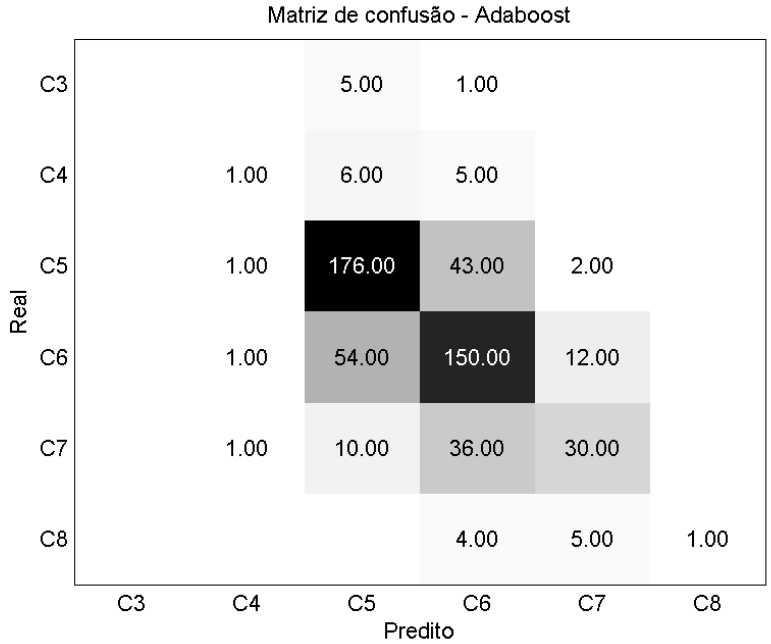

Figura 9: Matriz de confusão de um ensemble para o dataset de vinho tinto. Execuções feitas com 35 componentes e para cada um deles $h_0 = 20$ neurônios na camada oculta, $m = 5000$ como máximo de épocas e sem parada prematura.

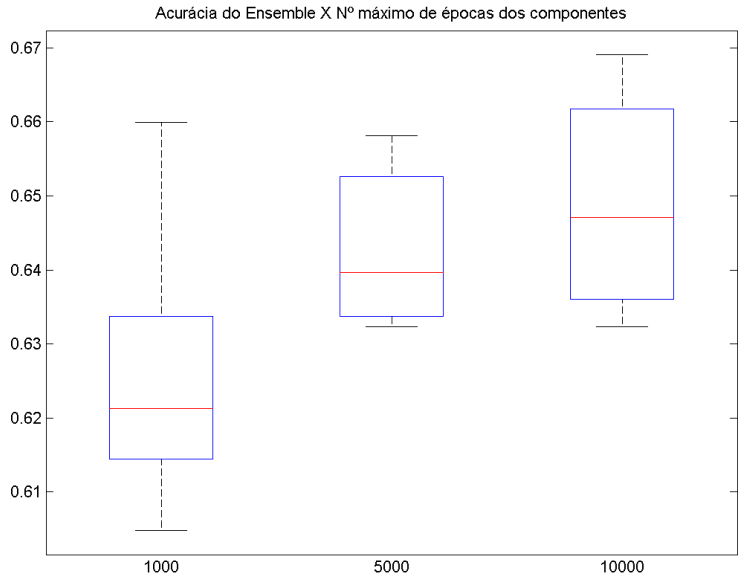

Figura 10: Acurácia dos ensembles testados com o dataset de vinho tinto. Execuções feitas com 35 componentes e para cada um deles $h_0 = 20$ neurônios na camada oculta, sem parada prematura e m, máximo de épocas, variável.

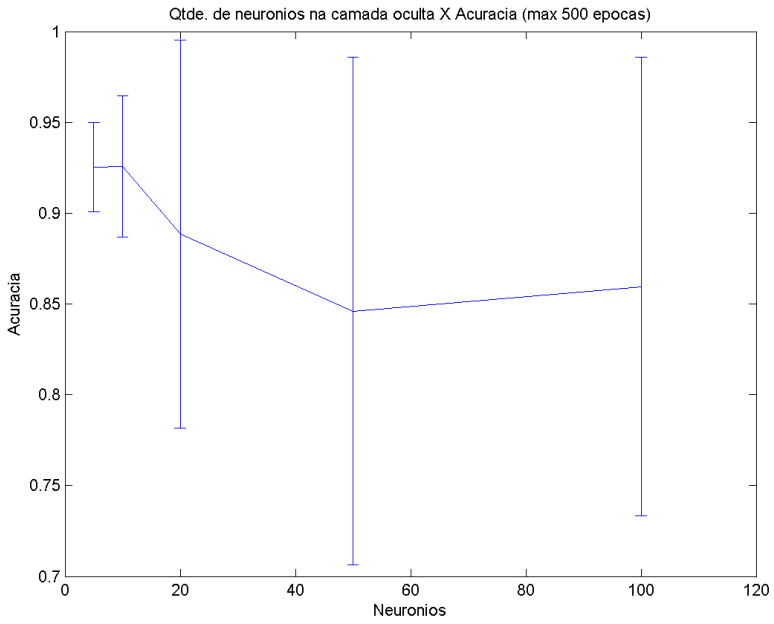

Figura 11: Evolução da acurácia média e desvio padrão para redes MLP aplicadas sobre o conjunto de dados de spam, utilizando bisseção e gradiente descendente.

de aprendizado. Na figura, a execução foi parametrizada com $m = 500$ épocas ao máximo, mas em demais parametrizações o comportamento observado foi praticamente o mesmo. Pode-se notar que para quantidades menores de neurônios na camada oculta, nominalmente *5* e *10* a rede atinge valores elevados de acurácia com desvio padrão baixo. Nas demais parametrizações, pode-se notar que não so o valor médio é mais baixo como também o desvio padrão é muito mais elevado. Contudo, não é interessante nesse caso analisar o desempenho das redes apenas em função de sua acurácia média. Especialmente no caso de um classificador aplicado a spam, é muito relevante atentar às classificações incorretas de e-mails comuns, ou seja, é importante observar medidas que indiquem o desempenho do classificador com foco em quantos e-mails ordinários são incorretamente classificados como spam.

Tabela I: Medidas da matriz de confusão para redes MLP aplicadas a spambase, conforme varia-se *h*, a quantidade de neurônios na camada oculta

h	5	10	20	50	100
TPR	0.8958	0.9018	0.8723	0.8605	0.8762
PPV	0.9127	0.9099	0.8696	0.8178	0.8245
FPR	0.0553	0.0587	0.1009	0.1636	0.1515
F-SCORE	0.9036	0.9055	0.8638	0.8251	0.841

A tabela I indica alguns valores calculados a partir da matriz de confusão para o problema de classificação binária de spam em e-mails. No cálculo das métricas e na análise as instâncias pertencentes à classe "spam"foram consideradas como exemplos positivos. Percebe-se que para $h \in \{5, 10\}$ são encontrados os melhores valores para as métricas apresentadas:

- *Revocação (TPR)*
  A medida indica o percentual de vezes em que um e-mail que é spam é classificado como tal. Notavelmente uma métrica importante para o contexto;
- *Precisão (PPV)*
  A medida indica o desempenho do classificador em relação aos acertos feitos quando as instâncias foram classificadas como pertencentes à classe positiva, no caso spam. É interessante nesse contexto que e-mails comuns não sejam classificados como spam, então essa medida deve ser observada juntamente à revocação;
- *Taxa de falsos positivos (TPR)*
  Nessa medida, pode-se verificar o comportamento do classificador em termos de classificar como spam os e-mails que não o são. No contexto, é altamente indesejável que essa seja uma medida com altos valores;
- *F-Score*
  Medida que combina TPR e PPV e que indica a capacidade do classificador em rotular corretamente as instâncias de spams disponíveis e também a capacidade de não rotular e-mails comuns como spams.

3) ***Ensemble****:* A estratégia de teste 5-fold cross-validation também foi utilizada com o ensemble. Da mesma forma que no conjunto de dados de vinhos, o ensemble é baseado no algoritmo Adaboost.M2 que, sendo uma generalização com suporte à classificação com múltiplas classes, é aplicável ao problema de classificação binária.

Tabela II: Comparação entre medidas da matriz de confusão para o ensemble e rede MLP aplicados a spambase

Classificador	MLP	Ensemble
TPR	0.9018	0.9195
PPV	0.9099	0.9435
FPR	0.0587	0.0350
F-SCORE	0.9055	0.9313

O ensemble foi executo com 35 componentes, redes MLP com uso do método de bisseção para o cálculo da taxa de aprendizado e gradiente conjugado, sem parada prematura. O número máximo de épocas para cada uma das redes foi definido com 1000. Da mesma forma que descrito na implementação e usado no conjunto de dados de vinhos, para adicionar diversidade ao ensemble cada rede possui número variável de neurônios em sua camada oculta, sendo que esse valor é uniformemente distruído no intervalo $\big[5, 15\big]$.

4) ***Comparação entre melhores****:* Pode-se observar na tabela II um comparativo entre o ensemble e a MLP da subseção anterior, com 10 neurônios na camada oculta. Notavelmente, o uso do ensemble trouxe melhoria em todas as métricas observadas.

VI. Tarefa de Agrupamento

Os experimentos de agrupamento assim como nos de classificação, estão divididos por base de dados. As principais análises sobre o comportamento do algoritmo foram realizadas com o dataset t4.8k, enquanto uma análise mais sucinta foi feita com o dataset path-based2 apenas para explorar o conjunto.

Os testes foram organizados da seguinte forma:

- t4.8k
  - Parametrização
  - Visualização
  - Análise do Aprendizado
  - Análise de Erros
  - Análise de Influência
- Path-based2
- Visão do grupo sobre o SOM

A. t4.8k

1) ***Parametrização****:* A parametrização começou de forma aleatória, testando-se diversos valores até chegar a uma configuração interessante. Os parâmetros dessa configuração eram raio inicial 30, taxa de aprendizado inicial 0.9, taxa de decaimento do raio de 32, taxa de decaimento da taxa de aprendizado de 32, com 300 iterações e dimensão de 20x20.

Essa parametrização inicial foi definida como base, e diferentes valores foram testados a partir dela. Decidimos manter a taxa de aprendizado inicial em 0.9 alterando somente seu decaimento, podendo ser mais rápido com uma taxa de decaimento maior ou mais lentamente caso contrário.

A avaliação da rede foi feita por meio da análise de convergência (a partir do erro de quantização e da variação média e máxima da atualização dos pesos), análise do erro topológico e com a visualização final da rede.

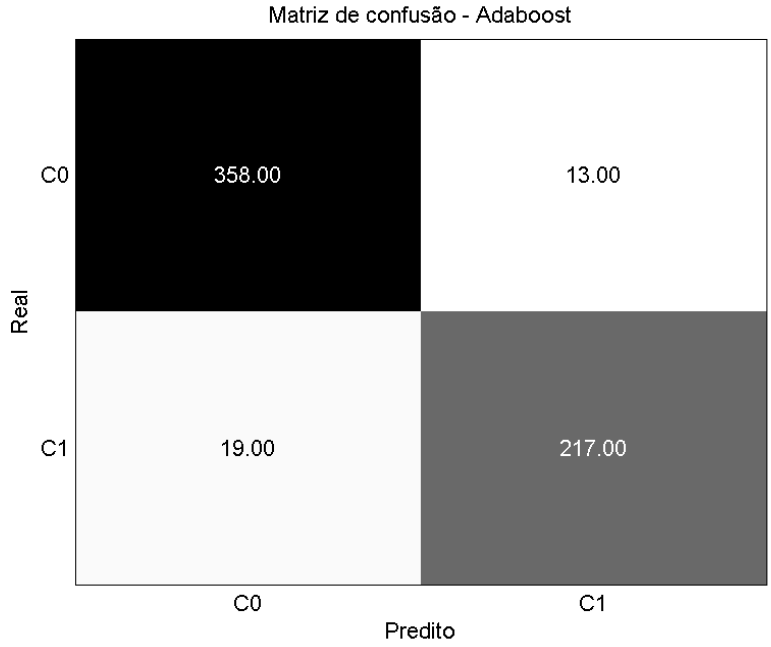

Figura 12: Matriz de confusão de um ensemble para o dataset de spam. Execuções feitas com 35 componentes e para cada um deles $h_0 = 10$ neurônios na camada oculta, $m = 1000$ como máximo de épocas e sem parada prematura. *C0* e *C1* correspondem às classes de e-mails comuns e spams, respectivamente

No geral, a parametrização inicial se mostrou apropriada para o problema e pouca auteração foi realizada. O raio inicial foi diminuido para 15 enquanto a dimensão do grid foi aumentada para 40x40. Esse aumento foi motivado pela criação da U-Matrix, em que decidiu-se usar um grid que garantisse uma boa resolução para este tipo de representação visual.

Tentativas de alterar a taxa de decaimento do raio e da taxa de aprendizado não foram bem sucedidas, os valores de 32 para ambas foram os que apresentaram uma visualização mais precisa dos dados. O número total de épocas foi diminuido para 100 a partir da observação do erro topológico (que será melhor discutido na Análise de Erros).

*2) **Visualização**:* Para a visualização da rede nos beneficiamos do fato do conjunto de entrada ser em duas dimensões para criar o mapa de neurônios no espaço vetorial com as vizinhanças topológicas. A figura **??** apresenta o Grid final da rede.

Com o Grid podemos notar como os neurônios se aproximam dos dados originais se concentrando sobre regiões mais densas. Pode-se até mesmo ver os desenhos dos clusters nitidamente representados pela rede.

A visualização do Grid também é uma boa forma de visualizar o funcionamento da rede durante seu treinamento. A figura 13 mostra como os neurônios se comportam durante a organização do SOM.

No inicio da execução os neurônios são aleatoriamente distribuídos pelo espaço vetorial, sem forma ou ordem. Após 33 iterações percebe-se uma ordenação central, isso se dá, pela alta função de vizinhança no inicio do aprendizado onde os neurônios vencedores influênciam uma grande quantidade de neurônios circunvizinhos. Com 66 iterações essa função de vizinhança já caiu e a rede começa a se expandir e abrir sobre os dados. E por fim, após 100 iterações a rede passa por um ajuste fino no qual os neurônios se aproximam dos dados e passa a representar mais fielmente o espaço vetorial em um espaço bidimensional.

Outro método de visualização implementado foi a U-matrix. Essa representação apresenta uma perspectiva bi ou tri-dimensional da rede gerada, calculada a partir das distâncias vetoriais dos neurônios em relação a sua vizinhança. Diferentemente do Grid apresentado anteriormente, a U-matrix garante a visualização para conjuntos em altas dimensões.

A U-matrix representada na imagen 15 foi criada a partir da rede de 40x40 neurônios dos testes, o que gera uma matriz de 79x79 tando uma resolução de 6241 pontos. A graduação de cores varia do azul escuro, para distâncias próximas, ao vermelo escuro, refletindo maiores distâncias.

Pode-se notar distintamente os 6 grupos de dados do conjunto original em azul.

A imagem 16 apresenta o formato 3D da matriz como uma superfície. Nessa representação a coloração se mantém, porém, é adicionado informações de profundidade. Os vales são grupos de dados próximos, enquanto os montes são separações de maior distância.

Ao gerar a U-matrix o algoritmo pode apresentar uma visualização invertida da imagem. Isso se da pela inicialização randômica da rede, onde os neurônios ao se organizarem não seguem uma orientação específica sobre o espaço.

*3) **Análise do Aprendizado**:* Para analisar o aprendizado e a convergência da rede decidimos observar a variação dos pesos dos neurônios em função das iterações. A figura 18 apresenta a atualização (deslocamento) médio e máximo dos neurônios a cada iteração.

O comportamento exponencial negativo da curva mostra como a rede aprende mais no inicio e com o tempo para de se mover e acaba covergindo. Este comportamento também está diretamente relacionado à taxa de aprendizado que também decai em função do tempo.

Percebe-se também que a atualização média (figura 18(a)) cai mais rapidamente que o a atualização máxima(figura 18(b)), isto porque ela ainda está realacionada à função de vizinhança que também decai em função do tempo.

Esse decaimento da função de vizinhança pode ser visto na figura 17, na qual o pico em 1 representa o BMU e a superfície é a influência excitatória sobre os neurônios circunvizinhos. Observa-se que a abertura da função vai diminuindo com o passar das épocas, e ao final apenas vizinhos próximos são influenciados pela atualização do BMU. O formato tetraédrico da superfície ocorre por assumirmos apenas a vizinhança direta dos neurônio no espaço matricial, calculada como uma distância manhattan. A suavisação dessas bordas poderia ser obtida utilizando uma distância euclidiana sobre o espaço matricial.

*4) **Análise de Erros**:* Além da forma como a rede atualiza seus pesos, duas medidas de erro foram estudadas para maior compreensão da qualidade da rede, o erro de quantização e o erro topológico.

O erro de quantização também se mostrou uma excelente ferramenta para avaliar a convergência do algoritmo. A figura 19(a) apresenta o decaimento do erro de quantização da rede dentro de 300 iterações. Percebe-se um convergência próxima de zero, com um valor final de 0.0251333, porém, com apenas 150 iterações obtem-se 0.02560752 de erro.

Já o erro topológico a princípio foi motivo de confusão sobre o comportamento da rede. Como pode ser visto na imagem 19(b), essa medida demora mais para convergir e além disso, sua convergência ocorre com valores piores que os iniciais.

A interpretação desse erro pode ser feita da seguinte forma: No começo do algoritmo a rede possui um erro topológico alto por causa da inicialização randômica dos neurônios. Este erro vai se amenizando conforme a rede abre e se organiza no espaço dos dados (como pode ser visto na imagem 13(c)). Entretanto, essa medida volta a subir enquanto a rede realiza o ajuste fino, após algum tempo o ajuste começa a ser excessivo e a rede começa a distorcer, fazendo com que o erro dispare.

Um fato que foi observado pelo grupo é que após o erro começar a subir, há um ponto em que é necessário interromper o treinamento para evitar distorções. Esse ponto deve ser antes do erro se tornar pior que no início.

Estranhamos esse comportamento no inicio, pois estávamos análisando a rede pela representação visual do Grid dos neurônios finais. O grid da rede após 300 iterações (figura 14(b)) aparenta ter uma maior proximidade aos dados de entrada do que o grid da imagem após 100 iterações(figura 14(a)), porém, analisando o erro topológico verifica-se que esta aproximação excessiva são distorções.

Pode-se visualizar como essas distorções prejudicam a rede a partir da U-matrix. Comparando a U-matrix resultante após 100 iterações (figura 15(a)) com a que resulta após 300 iterações (figura 15(b)) nota-se a perda de qualidade para identificação de grupos.

*5) **Influencia dos Neurônios**:* Uma outra forma de visualização implementada foi a Quantização. Essa visualização é construída conectando cada dados do conjunto de entrada ao neurônio mais próximo (BMU). A figura 20(a) apresenta a quantização obtida pela rede após 100 época. A quantização possibilita observar mais claramente a redução da dimensionalidade, onde as 8000 entradas do conjunto original são representadas por apenas 1600 vetores, o equivalente a 20% dos dados.

Nota-se ainda que alguns neurônios possuem uma área de influência mais ampla e uma maior quantidade de dados influenciados em relação a outros neurônios. Podemos definir um limiar de influência mínima para que apenas neurônio mais representativos sejam considerados. A figura 20(b) apresenta esse a quantização apenas com neurônios com influência mínima sobre 5 instâncias. É interessante notar que ao definir uma influência mínima, excluindo neurônios pouco representativos, garante-se uma redução ainda maior da dimensionalidade dos dados. Com a influência mínima de 5 no exemplo, temos a redução de 1600 vetores para apenas 655, o que reduz o conjunto de entrada para 8,18%.

B. Path-Based2:Spiral

Para a execução do SOM para essa base de dados testou-se diferentes configurações de parâmetros buscando garantir a diminuição da dimensionalidade e a boa visualização da saída. Porém, nenhuma configuração foi suficiente para garantir o resultado esperado. Acreditamos que isso ocorreu pela baixa quantidade de exemplos do conjunto de entrada, a diminuição do conjunto para um número menor de neurônios implica em uma baixa qualidade da U-matrix.

O melhor resultado com diminuição de dimensionalidade pode ser visto a partir do grid gerado na figura 21(a) e da U-matrix da figura 22(a). Para obter esses resultados o SOM foi parametrizado com um raio inicial de 15, uma taxa de aprendizado inicial de 0.9, taxa de decaimento do raio de 32, taxa de decaimento da taxa de aprendizado de 32, dimensão de 10x10 e com 125 épocas. Esse número de épocas foi obtido após testes e análises de convergência e comportamento do erro. O erro de quatização obtido foi 0.1072 e o erro topológico obtido foi 0.0673.

Já ignorando a diminuição da dimensionalidade e focando na visualização, mantivemos os mesmos valores de parâmetros e alteramos apenas o número de neurônios de saída, aumentando a dimensão para 17x17 (o que representa 92.62% dos 312 dados de entrada). As figuras 21(b) e 22(b) apresentam as visualizações do Grid e da U-matrix muito mais informativas sobre os grupos do conjunto original. O erro de quatização obtido foi de 0.0535 e o erro topológico foi de 0.1378.

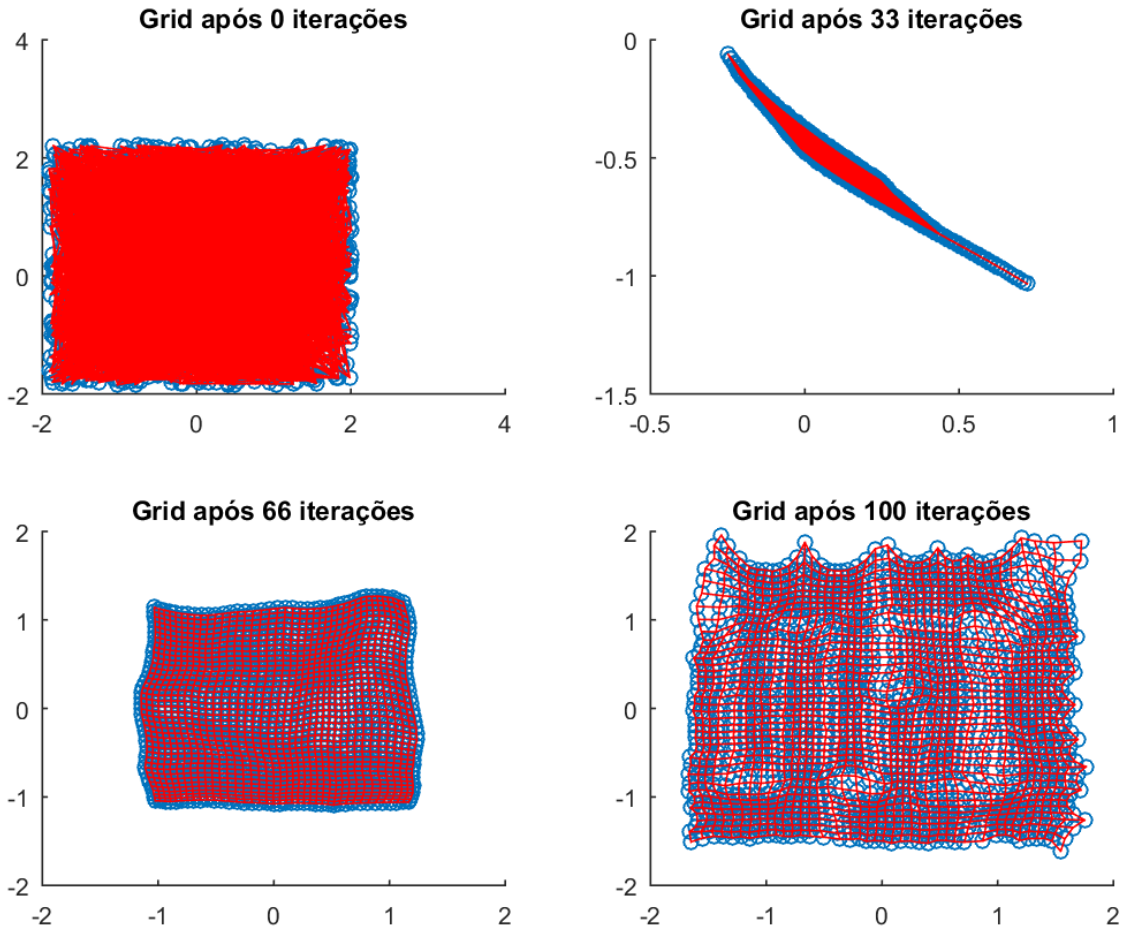

Figura 13: Evolução do Grid durante o aprendizado do SOM. (a) Estutura do Grid inicial com pesos aleatórios. (b) Estrutura do Grid iniciando a organização (c) Estrutura do Grid após abertura sobre os dados. (d) Estrutura do Grid após a convergência.

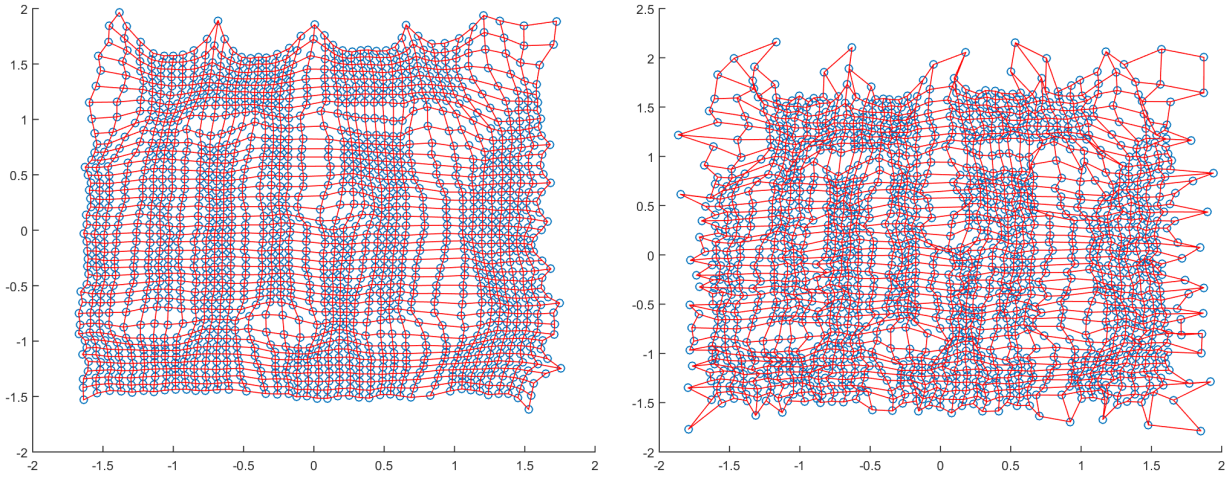

Figura 14: Grid final da rede com sua melhor parametrização. (a) Estruturação ideal após 100 épocas. (b) Estruturação distorcida após 300 épocas

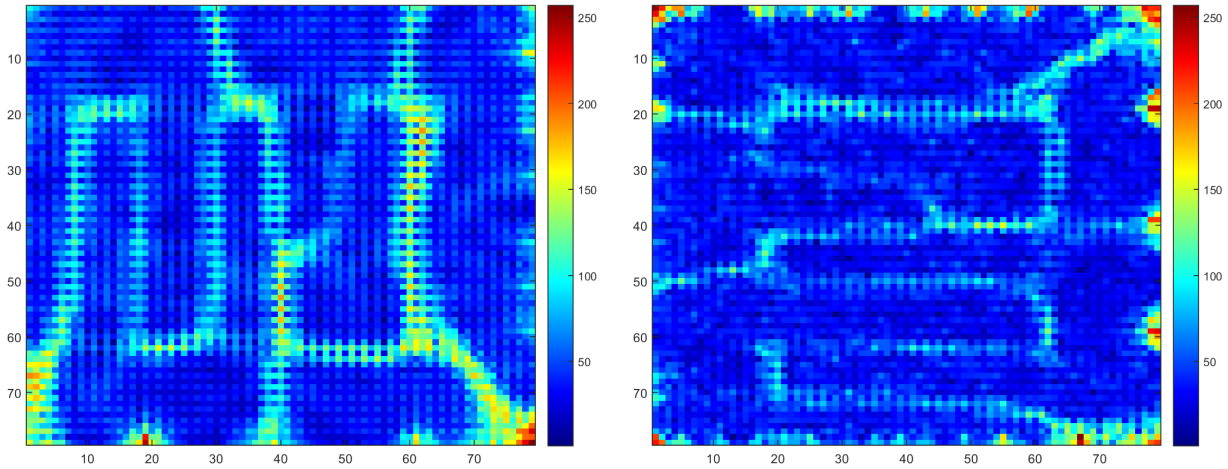

Figura 15: Matriz-U 2D: (a) Resultados após 100 épocas. (b) Resultado distorcido após 300 épocas.

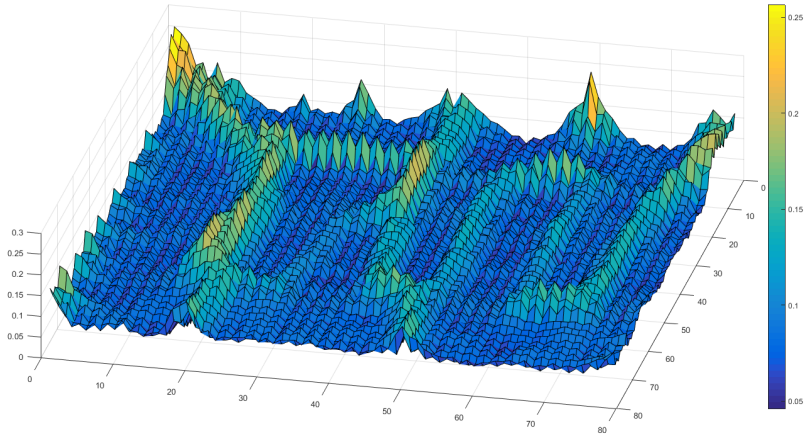

Figura 16: Matriz-U 3D após 100 épocas

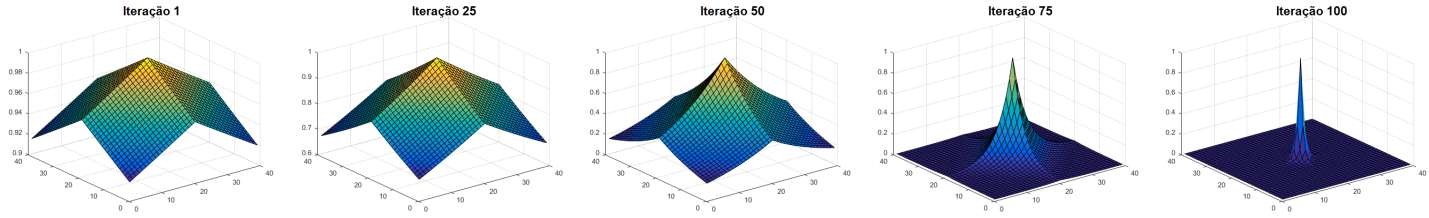

Figura 17: Evolução da Função de Vizinhança

Podemos desprender disso que quando o foco da utilização do SOM for a visualização dos dados, talvez seja interessante manter a quantidade de neurônios na saída da rede semelhante a quantidade de dados da entrada.

VII. Tarefa de Associação

A. *Congressional Voting Records*

Primeiramente devemos analisar a distribuição de classes dos respresentantes, com 61,38% de democratas e 38,62% de republicanos. Esse desbalanceamento das classes facilita a mineração de dados sobre democratas e dificulta sobre republicanos. Executar um algoritmo de regras de associação com um suporte maior que 40% eliminará qualquer referência a republicanos.

Começamos a análise com uma parametrização bastante restritiva, com um suporte de 50% e uma confiança de 90%. Esse teste retornou apenas 12 itemsets frequentes com 1 item, 4 itemsets frequentes com 2 itens e 1 itemset frequente com 3 itens. Com esses resultados foi possível a construção de 6 regras de associação fortes

- Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat (0.913)
- Aid-to-Nicaraguan-Contras=Y $\Longrightarrow$ Democrat (0.901)
- Physician-Fee-Freeze=N $\Longrightarrow$ Democrat (0.992)

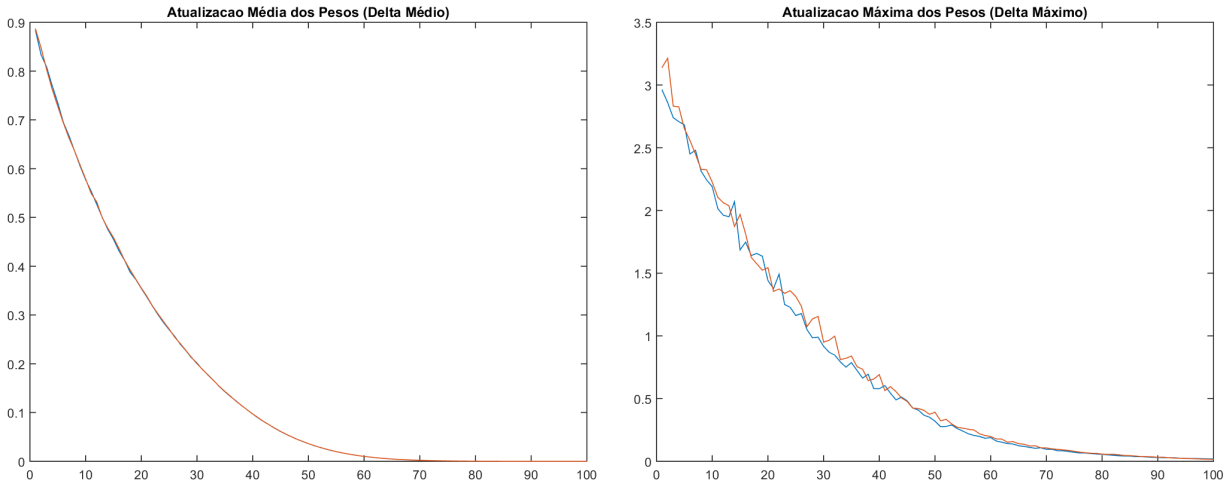

Figura 18: Atualização dos pesos da rede durante o treinamento até a convergência: (a) Média de atualizações. (b) Valor máximo de atualização

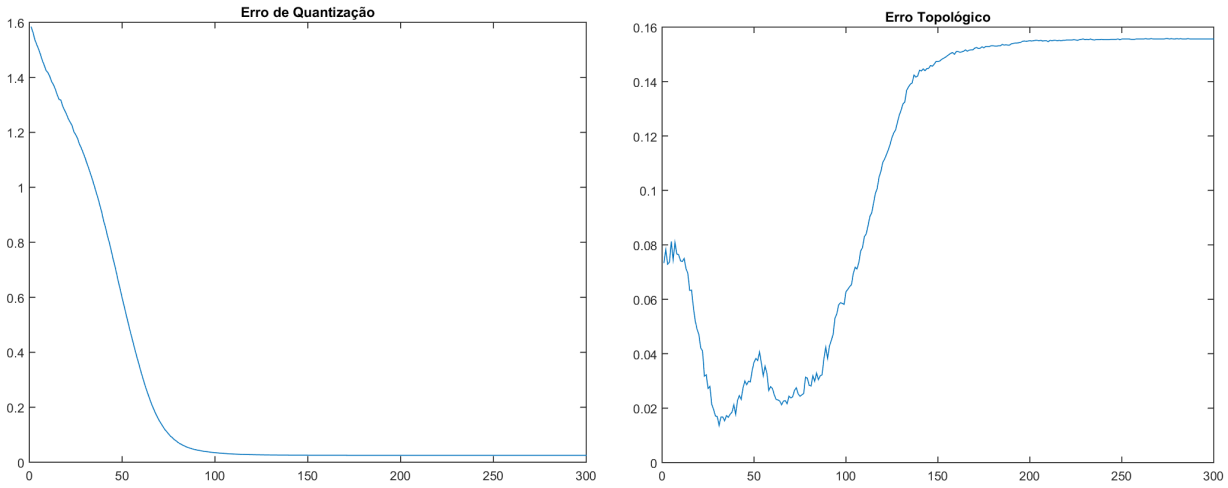

Figura 19: Erro de Quantização (a) e Erro Topológico (b) no intervalo de 300 épocas

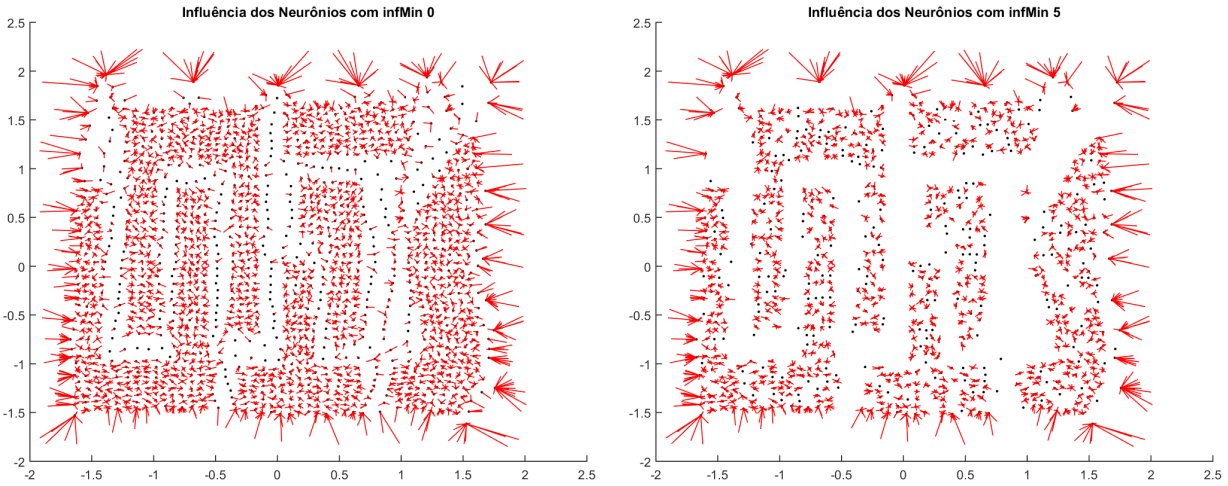

Figura 20: Quantização obtida após 100 épocas: (a) Influência de todos os neurônios. (b) Neurônios com influência mínima sobre 5 instâncias

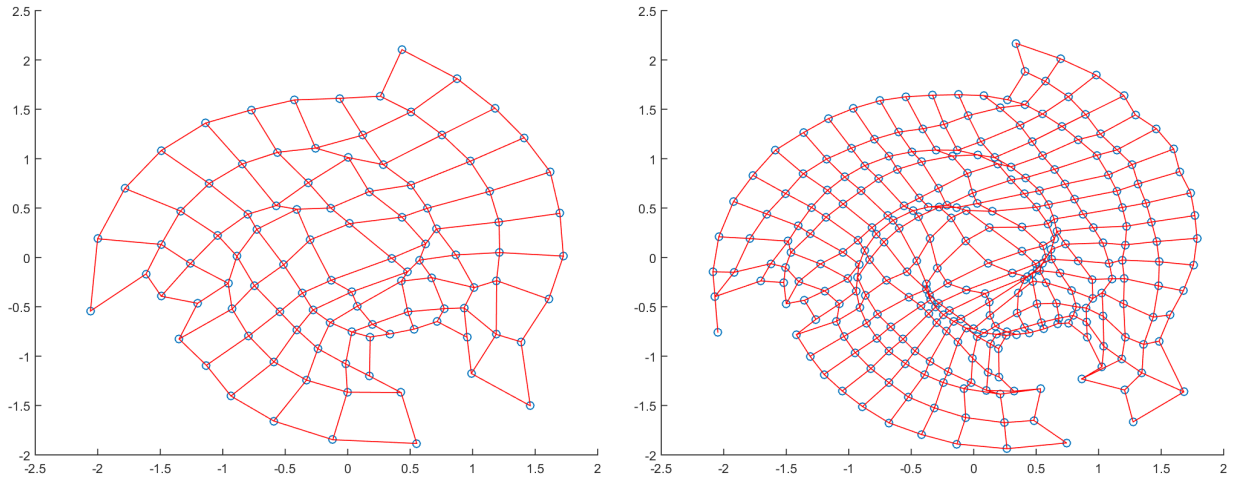

Figura 21: Grid final da rede para o Path-Based2:spiral. (a) Dimensão da rede de 10x10 neurônios. (b) Dimensão da rede de 17x17 neurônios.

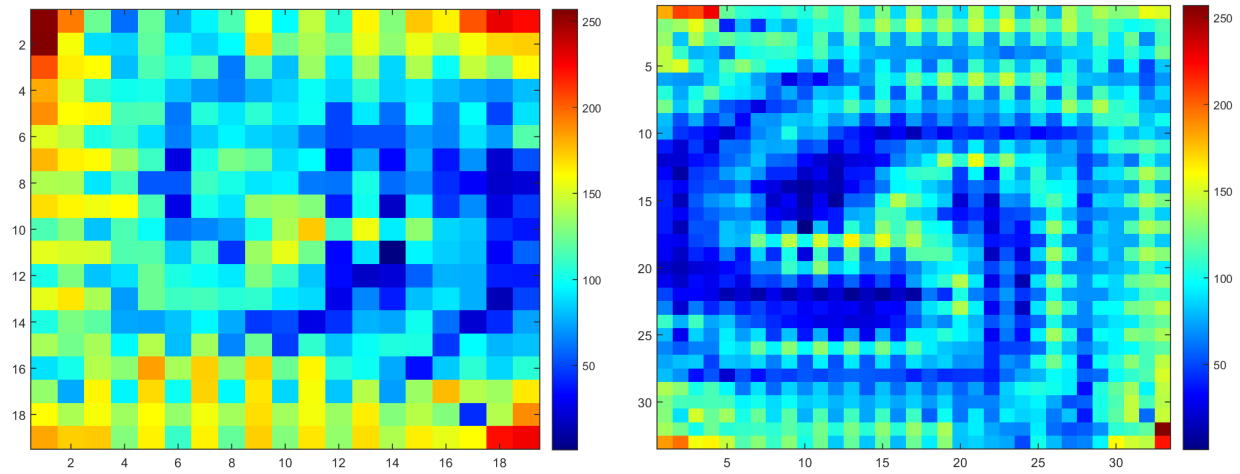

Figura 22: U-matrix do Path-Based2:spiral. (a) Dimensão da rede de 10x10 neurônios (b) Dimensão da rede de 17x17 neurônios

- Democrat $\Longrightarrow$ Physician-Fee-Freeze=N (0.918)
- Physician-Fee-Freeze=N, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat (1.000)
- Democrat, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Physician-Fee-Freeze=N (0.948)

Com esses resultados já conseguimos minerar as associações mais frequentes. Podemos notar, por exemplo, que quase todos os democratas votaram contrários ao Physician Fee Freeze. Ainda podemos identificar a polaridade existente no congresso americano, para as votações favoráveis a Adoption of the Budget Resolution, Aid to Nicaraguan Contras e contrárias a Physician Fee Freeze, com mais de 90% de confiança os votos foram de democratas, deixando os republicanos no outro espectro dos votos (contrários/favoráveis ou indefinido).

Essa polaridade pode ainda ser notada ao verificarmos que nenhuma votação obteve consentimento de votos, com o maior itemset único sendo Religious Groups in Schools que obteve 62,50% de suporte.

Buscando regras menos frequentes, diminuimos a confiança para 40% mantendo o suporte em 90%. Essa execução resultou em 27 itemsets frequentes com 1 item, 41 itemsets frequentes com 2 items, 33 itemsets frequentes com 3 items, 14 itemsets frequentes com 4 items e 3 itemsets frequentes com 5 items. Dentre esses 118 itemsets frequentes, pode-se obter 192 regras fortes.

Sete dessas regras obtiveram 100% de confiança, todas elas relacionando um padrão de votos a democratas,

- Physician-Fee-Freeze=N, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat
- Physician-Fee-Freeze=N, Education-Spending=N, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat
- Aid-to-Nicaraguan-Contras=Y, Physician-Fee-Freeze=N, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat
- Physician-Fee-Freeze=N, Anti-Satellite-Test-Ban=Y, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat
- Physician-Fee-Freeze=N, El-Salvador-Aid=N, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat
- Aid-to-Nicaraguan-Contras=Y, Physician-Fee-Freeze=N, El-Salvador-Aid=N, Adoption-of-the-Budget-

Resolution=Y $\Longrightarrow$ Democrat
- Aid-to-Nicaraguan-Contras=Y, Physician-Fee-Freeze=N, Anti-Satellite-Test-Ban=Y, Adoption-of-the-Budget-Resolution=Y $\Longrightarrow$ Democrat

Podemos notar uma alta associação entre representantes a favor de Aid to Nicaraguan Contras, Anti Satellite Test Ban, Adoption of the Budget Resolution e contra Physician Fee Freeze e El Salvador Aid como sendo Democratas.

Com exceção da primeira regra (que já apareceu no último teste com 50% de suporte) todas as outras possuem um suporte entre 40% e 41,6%. Como o esquema de classes é binário e elas estão sempre do lado direito da relação, podemos calcular a partir desses valores a confiança mínima necessária para se obter a recíproca dessas regras. Assim, com uma confiança mínima de 65% temos o retorno das implicações descritas a cima.

Para obter regras de associação referêntes aos republicanos, diminuimos o suporte mínimo ainda mais para 30% e mantivemos a confiança mínima em 90%. 973 itemsets foram retornados e 2990 regras foram criadas.

Dentre essas regras, três delas apresentam comportamento quase unanime dentre os republicanos, são elas:

- Republican $\Longrightarrow$ Crime=Y - (0.940)
- Republican $\Longrightarrow$ El-Salvador-Aid=Y - (0.935)
- Republican $\Longrightarrow$ Physician-Fee-Freeze=Y - (0.970)

Novamente a polaridade do congresso fica evidente ao notarmos uma relação contrária entre a posição de Republicanos e Democratas para a votação de El Salvador Aid, no qual com mais de 90% de confiança republicanos votaram a favor enquanto Democratas votaram contrários.

B. Contraceptive Method Choice

Começamos novamente a análise com uma parametrização bastante restritiva, com um suporte de 70% e uma confiança de 80%. Esse teste retornou quatro itemsets frequentes, sendo três deles com 1 item e apenas um com 2 items. A partir desses itemsets as duas únicas regras possíveis de serem criadas apresentaram confiança suficiente para serem regras fortes. As regras são:

- MediaExposure=Good $\Longrightarrow$ Religion=Islam (0.845)
- Religion=Islam $\Longrightarrow$ MediaExposure=Good (0.919)

Essas regras não aparentam apresentar informações relevantes de associação de variáveis, uma vez que o suporte de cada item sozinho é alto (Religion=Islan com 80% e MediaExposure=Good com 92.60%) refletindo regras óbvias, estando mais relacionadas a cultura do país do que relação de causalidade.

Decidimos então retirar o atributo MediaExposure e executar o algoritmo novamente diminuindo o suporte para 30% e a mantendo a confiança em 80%. Foram retornados 110 itemsets, o que possibilitou a criação de 46 regras dentro da confiança mínima definida.

Encontramos regras fortes bastante relacionadas a religião.

- Husband Occupation=1 $\Longrightarrow$ Religion=Islam (0.835)
- Husband Occupation=3 $\Longrightarrow$ Religion=Islam (0.897)
- Contraceptive=No-use $\Longrightarrow$ Religion=Islam (0.881)
- Contraceptive=Short-term $\Longrightarrow$ Religion=Islam (0.865)
- Husband's Education=mid-high $\Longrightarrow$ Religion=Islam (0.923)
- Standard-of-living=mid-high $\Longrightarrow$ Religion=Islam (0.896)
- Wife's Education=mid-high $\Longrightarrow$ Religion=Islam (0.873)
- Wife's Education=mid-low $\Longrightarrow$ Religion=Islam (0.934)
- $[20 < age <= 30] \Longrightarrow$ Religion=Islam (0.916)

Nota-se o aumento da distribuição de Islâmicos dentro de certas condições. Mulheres em que a ocupação do marido é classificada como 1 e 3 têm 83.5% e 89.7% de confiança de serem Islâmicas. Da mesma forma, mulheres com idade entre 20 e 30 anos são de religião Islâmica com confiança de 91.6%.

Outras regras fortes foram encontradas em relação a educação do marido.

- Husband Occupation=1 $\Longrightarrow$ Husband Education=high (0.915)
- Wife's Education=high $\Longrightarrow$ Husband Education =4',) - 0.943)

Aqui percebe-se que os maridos que possuem uma ocupação classificada como 1 possuem, com confiança de 91.5%, um alto grau de educação. De forma análoga, mulheres que possuem um alto grau de educaçao são casadas com maridos que também possuem um alto grau de educação.

REFERÊNCIAS

[1] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[2] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191–203, 2008.

[3] E. Frank and M. Hall, "A simple approach to ordinal classification," *Machine Learning: ECML 2001*, pp. 145–156, 2001.

[4] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory*. Springer, 1995, pp. 23–37.