Agrupamento baseado em SOM com pesos adaptativos para múltiplas tabelas de dissimilaridade

Anderson B. S. Dantas[1], **Francisco de A. T. de Carvalho**[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (CIn/UFPE)
Av. Prof. Luiz Freire, s/n - Cidade Universitária, CEP 50740-540, Recife – PE – Brazil

{absd,fatc}@cin.ufpe.br

Abstract. *This paper introduces a clustering algorithm based on batch Self-organizing map to partition objects taking into account their relational descriptions given by multiple dissimilarity matrices. The presented approach provide a partition of the objects and a prototype for each cluster, moreover the method is capable of learn relevance weights for each dissimilarity matrix by optimizing an adequacy criterion that measures the fit between clusters and the respectives prototypes. These relevance weights change at each iteration and are different from one cluster to another. Experiments using real-world data bases are considered to show the usefulness of the method.*

Resumo. *Este trabalho apresenta um modelo de algoritmo de agrupamento baseado no algoritmo SOM com treinamento batch para classificar objetos levando em consideração suas descrições relacionais dadas por múltiplas matrizes de dissimilaridade. O método apresentado tem como resultado uma partição dos dados de entrada, assim como um protótipo para cada agrupamento, além de adaptar pesos que calculam a relevância de cada matriz de dissimilaridade otimizando um critério de adequação entre os agrupamentos e seus respectivos protótipos. Estes pesos mudam a cada iteração do algoritmo e são diferentes de um grupo para outro. Experimentos com bases de dados reais foram realizado com o objetivo de mostrar a utilidade do algoritmo.*

1. Introdução

Algoritmos de agrupamento (*clutering*) têm sido largamente utilizados em áreas de aprendizagem de máquina como mineração de dados, recuperação de documentos, segmentação de imagens e classificação de padrões [Jain et al. 1999]. Métodos de agrupamento objetivam organizar um conjunto de dados em grupos (*clusters*), de forma que itens pertencentes a um mesmo grupo (*cluster*) tenham alto grau de similaridade, por outro lado, itens em grupos diferentes possuem alto grau de dissimilaridade.

O mapa auto-organizável de Kohonen (Self-organizing map) [Kohonen 1990] é um tipo especial de rede neural não-supervisionada e possui propriedades de agrupamento e visualização [Gopalakrishnan et al. 2008]. SOM pode ser considerado um algoritmo que faz a projeção de dados multidimensionais em um espaço de uma, duas ou, em casos especiais, três dimensões. Esta projeção possibilita uma partição das entradas em grupos similares ao mesmo tempo em que preserva a topologia original dos dados.

Existem duas representações comuns dos objetos em que os algoritmos de agrupamento podem ser baseados: dados caracterizados ou relacionais. Quando cada objeto é

descrito por um vetor de valores quantitativos ou qualitativos, os vetores que descrevem os objetos são chamados dados caracterizados. Quando cada par de objetos é representado por uma relação então temos dados relacionais. O modelo mais comum de dados relacionais é o caso de uma matriz de dissimilaridades $R = [r_{il}]$, onde r_{il} é a dissimilaridade pareada (geralmente uma distância) entre os objetos i e l.

[Golli et al. 2004] propõe uma adaptação do modelo SOM para dados de dissimilaridade. [Lechevallier et al. 2010] apresenta um algoritmo de agrupamento que realiza partição de objetos baseado nas descrições relacionais destes objetos dadas por múltiplas matrizes de dissimilaridade.

Este trabalho propõe um algoritmo baseado em SOM com pesos adaptativos para classificar dados baseados em matrizes de dissimilaridades. Estas matrizes podem ser obtidas através de diferentes conjuntos de variáveis e uma função de dissimilaridade fixa (neste caso a partição final apresenta um consenso entre diferentes conjuntos de variáveis descrevendo os objetos), através de um conjunto fixo de variáveis e diferentes funções de dissimilaridade (consenso entre diferentes funções de dissimilaridade) ou através de diferentes conjuntos de variáveis e funções de dissimilaridade. A relevância de diferentes matrizes de dissimilaridades não é igualitária na definição final dos agrupamentos. Assim, para obter uma partição significativa partindo de todas as matrizes faz-se necessário o uso de pesos adaptativos para cada matriz de dissimilaridade e diferentes para cada agrupamento.

Este trabalho está organizado da seguinte forma: a seção 2 traz uma introdução sobre o algoritmo SOM e sua definição formal. A seção 3 trata do algoritmo SOM com treinamento batch aplicado a dados de dissimilaridade. O algoritmo SOM com pesos adaptativos aplicado a dados representados por matrizes de dissimilaridade é apresentado na seção 4. Os experimentos realizados com bases de dados reais são apresentados na seção 5. Finalmente, a seção 6 traz as conclusões do trabalho.

2. Self-organizing map (SOM)

Os algoritmos chamados *self-organizing maps* [Kohonen 1990] (mapas auto-organizáveis) fazem parte da classe de métodos de aprendizado não-supervisionado, todas as propriedades de cada grupo em que são classificados os dados, são estimadas ou aprendidas sem o uso de informação a priori [Murtagh and Hernández-Pajares 1995]. O objetivo central desses modelos é encontrar uma estrutura nos dados fornecidos.

A arquitetura do algoritmo SOM consiste em um conjunto de neurônios organizados topologicamente em uma grade, geralmente com uma, duas, ou três dimensões, chamada mapa. De modo mais formal, o mapa é descrito por um grafo não-orientado (C, Γ). C é um conjunto de m neurônios interconectados com topologia definida por Γ. A estrutura de grafo permite que se defina uma função de distância entre dois neurônios no mapa. Para cada par de neurônios (*c,r*) no mapa, $\delta(c, r)$ é o comprimento do caminho mais curto entre *c* e *r* no grafo *C*. Essa função de distância permite que haja uma relação de vizinhança entre neurônios. Cada neurônio c é representado por um vetor de referência $w_c = \{w_c^1, \ldots, w_c^p\}$, onde p é igual à dimensão dos vetores de entrada. A quantidade de neurônios varia de acordo com a aplicação.

A principal característica dos mapas auto-organizáveis é a possibilidade de se comparar os agrupamentos. Cada elemento que compõe os dados fornecidos é afetado a

um grupo. Cada grupo é projetado em um neurônio do mapa. Elementos semelhantes são projetados no mesmo neurônio, enquanto que a dissimilaridade aumenta com a distância que separa duas projeções.

O conceito de vizinhança é incorporado ao algoritmo através do uso de uma função *kernel* K positiva e tal que $\lim_{|x|\to\infty} K(x) = 0$ [Badran et al. 2005]. As distâncias $\delta(c, r)$ entre os neurônios *c* e *r* do mapa permitem a definição da influência relativa dos neurônios sobre os objetos. $K(\delta(c, r))$ quantifica essa influência.

Para que o tamanho da vizinhança seja um parâmetro relevante, é utilizada a família de *kernels* K^T parametrizada por T: $K^T(\delta) = K(\delta/T)$. Quanto menor o valor de T, menor a quantidade de neurônios pertencentes à vizinhança de um determinado neurônio c. O valor de T, na iteração t, é atualizado segundo a equação:

$$T = T_{max} * (\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}} \tag{1}$$

T_{min} e T_{max} representam, respectivamente, o valor final e inicial de T. N_{iter} é o número máximo de iterações.

Seja $E = \{e_1, \ldots, e_n\}$ um conjunto de n objetos. A cada iteração do algoritmo de treinamento, são realizadas duas etapas. A primeira etapa é uma etapa de afetação, a segunda fase é de minimização de uma função de custo J. Durante a etapa de afetação, os dados são particionados de acordo com uma função de afetação f, que associa cada elemento e_i do conjunto de dados ao vetor de referência mais "próximo"a e_i. O algoritmo de Kohonen usa a seguinte função de afetação:

$$f(e_i) = arg\min_{c} \parallel e_i - \mathbf{w}_c \parallel^2 \tag{2}$$

A função de custo é definida da seguinte forma:

$$J(f, W) = \sum_{e_i \in E} \sum_{c \in C} K^T(\delta(c, f(e_i))) \parallel e - \mathbf{w}_c \parallel^2 . \tag{3}$$

Onde W é o conjunto de vetores de referência e $\parallel \cdot \parallel$ representa uma medida de dissimilaridade que pode ser expressa em termos de uma função de distância, por exemplo, a distância Euclidiana seria uma escolha plausível. É importante notar que neste tipo de algoritmo somente um indivíduo é apresentado por vez, os neurônios do mapa se ajustam às entradas a medida em que elas são apresentadas.

A etapa seguinte, de minimização, atualiza o conjunto de vetores de referência W para minimizar a função de custo. O novo conjunto W^t é calculado de acordo com:

$$w_c^t = w_c^{t-1} - \mu^t K^T(\delta(c, f_t(e_i)))(w_c^{t-1} - e_i), \tag{4}$$

onde μ^t é a taxa de aprendizado na iteração t.

3. Batch self-organizing map aplicado em dados de dissimilaridade

O algoritmo SOM batch baseado em dados de dissimilaridade conforme proposto por [Golli et al. 2004] também é descrito por um grafo (C, Γ). A principal diferença está nos dados que serão classificados. Neste caso, os dados são representados por uma relação de dissimilaridade. O algoritmo de treinamento do tipo *batch* é um algoritmo iterativo onde todo o conjunto de exemplos (E) é apresentado ao mapa antes que qualquer ajuste seja realizado. A seguir é dada a definição formal deste modelo.

Seja $E = \{e_1, \ldots, e_n\}$ um conjunto de n objetos e uma medida de dissimilaridade $d(e_i, e_l)$ entre os objetos e_i e e_l. Cada neurônio c é representado por um protótipo $g_c = e_{j_i},\ e_{j_i} \in E$. Os protótipos iniciais são selecionados aleatoriamente. Cada iteração do algoritmo alterna entre duas fases (afetação e representação) objetivando minimizar a seguinte função custo (ou função de adequação):

$$J(f, g) = \sum_{i=1}^{n} \sum_{l \in C} K^T(\delta(f(e_i), l)) d(e_i, g_l) \tag{5}$$

Durante a etapa de afetação, a função f afeta cada indivíduo e_i ao neurônio mais próximo. O valor do parâmetro T é definido conforme a equação 1.

$$f(e_i) = arg \min_{c \in C} \sum_{l \in C} K^T(\delta(c, l))\, d(e_i, g_l) \tag{6}$$

Durante a etapa de representação, são selecionados novos protótipos que representam o conjunto de observações. Esta etapa de otimização pode ser realizada independentemente para cada neurônio. O protótipo g_l^* do grupo C_l, que minimiza o critério de adequação J é calculado segundo a equação:

$$g_l^* = arg \min_{e \in E} \sum_{i=1}^{n} \exp\{-\frac{(\delta((f(e_i))^{(t-1)}, l))^2}{2T^2}\} d(e_i, e) \tag{7}$$

4. Self-organizing map adaptativo para múltiplas tabelas de dissimilaridade

4.1. Introdução

Nesta seção apresentamos um algoritmo adaptativo baseado no algoritmo SOM para dados de dissimilaridade [Golli et al. 2004] e no algoritmo de agrupamento para múltiplas tabelas de dissimilaridade de [Lechevallier et al. 2010]. O objetivo do modelo apresentado é mapear objetos levando em conta suas descrições relacionais dadas por múltiplas matrizes de dissimilaridade. O algoritmo é capaz de associar dados de entrada com pontos representativos de um mapa formando grupos de objetos. O método utiliza pesos adaptativos diferentes de um grupo para outro com o intuito de ponderar sobre as matrizes de dissimilaridades e medir sua relevância na formação de cada agrupamento.

Seja $E = \{e_1, \ldots, e_n\}$ um conjunto de n objetos e p o número de matrizes de dissimilaridade $\mathbf{D}_j = [d_j(e_i, e_l)](j = 1, \ldots, p)$, onde $d_j(e_i, e_l)$ é a dissimilaridade entre os objetos e_i e $e_l(i, l = 1, \ldots, n)$ na matriz de dissimilaridades $\mathbf{D}_j$. Assuma que o protótipo

g_l do grupo (*cluster*) C_l é um elemento do conjunto de objetos E, i.e., $g_l \in E \; \forall l = 1, \ldots, L$.

O algoritmo SOM ponderado para múltiplas tabelas de dissimilaridade calcula uma partição $P = (C_1, \ldots, C_l)$ de E em L grupos e o respectivo protótipo g_l representando o grupo C_l em P, tal que um determinado critério de adequação (função objetivo), que mede o ajuste entre os grupos e seus repectivos protótipos é localmente otimizado. O critério de adequação é calculado de forma semelhante às equações 3 e 5:

$$J = \sum_{i=1}^{n} \sum_{l=1}^{c} \exp\{-\frac{(\delta((f(e_i)), l))^2}{2T^2} \sum_{j=1}^{p} \lambda_{lj} d_j(e_i, g_l) \tag{8}$$

onde a função kernel K é definida por: $K^T(\delta) = \exp(-\frac{\delta^2}{T^2})$. $\delta(c, r)$ calcula a distância entre dois neurônios no mapa. A matriz de pesos λ é composta por c vetores de pesos $\lambda_k = (\lambda_k^1, \ldots, \lambda_k^j, \ldots, \lambda_k^p)(k = 1, \ldots, c)$. A matriz de pesos muda a cada iteração do algoritmo, ou seja, os vetores não possuem valores determinados absolutamente e são diferentes para cada matriz de dissimilaridade. O objetivo da matriz de pesos é ponderar sobre cada matriz de dissimilaridade avaliando sua relevância na minimização do critério de adequação. O modelo segue três etapas definidas (representação, atualização de pesos e afetação):

Etapa 1: Definição dos melhores protótipos

Nesta etapa, a partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos. O protótipo $g_l^{(t)} = g^* \in E$ do grupo C_l, que minimiza o critério de adequação E é calculado segundo a equação:

$$g^* = arg \min_{e \in E} \sum_{i=1}^{n} \exp\{-\frac{(\delta((f(e_i))^{(t-1)}, l))^2}{2T^2}\} \sum_{j=1}^{p} \lambda_{lj}^{(t-1)} d_j(e_i, e) \tag{9}$$

Etapa 2: Definição dos melhores pesos

Nesta etapa, a partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ são fixos. O elemento j do vetor de pesos $\lambda_l = (\lambda_k^1, \ldots, \lambda_k^p)$, que minimiza o critério de adequação E é calculado segundo a expressão:

$$\lambda_{lj}^{(t)} = \frac{\left\{\prod_{h=1}^{p} (\sum_{i=1}^{n} \exp\{-\frac{\delta^2(f^{(t-1)}(e_i), l)}{2T^2}\} d_h(e_i, g_l^{(t)}))\right\}^{\frac{1}{p}}}{\sum_{i=1}^{n} \exp\{-\frac{\delta^2(f^{(t-1)}(e_i), l)}{2T^2}\} d_j(e_i, g_l^{(t)})} \tag{10}$$

Etapa 3: Definição da melhor partição

Nesta etapa, os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos. O grupo C_r que minimiza o critério E é determinado de acordo com a equação:

$$r = (f(e_i))^{(t)} = arg \min_{1 \leq h \leq c} \sum_{l=1}^{c} \exp\{-\frac{(\delta(h,l))^2}{2T^2}\} \sum_{j=1}^{p} \lambda_{lj}^{(t)} d_j(e_i, g_l^{(t)}) \quad (11)$$

4.2. O algoritmo

1. *Inicialização*
   Fixe: o número c de grupos, a função δ, o número de iterações N_{iter} e os valores de T_{min} e T_{max};
   Atribua $T \leftarrow T_{max}$; Atribua $t \leftarrow 0$;
   Selecione c protótipos aleatoriamente $g_l^{(0)} \in E(l = 1, \ldots, c)$;
   Configure $\lambda_l^{(0)} = (\lambda_{l1}^{(0)}, \ldots, \lambda_{lp}^{(0)}) = (1, \ldots, 1)(l = 1, \ldots, c)$;
   Configure o mapa $L(c, \mathbf{G}^0)$, onde $\mathbf{G}^0 = (g_1^{(0)}, \ldots, g_c^{(0)})$.
   Cada objeto e_i é afetado ao protótipo mais próximo com o objetivo de obter a partição $P^{(0)} = (P_1^{(0)}, \ldots, P_c^{(0)})$ de acordo com a equação 11, para $t = 0$.
2. *Passo 1: Representação*
   Atribua $t = t + 1$ e $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$;
   A partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos.
   Calcule o protótipo $g_l^{(t)} = g^* \in E$ do grupo $P_l^{(t-1)}(l = 1, \ldots, c)$ de acordo com a equação 9.
3. Passo 2: Atualização dos pesos
   A partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ são fixos.
   Atualize o vetor de pesos de acordo com a equação 10.
4. *Passo 3: Afetação*
   Os protótipos $g_l^{(t)} \in E(l = 1, \ldots, c)$ e $\lambda_l^{(t)}(l = 1, \ldots, c)$ são fixos
   $P^{(t)} \leftarrow P^{(t-1)}$
   para $i = 1$ até n faça:
   encontre o grupo $C_m^{(t)}$ ao qual o objeto e_i pertence
   encontre o grupo vencedor $C_r^{(t)}$, onde r é obtido pela equação 11

   se $r \neq m$:
   $C_r^{(t)} \leftarrow C_r^{(t)} \cup \{e_i\}$
   $C_m^{(t)} \leftarrow C_m^{(t)} \setminus \{e_i\}$

5. *Critério de parada*
   Se $T == T_{min}$ então PARE; senão volte ao passo 1

5. Experimentos e resultados

Para mostrar a utilidade do algoritmo proposto, foram realizados experimentos utilizando bases de dados obtidas do repositório UCI Machine Learning Repository [Frank and Asuncion 2010]. No algoritmo SOM adaptativo para múltiplas tabelas, cada um dos atributos que compõem as bases é representado por uma matriz de dissimilaridade. Para o modelo SOM batch não-adaptativo é considerada somente uma matriz

de dissimilaridade. Com a finalidade de avaliar os resultados dos agrupamentos realizados pelo método proposto neste trabalho foi considerado o índice de Rand corrigido (CR) [Hubert and Arabie 1985], a medida F-*measure* [van Rijsbergen 1979] e a taxa de erro global de classificação (OERC - overall error rate of classification) [Breiman et al. 1984].

O índice CR avalia o grau de similaridade entre a partição a priori e a partição fornecida pelo algoritmo de agrupamento. Além disso, o índice CR não é sensível ao número de classes nas partições ou à distribuição dos itens dentro dos grupos. O índice CR pode assumir valores no intervalo [-1,1], onde o valor 1 indica perfeita combinação entre as partições, valores perto de 0 ou negativos indicam pouca semelhança. O índice F-*measure* entre a partição a priori e a partição obtida pelo algoritmo de agrupamento assume valores no intervalo [0,1], em que 1 indica perfeita combinação entre as partições. O índice OERC mede a habilidade de um algoritmo de agrupamento encontrar classes a priori presentes na base de dados.

Além dos índices é apresentada a matriz de confusão para cada resultado. A matriz de confusão mostra a quantidade de elementos de cada classe que foi classificada em um determinado grupo. É possível, então identificar qual classe um determinado grupo representa.

Alguns parâmetros são considerados para a realização dos experimentos. São eles: topologia do mapa, o valor de T_{min}, T_{max} e o número de iterações N_{iter}. A topologia do mapa define o número máximo de grupos que serão formados. Todos os parâmetros foram definidos empiricamente. A tabela 1 mostra os valores dos parâmetros utilizados em todos os experimentos descritos a seguir. Cada experimento foi repetido 50 vezes com a mesma configuração, dentre as 50 repetições foi selecionada a que obteve menor critério de adequação.

Tabela 1. Parâmetros utilizados nos experimentos

Parâmetros	Valor
Topologia	2x5
T_{min}	0,3
T_{max}	3,0
N_{iter}	500

5.1. Base de dados Wine

A base de dados wine consiste em resultados da análise química de vinhos produzidos numa mesma região na Itália, mas derivados de três cultivos diferentes, formando três classes de vinhos. As classes 1, 2 e 3 possuem, respectivamente 59, 71 e 48 instâncias. Cada indivíduo é descrito por 13 atributos com valores reais representando os valores de 13 componentes encontrados em cada um dos três tipos de vinho.

As tabelas 2 e 3 mostram as matrizes de confusão obtidas pelos algoritmos batch SOM e batch SOM adaptativo para múltiplas tabelas de dissimilaridade, respectivamente. Analisando essas tabelas é possível perceber regiões onde dados da mesma classe a-priori são classificados em grupos próximos no mapa. Na tabela 3, os grupos (0,0), (0,1), (1,0) e (1,1), localizados na parte esquerda do mapa, possuem dados da classe 3, formando uma região bem definida. Os grupos (1,3) e (1,4) possuem dados da classe 1, os demais

grupos são da classe 2. Na tabela 2, na parte mais à esquerda do mapa, predominam dados da classe 2, na parte central observa-se dados da classe 3, enquanto que na parte mais à direita do mapa são predominantes dados pertencentes à classe 1.

Tabela 2. Wine: Matriz de confusão obtida pelo algoritmo batch SOM

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	0	1	7	10	6	0	0	5	16	14
2	20	11	3	0	0	25	6	5	1	0
3	10	16	5	0	0	2	9	6	0	0

Tabela 3. Wine: Matriz de confusão obtida pelo algoritmo batch SOM adaptativo

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	0	0	0	2	6	0	0	0	24	27
2	5	1	21	22	10	0	0	9	2	1
3	16	4	0	0	0	15	13	0	0	0

Os índices CR, F-*measure* e OERC obtiveram, respectivamente, os valores 0,42, 0,52, 0,09 para o modelo SOM adaptativo, e os valores, respectivamente 0,31, 0,45, 0,27 para o modelo SOM batch não-adaptativo. O algoritmo SOM adaptativo para múltiplas tabelas de dissimilaridade obteve melhores índices do que o modelo batch não-adaptativo.

A tabela 4 mostra a matriz de pesos final obtida pelo algoritmo batch SOM adaptativo para múltiplas tabelas de dissimilaridade. Cada peso representa a relevância de uma determinada matriz de dissimilaridade para a formação de cada grupo. Valores acima de 1 indicam uma forte influência da matriz durante a afetação de objetos em um determinado grupo. Por exemplo, na tabela 4 o grupo (0,0) tem uma influência maior da matriz 7 (3,05), enquanto que o grupo (0,1) tem maior influência da matriz 13 (8,64).

5.2. Base de dados Iris

Esta base de dados consiste em três tipos (classes) de planta íris: iris setosa, iris versicolor e iris virginica. Cada uma das classes possui 50 exemplos. Cada instância é descrita por quatro atributos.

As tabelas 5 e 6 mostram a matriz de confusão obtidas pelos algoritmos batch SOM e batch SOM adaptativo para múltiplas tabelas de dissimilaridade. É possível identificar regiões no mapa com dados da mesma classe. Por exemplo, na tabela 5, a parte inferior do mapa tem predominância de dados da classe 3 (grupos (1,0), (1,1), (1,2) e (1,3)) e na tabela 6, a classe 1 está mais presente na parte esquerda do mapa (grupos (0,0) e (1,0)).

Os índices CR, F-*measure* e OERC obtiveram, respectivamente, os valores 0,54, 0,64 e 0,04 para o modelo SOM adaptativo para múltiplas tabelas de dissimilaridade, e os valores, respectivamente 0,50, 0,62, 0,06 para o modelo SOM batch não-adaptativo. O modelo adaptativo obteve melhores índices do que o modelo não adaptativo.

A tabela 7 mostra a matriz de pesos final obtida pelo algoritmo batch SOM adaptativo para múltiplas tabelas de dissimilaridade. Pode-se perceber que os pesos para as

Tabela 4. Wine: Matriz de pesos obtida pelo algoritmo batch SOM adaptativo

Matriz/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	**1,03**	0,99	**1,27**	0,69	0,76	**1,02**	0,46	0,42	0,77	0,48
2	0,27	**1,05**	0,52	0,33	**4,19**	0,38	0,21	0,61	0,55	**6,01**
3	**1,31**	0,55	0,41	0,93	0,27	0,63	0,79	0,51	0,69	0,37
4	**1,07**	**1,09**	0,43	**2,64**	0,99	0,89	0,90	**1,35**	0,16	0,33
5	0,48	0,27	**1,47**	**1,32**	0,16	0,47	**1,85**	0,16	0,29	0,61
6	0,90	0,69	0,60	0,48	**1,92**	**1,68**	0,49	**2,93**	**1,38**	1,64
7	**3,05**	0,79	**2,42**	**2,29**	0,84	**2,42**	**5,07**	**3,12**	**3,13**	**1,65**
8	0,33	0,51	0,90	**1,22**	**1,96**	0,25	0,86	0,24	**2,14**	0,94
9	**1,87**	0,41	**1,57**	0,45	0,24	**2,79**	0,24	**2,27**	**1,12**	0,40
10	**1,25**	0,92	**5,17**	**2,39**	**3,04**	0,46	0,36	**2,96**	**1,95**	**1,50**
11	0,57	**1,80**	0,18	0,36	**3,72**	**1,43**	**2,22**	**1,33**	**1,01**	**1,74**
12	**2,07**	**3,60**	0,51	**4,02**	**2,15**	**5,12**	**5,23**	**1,72**	**3,90**	**2,79**
13	**2,02**	**8,64**	**5,65**	0,63	0,38	0,99	**3,48**	**1,01**	0,86	0,60

Tabela 5. Iris: Matriz de confusão obtida pelo algoritmo batch SOM

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	0	0	0	5	32	0	0	0	0	13
2	1	0	0	0	0	26	7	0	16	0
3	15	11	9	0	0	1	7	7	0	0

Tabela 6. Iris: Matriz de confusão obtida pelo algoritmo SOM adaptativo

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	13	0	0	0	0	37	0	0	0	0
2	0	0	17	17	4	0	2	9	1	0
3	0	9	0	1	14	0	0	0	6	20

matrizes 3 e 4 possuem valores maiores indicando maior influência na formação dos agrupamentos.

Tabela 7. Iris: Matriz de pesos obtida pelo algoritmo batch SOM adaptativo

Matriz/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	0,63	0,80	0,34	0,15	0,59	0,26	**1,59**	0,35	0,25	0,52
2	0,17	0,81	0,26	0,29	0,32	0,07	0,52	0,50	**1,53**	0,26
3	**3,40**	**1,45**	**2,95**	**10,14**	**4,99**	**5,06**	0,95	**1,50**	**1,47**	**2,59**
4	**2,62**	**1,04**	**3,79**	**2,26**	**1,05**	**11,38**	**1,26**	**3,73**	**1,76**	**2,83**

5.3. Base de dados Thyroid

Esta base possui dados sobre os estados da glândula tireóide: normal, hipertireoidismo e hipotireoidismo. As três classes (1, 2 e 3) possuem, respectivamente, 150, 35 e 30 exemplos. Cada instância é descrita por 5 atributos com valores reais.

As tabelas 8 e 9 mostram a matriz de confusão obtidas pelos algoritmos batch SOM e batch SOM adaptativo para múltiplas tabelas de dissimilaridade.

Tabela 8. Thyroid: Matriz de confusão obtida pelo algoritmo batch SOM

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	0	10	8	0	0	25	61	45	1	0
2	16	0	3	0	0	10	6	0	0	0
3	0	3	0	7	5	0	1	1	4	9

Tabela 9. Thyroid: Matriz de confusão obtida pelo algoritmo batch SOM adaptativo

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	0	33	22	0	0	29	38	24	4	0
2	34	1	0	0	0	0	0	0	0	0
3	0	1	0	6	7	0	2	1	0	13

Os índices CR, F-*measure* e OERC obtiveram, respectivamente, os valores 0,37, 0,52 e 0,023 para o modelo SOM adaptativo, e os valores, respectivamente 0,41, 0,55 e 0,11 para o modelo SOM batch não-adaptativo. O algoritmo SOM não-adaptativo para múltiplas tabelas de dissimilaridade obteve melhores índices do que o modelo batch adaptativo, porém o modelo adaptativo obteve um valor menor para o erro global.

A tabela 10 mostra a matriz de pesos final obtida pelo algoritmo batch SOM adaptativo para múltiplas tabelas de dissimilaridade.

5.4. Base de dados Ecoli

A base de dados Ecoli contém dados de proteínas classificadas em 8 classes: (1) cp (citoplasma), com 143 exemplos; (2) im (membrana interna sem sequência de sinal), com 77 exemplos; (3) pp (perisplasm), com 52 exemplos; (4) imU (inner membrane, uncleavable signal sequence), com 35 exemplos; (5) om (outer membrane), com 20 exemplos; (6) omL (outer membrane lipoprotein), com 5 exemplos; (7) imL (inner membrane lipoprotein) com 2 exemplos e (8) imS (inner membrane, cleavable signal sequence), com 2 exemplos. Cada indivíduo é descrito por 6 atributos.

As tabelas 11 e 12 mostram a matriz de confusão obtidas pelos algoritmos batch SOM e batch SOM adaptativo para múltiplas tabelas de dissimilaridade, respectivamente.

Tabela 10. Thyroid: Matriz de pesos obtida pelo algoritmo batch SOM adaptativo

Matriz/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	0,05	0,37	0,21	0,14	**5,38**	0,28	0,12	0,14	**1,15**	0,91
2	0,16	0,17	0,29	**2,96**	**3,20**	0,24	0,24	0,64	0,91	**9,47**
3	0,05	0,20	**4,45**	**3,01**	**1,64**	0,35	0,38	**1,53**	**4,18**	**4,83**
4	**27,94**	**19,29**	**2,61**	**1,27**	0,86	**9,17**	**15,75**	**27,39**	0,24	0,06
5	**85,90**	**4,14**	**1,36**	0,61	0,04	**4,59**	**5,72**	0,26	0,94	0,38

Tabela 11. Ecoli: Matriz de confusão obtida pelo algoritmo batch SOM

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,1	1,2	1,3	1,4
1	47	18	19	0	0	25	27	6	1	0
2	0	1	0	2	25	0	0	0	0	7
3	0	0	0	0	1	0	0	0	1	0
4	0	0	0	0	1	0	0	1	0	0
5	5	0	0	32	24	1	0	0	1	14
6	1	1	0	0	1	0	0	13	35	1
7	0	0	0	0	0	0	0	4	1	0
8	0	0	0	0	0	0	0	8	11	1

Tabela 12. Ecoli: Matriz de confusão obtida pelo algoritmo batch SOM adaptativo

Classe/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,2	1,3	1,4
1	0	0	3	36	57	0	0	3	44
2	10	6	1	0	0	17	0	1	0
3	0	1	0	0	0	0	0	1	0
4	1	0	1	0	0	0	0	0	0
5	23	17	0	8	0	28	0	1	0
6	0	1	5	2	2	0	13	29	0
7	0	0	5	0	0	0	0	0	0
8	0	0	1	0	0	0	11	8	0

Os índices CR, F-*measure* e OERC obtiveram, respectivamente, os valores 0,43, 0,52 e 0,24 para o modelo SOM adaptativo, e os valores, respectivamente 0,37, 0,52 e 0,24 para o modelo SOM batch não-adaptativo. O algoritmo SOM adaptativo para múltiplas tabelas de dissimilaridade obteve índices semelhantes ao modelo batch não-adaptativo, exceto para o índice CR, onde o modelo adaptativo obteve um melhor resultado.

A tabela 13 mostra a matriz de pesos final obtida pelo algoritmo batch SOM adaptativo para múltiplas tabelas de dissimilaridade.

Tabela 13. Ecoli: Matriz de pesos obtida pelo algoritmo batch SOM adaptativo

Matriz/Grupo	0,0	0,1	0,2	0,3	0,4	1,0	1,2	1,3	1,4
1	0,36	0,49	0,96	0,86	**1,26**	0,5	**2,44**	**2,55**	0,48
2	0,5	0,75	0,88	**1,46**	0,9	0,65	0,83	0,53	0,72
3	0,53	0,54	0,78	**1,12**	0,63	0,64	0,25	0,38	0,46
4	**2,55**	**2,27**	**2,09**	0,78	**1,2**	**2,15**	**1,93**	**1,53**	**2,28**
5	**3,96**	**2,16**	0,71	0,89	**1,15**	**2,2**	0,98	**1,25**	**2,69**

6. Conclusão

Este trabalho apresentou um modelo de agrupamento baseado no algoritmo SOM com treinamento batch. Este modelo realiza o agrupamento de objetos levando em consideração seus dados relacionais descritos por uma tabela de dissimilaridades. Como saída, o algoritmo produz uma partição dos dados de entrada, os protótipos de cada

grupo e pesos adaptativos que indicam a relevância de cada tabela de dissimilaridade na formação dos agrupamentos, otimizando um critério de adequação que mede o ajuste entre os grupos e seus protótipos. Tais pesos mudam a cada iteração e são diferentes para cada grupo. Experimentos utilizando bases de dados reais foram realizados para mostrar a utilidade do modelo proposto comparando-o com o modelo apresentado por [Golli et al. 2004]. Na maioria dos experimentos os resultados foram satisfatórios, mostrando a relevância do trabalho proposto, como também, seu desempenho utilizando diferentes bases de dados.

Agradecimentos

Gostaríamos de agradecer pelo suporte financeiro das agências CNPq e FACEPE, que tornaram possível o desenvolvimento e crescimento deste trabalho.

Referências

Badran, F., Yacoub, M., and Thiria, S. (2005). Self-organizing maps and unsupervised classification. In *Neural networks: methodology and applications*, pages 379–442. Springer-Verlag.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC.

Conan-Guez, B., Rossi, F., and Golli, A. E. (2006). Fast algorithm and implementation of dissimilarity self-organizing maps. *Neural Networks*, 19:855–863.

Frank, A. and Asuncion, A. (2010). Uci machine learning repository.

Golli, A., Conan-Guez, B., and Rossi, F. (2004). A self-organizing map for dissimilarity data. In *Classification, Clustering, and Data Mining Applications*, pages 61–68. Springer Berlin Heidelberg.

Gopalakrishnan, K., Khaitan, S., and Manik, A. (2008). Enhanced clustering analysis and visualization using kohonen's self-organizing feature map networks. *International Journal of Computational Intelligence*, 4:64–71.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2:193–218.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Comput. Surv.*, 31:264–323.

Kohonen, T. (1990). The self-organizing maps. *Proceedings of the IEEE*, 78:1464–1480.

Lechevallier, Y., de Carvalho, F. A. T., Despeyroux, T., and de Melo, F. M. (2010). Clustering of multiple dissimilarity data tables for documents categorization. *19th International Conference on Computational Statistics - COMPSTAT 2010*, pages 1263–1270.

Murtagh, F. and Hernández-Pajares, M. (1995). The kohonen self-organizing map method: An assessment. *Journal of Classification*, 12:165–190.

van Rijsbergen, C. J. (1979). *Information Retrieval*. London: Butterworths.