Pós-Graduação em Ciência da Computação

"SOM para dados relacionais baseados em múltiplas tabelas de dissimilaridade"

Por

Anderson Berg dos Santos Dantas

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, 2012

Universidade Federal de Pernambuco
Centro de Informática
Pós-graduação em Ciência da Computação

Anderson Berg dos Santos Dantas

"SOM para dados relacionais baseados em múltiplas tabelas de dissimilaridade"

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Francisco de Assis Tenório de Carvalho*

RECIFE, 2012

Agradecimentos

...

Resumo

Abstract

Sumário

Lista de Figuras

Lista de Tabelas

1

Introdução

Métodos de agrupamento buscam organizar uma coleção de padrões em grupos baseados na similaridade entre eles. Os grupos são gerados de forma que itens localizados em um mesmo grupo possuem alto grau de similaridade, por outro lado, itens de grupos distintos têm alto grau de dissimilaridade. Estes modelos têm sido amplamente aplicados em campos como taxonomia, processamento de imagem, recuperação de informação e mineração de dados (Jain *et al.*, 1999). O propósito destes métodos é encontrar uma estrutura dentro de um conjunto de dados e nenhum dos elementos presentes nesses dados é, ou possui a resposta esperada.

Algoritmos de agrupamento são métodos de classificação não-supervisionada. É importante notar a diferença entre estes e os algoritmos de classificação supervisionada. Na classificação supervisionada, dados previamente classificados, ou seja, já categorizados, são apresentados ao algoritmo para que este aprenda com as descrições dos dados e busque classificar um novo padrão ainda não classificado. Por outro lado, o objetivo dos algoritmos de agrupamento é agrupar um conjunto de dados não classificados, formando grupos por similaridade, ou seja, as categorias são encontradas utilizando-se somente das informações contidas nos dados apresentados. As técnicas mais populares de agrupamento são os métodos hierárquicos e de particionamento (Jain *et al.*, 1999; Xu and Wunsch, 2005).

Métodos hierárquicos fornecem uma hierarquia completa, isto é, uma sequência de partições aninhadas a partir dos dados de entrada. Métodos hierárquicos podem ser aglomerativos (Sneath and Sokal, 1973; Zhang *et al.*, 1996; Guha *et al.*, 1998; Karypis *et al.*, 1999; Guha *et al.*, 2000) ou divisivos (Lance and Williams, 1968; Gowda and Krishna, 1978; Kaufman and Rousseeuw, 1990; Guenoche *et al.*, 1991; Chavent, 1998). Métodos aglomerativos fornecem uma sequência de partições aninhadas iniciando com agrupamento trivial onde cada item está em um único grupo e termina com um

agrupamento onde todos os itens estão no mesmo grupo. Um método divisivo inicia com todos os itens no mesmo grupo e executa um procedimento de divisão até que um critério de parada seja alcançado.

Métodos de particionamento buscam obter uma única partição dos dados em um número fixo de agrupamentos. Estes métodos frequentemente buscam uma partição que otimize (geralmente uma solução local) uma função objetivo. Para melhorar a qualidade do agrupamento, o algoritmo é executado diversas vezes com diferentes inicializações e a melhor configuração obtida do total de execuções é usada como saída do algoritmo de agrupamento. Métodos de particionamento podem ser divididos em agrupamentos HARD (Forgy, 1965; Huang, 1998; Kanungo *et al.*, 2000; Hansen and Mladenoviae, 2001; Su and Chou, 2001) e agrupamentos fuzzy (Bezdek, 1981; Hoeppner *et al.*, 1999; Hathaway *et al.*, 2000; Hung and Yang, 2001; Kolen and Hutcheson, 2002). Agrupamentos HARD fornecem uma partição na qual cada objeto do conjunto de dados é atribuído a um e somente um grupo. Agrupamento fuzzy gera uma partição fuzzy que fornece um grau de atribuição a cada padrão em um dado grupo, que dá flexibilidade para expressar que este objeto pertence a mais de um grupo ao mesmo tempo.

Existem duas representações comuns dos objetos nos quais os modelos 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. Agrupamento baseado em dados relacionais é muito útil quando os objetos não podem ser representados por um vetor de valores, quando a medida de distância não tem uma forma definida, etc (Kaufman and Rousseeuw, 1990; Lechevallier, 1974; De Carvalho *et al.*, 2009; Davenport *et al.*, 1989; Hathaway and Bezdek, 1994).

Os mapas auto-organizáveis de Kohonen (Kohonen, 1990) (Self organizing maps - SOM) fazem parte do grupo de modelos de rede neurais não-supervisionadas e de aprendizado competitivo. A rede SOM possui propriedades de agrupamento e visualização. O objetivo destes modelos é encontrar uma estrutura lógica entre os dados fornecidos, não existe uma resposta esperada nem uma ação determinada que deva ser realizada. Para descobrir essa estrutura lógica, o algoritmo se utiliza de interações laterais entre os neurônios formando uma vizinhança. O neurônio que obtiver o melhor valor de similaridade para uma dada entrada é atualizado, da mesma forma neurônios vizinhos

também são atualizados para representar melhor a entrada, resultando em regiões nas quais os neurônios são mais similares entre si.

O mapa auto-organizável pode ser considerado como um algoritmo que mapeia dados de alta dimensionalidade espacial em um espaço de dimensionalidade reduzida, geralmente uma, duas ou três dimensões. Esta projeção habilita o particionamento dos dados em grupos similares e possui a propriedade de preservar a topologia dos dados. A característica mais importante dos mapas auto-organizáveis é a possibilidade de comparar agrupamentos (Badran *et al.*, 2005). Cada objeto é afetado a um grupo e cada grupo é projetado em um nó do mapa. Objetos semelhantes são projetados no mesmo nó. A dissimilaridade entre os objetos projetados aumenta com a distância que separa os nós.

1.1 Motivação

Diversos métodos de análise de dados se baseiam em dados que podem ser descritos por valores reais, ou seja, por vetores em um espaço dimensional fixo e finito. Entretanto, muitos dados do mundo real requerem estruturas mais complexas para que sejam representados adequadamente. Textos, por exemplo, não são numéricos e possuem uma estrutura interna complexa que é difícil de representar em um vetor.

Quando cada objeto da base de dados é descrito por um vetor de valores quantitativos ou qualitativos, temos dados de características, ou seja, que representam propriedades dos objetos. Este tipo de dado pode ser descrito por uma matriz contendo valores de observações, onde as linhas da matriz correspondem aos objetos e as colunas às variáveis. Alternativamente, quando cada par de objetos é representado por uma relação, então temos dados relacionais. O caso mais comum de dados relacionais é quando temos uma matriz de dados de dissimilaridade, por exemplo, $R = [r_{il}]$, onde r_{il} é a dissimilaridade (geralmente uma distância) entre os objetos i e l. Neste caso, as linhas e as colunas da matriz correspondem aos objetos do conjunto de dados.

Agrupamento de dados relacionais é mais utilizado em situações onde os dados não podem ser descritos por características numéricas, também é mais prático quando a distância possui alto grau de complexidade computacional ou quando grupos de objetos similares não podem ser representados eficientemente por um único protótipo. Muitos algoritmos foram adaptados para analisar dados relacionais. (Kaufman and Rousseeuw, 1987) apresenta uma adaptação do k-means para dados relacionais. Ainda, (Golli *et al.*, 2004) apresenta um modelo de mapa auto-organizável por lote baseado em dados relacionais.

Em diversas situações, dados relacionais são descritos, não por uma, mas por múltiplas tabelas de dissimilaridade. Como apontado por (Frigui *et al.*, 2007) muitas aplicações podem se beneficiar de algoritmos de agrupamento baseados em múltiplas matrizes de dissimilaridade. Na categorização de imagens, pode-se ter uma matriz com informações de cor, outra matriz com informação de textura e outra com informação de estrutura.

Porém, diferentes matrizes não são igualmente importantes, algumas podem ser redundantes, outras irrelevantes, ou ainda, podem interferir negativamente na formação dos agrupamentos. Para que haja uma formação adequada dos agrupamentos faz-se necessário o uso de pesos para cada matriz de dissimilaridade, pesos estes que dependem de cada agrupamento. O objetivo da ponderação sobre as matrizes é encontrar graus de relevância e identificar quais características descrevem melhor os dados, tornando o agrupamento mais significativo.

A ponderação de características deriva da seleção de características e tem sido um tópico de pesquisa importante em algoritmo de aprendizado não-supervisionado. Em (Qiang Wang and Huang, 2008), os autores introduzem um algoritmo fuzzy k-means que tem a vantagem de trabalhar com ponderação para objetos e variáveis simultaneamente. (Grozavu *et al.*, 2009) desenvolveu dois modelos usando mapas auto-organizáveis (SOM), que realizam simultaneamente agrupamento e ponderação de variáveis. (Frigui *et al.*, 2007) propõe um algoritmo de agrupamento baseados em múltiplas tabelas de dissimilaridade (CARD) que calcula pesos relacionados à relevância de cada matriz de dissimilaridade sobre cada agrupamento. Em outro trabalho, (Frigui and Nasraoui, 2004) apresenta uma abordagem que realiza agrupamento e ponderação de variáveis simultaneamente.

1.2 Trabalhos Relacionados

(Golli *et al.*, 2004) e (Conan-Guez *et al.*, 2006) propõem uma adaptação dos mapas auto-organizáveis em lote para dados de dissimilaridade. (Frigui *et al.*, 2007) propôs um algoritmo de agrupamento fuzzy para dados relacionais (CARD), que é capaz de particionar objetos levando em conta múltiplas matrizes de dissimilaridade e que ainda calcula pesos que medem a relevância de cada matriz de dissimilaridade para cada um dos grupos. O CARD é baseado em algoritmos de agrupamento fuzzy para dados relacionais bastante conhecidos, o NERF (Hathaway and Bezdek, 1994) e o FANNY (Kaufman and Rousseeuw, 1990). Como citado por (Frigui *et al.*, 2007), diversas aplicações podem se beneficiar de algoritmos de agrupamento de dados relacionais baseados em

múltiplas matrizes de dissimilaridade. No campo de categorização de base de dados de imagem, a relação entre os objetos pode ser descrita por múltiplas matrizes e a medida de dissimilaridade mais efetiva não possui uma forma definida ou não é diferenciável com respeito aos parâmetros do protótipo.

1.3 Objetivos

A maioria dos métodos de análise de dados busca classificar novos objetos com base no conhecimento adquirido pela observação anterior de objetos semelhantes. Métodos de agrupamento, por outro lado, apenas buscam uma estrutura inerente aos dados, agrupando-os de acordo com as características semelhantes entre si. Em diversas situações na área de análise de dados, a representação destes dados é melhor descrita através de múltiplas matrizes de dissimilaridade. Visto isto, é necessário a criação de modelos que sejam capazes de agrupar objetos levando em consideração a descrição desses dados, contidas em múltiplas tabelas de dissimilaridade, simultaneamente. Este trabalho propõe um modelo que seja capaz de tratar múltiplas tabelas de dissimilaridade simultaneamente, que possa, também, aprender e calcular pesos medindo a relevância de cada tabela na formação dos grupos. Além disso, o método aqui proposto possui propriedades de visualização, pois se baseia no algoritmo de mapas auto-organizáveis.

1.4 Organização da Dissertação

2

Mapas auto-organizáveis

2.1 Mapas auto-organizáveis de Kohonen

Os mapas auto-organizáveis de Kohonen (SOM) têm sido bastante utilizados em diversos domínios e têm sido aplicados com sucesso em diversas aplicações. Tornou-se uma ferramenta muito popular usada para visualização de dados de alta dimensionalidade espacial. SOM é capaz de realizar quantização de vetores e/ou agrupamento de dados preservando o ordenamento espacial dos dados. A preservação topológica dos dados é possível através da ordenação dos vetores dos protótipos (também chamados de centros, centróides ou vetores de referência) em um espaço de uma ou duas dimensões. O mapa auto-organizável consiste em neurônios organizados em uma grade , geralmente com baixa dimensionalidade (comumente duas dimensões), chamado de mapa. Formalmente, o mapa é descrito por um grafo (C , Γ). C é um conjunto de m neurônios interconectados por uma topologia discreta definida por Γ. Para cada par de neurônios (c,r) no mapa, $\delta(c,r)$ é definida como a função de distância entre c e r no grafo. Esta distância impõe uma relação de vizinhança entre os neurônios. Cada neurônio c é representado por um protótipo p-dimensional $\mathbf{w}_c = (w_c^1, ..., w_c^p)$, onde p é igual à dimensão dos vetores dos dados de entrada.

Seja $E = \{1, \ldots, n\}$ o conjunto de objetos, onde cada objeto $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})\,(i = 1, \ldots, n)$ pertence a $\mathbb{R}^p$.

O algoritmo

O algoritmo SOM básico executa as seguintes etapas:

1) *Inicialização*.
   Fixe o número m de neurônios (grupos);

Fixe $h_{\delta(j,l)}(0)$ (função de vizinhança inicial, onde $\delta(j,l)$ é uma função de distância fixa entre os neurônios j e l);
Fixe a função kernel K;
Fixe o número de iterações N_{iter};
Fixe $\eta(0)$ (taxa de aprendizado inicial);
Defina $t \leftarrow 1$;
Selecione m protótipos distintos aleatoriamente $\mathbf{w}_c^{(0)} \in E\,(c = 1,\ldots,m)$;
Defina o mapa $L(m,\mathbf{W}^0)$, onde $\mathbf{W}^0 = (\mathbf{w}_1^{(0)},\ldots,\mathbf{w}_m^{(0)})$;

2) *Etapa 1: Amostragem.*
Obtenha um vetor de entrada aleatório $\mathbf{x}_i(t)$;

3) *Etapa 2: Seleção.*
Encontre o melhor neurônio (vencedor) $\mathbf{w}_c(t)$ em relação à distância Euclidiana mínima:

$$f(\mathbf{x}_i(t)) = \min_{1\leq j\leq m} \sum_{k=1}^{p} (x_{ik}(t) - w_{jk}(t))^2;$$

4) *Etapa 3: Modificação do pesos.*
Para todos os neurônios j dentro de um determinado raio de vizinhança do neurônio vencedor $\mathbf{w}_c$, ajuste os pesos de acordo com a expressão:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \eta(t) h_{\delta(\mathbf{w}_c,w_j)}(t)(\mathbf{x}_i(t) - \mathbf{w}_j(t));$$

5) *Atualizando.*
Atualize a taxa de aprendizado $\eta(t)$ e a função de vizinhança $h_{l,j}(t)$

6) *Critério de parada.*
Se $t = N_{Iter}$, pare; senão, vá para 2 (Etapa 1).

2.2 Mapas auto-organizáveis por lote

Esta seção apresenta o mapa auto-organizável por lote introduzido por Kohonen and Somervuo (2002).

Seja $E = \{1,\ldots,n\}$ o conjunto de objetos, onde cada objeto $\mathbf{x}_i = (x_{i1},\ldots,x_{ip})\,(i = 1,\ldots,n)$ pertence a $\mathbb{R}^p$. Cada neurônio do mapa é representado por um protótipo $\mathbf{w}_c = (w_{c1},\ldots,w_{cp})\,(c = 1,\ldots,m)$ que também pertence a $\mathbb{R}^p$.

O algoritmo de treinamento em lote dos mapas auto-organizáveis Kohonen and Somervuo (2002) é um algoritmo iterativo composto de duas etapas (afetação e representação, discutidas posteriormente), onde todo o conjunto de dados (chamado E) é apresentado ao mapa antes que qualquer alteração seja realizada. O algoritmo minimiza a seguinte função objetivo:

$$J = \sum_{i=1}^{n} \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r)) d^2(\mathbf{x}_i, \mathbf{w}_r) \tag{2.1}$$

onde f é a função de alocação e $f(\mathbf{x}_i)$ representa o neurônio do mapa que é associado ao objeto $\mathbf{x}_i$ e $\delta(f(\mathbf{x}_i), r))$ é a distância, no mapa, entre um neurônio r e o neurônio que está alocado ao objeto $\mathbf{x}_i$. Além disso, K^T, parametrizado por T (T significa temperatura) é a função *kernel* de vizinhança que define a região de influência ao redor do neurônio r.

A função objetivo é uma extensão da função objetivo do *k-means*, onde a distância Euclidiana é substituída por uma distância generalizada:

$$d^T(\mathbf{x}_i, \mathbf{w}_{f(\mathbf{x}_i)}) = \sum_{r=1}^{m} K^T(\delta(f^T(\mathbf{x}_i), r)) d^2(\mathbf{x}_i, \mathbf{w}_r) \tag{2.2}$$

onde

$$d^2(\mathbf{x}_i, \mathbf{w}_r) = \sum_{j=1}^{p} (x_{ij} - w_{rj})^2 \tag{2.3}$$

é a distância Euclidiana. Esta distância generalizada é a soma ponderada das distâncias euclidianas entre $\mathbf{x}_i$ e todos os vetores de referência da vizinhança do neurônio $f(\mathbf{x}_i)$, e que leva em conta todos os neurônios do mapa.

Quando T é mantido fixo, a minimização de J é realizada iterativamente em duas etapas: afetação e representação.

Durante a etapa de afetação, os vetores de referência (protótipos) são mantidos fixos. A função objetivo é minimizada de acordo com a função de alocação e cada indivíduo $\mathbf{x}_i$ é associado ao neurônio mais próximo:

$$c = f^T(\mathbf{x}_i) = arg \min_{1 \le r \le m} d^T(\mathbf{x}_i, \mathbf{w}_r) \tag{2.4}$$

Durante a etapa de representação, a função de alocação é mantida fixa. A função objetivo J é minimizada de acordo com a atualização dos protótipos. O protótipo $\mathbf{w}_c$ é atualizado segundo a expressão:

$$\mathbf{w}_c = \frac{\sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), c))\mathbf{x}_i}{\sum_{i=1}^{n} K^T(\delta(f^T(\mathbf{x}_i), c))}. \tag{2.5}$$

O algoritmo

O algoritmo de mapas auto-organizáveis por lote pode ser resumido da seguinte forma:

1) *Inicialização.*
   Fixe o número m de neurônios (grupos);
   Fixe δ; Fixe a função *kernel* K^T;
   Fixe o número de iterações N_{iter};
   Fixe T_{min}, T_{max}; Defina $T \leftarrow T_{max}$; Defina $t \leftarrow 0$;
   Selecione aleatoriamente m protótipos distintos $\mathbf{w}_c^{(0)} \in E\,(c = 1, \ldots, m)$;
   Defina o mapa $L(m, \mathbf{W}^0)$, onde $\mathbf{W}^0 = (\mathbf{w}_1^{(0)}, \ldots, \mathbf{w}_m^{(0)})$;
   Associe cada objeto $\mathbf{x}_i$ ao neurônio mais próximo (grupo) conforme a equação (2.4);

2) *Etapa 1: Representação.*
   Defina $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$;
   A função de alocação é mantida fixa;
   Calcule os protótipos $\mathbf{w}_c^{(t)}\,(c = 1, \ldots, m)$ conforme a equação (2.5);

3) *Etapa 2: Afetação.*
   Os protótipos $\mathbf{w}_c^{(t)}\,(c = 1, \ldots, m)$ são fixos. Associe cada indivíduo $\mathbf{x}_i\,(i = 1, \ldots, n$ ao neurônio mais próximo conforme a equação (2.4);

4) *Critério de parada.*
   Se $T = T_{min}$ então PARE; senão defina $t = t + 1$ e vá para 2 (Etapa 1).

2.3 Mapas auto-organizáveis por lote para dados de dissimilaridade

O algoritmo de mapas auto-organizáveis por lote para dados de dissimilaridade introduzido por Golli *et al.* (2004) deriva do SOM original como descrito acima. A diferença principal está nos dados que serão agrupados. Estes dados são representados por uma relação de dissimilaridade.

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 vetor referência (ou protótipo) $g_c = e_j$, $e_j \in E$. No modelo clássico do SOM cada vetor referência pode assumir qualquer valor no espaço de entrada $\mathbb{R}^p$, nesta abordagem, cada neurônio possui um número finito de representações.

O algoritmo de treinamento em lote é um algoritmo iterativo composto de duas etapas (afetação e representação, discutidas a seguir) onde todo o conjunto de dados (chamado E) é apresentado ao mapa antes que qualquer alteração seja realizada. Durante o aprendizado, a seguinte função objetivo é minimizada:

$$J = \sum_{i=1}^{n} \sum_{r=1}^{m} K^T(\delta(f(e_i), r)) d(e_i, g_r) \tag{2.6}$$

onde é a função de alocação e $f(e_i)$ representa o neurônio do mapa que é associado ao objeto e_i e $\delta(f(e_i), r)$ é a distância no mapa entre um neurônio r e o neurônio que está alocado ao objeto e_i. O conceito de vizinhança é incorporado através de funções *kernel* K que são positivas e tais que $\lim_{|x|\to\infty} K(x) = 0$ Badran *et al.* (2005). Além disso, K^T, parametrizado por T (T significa temperatura) é a função *kernel* de vizinhança que define a região de influência ao redor do neurônio r. Quanto menor o valor de T, menos neurônios irão pertencer à vizinhança de um dado neurônio r.

A função objetivo é uma extensão da função objetivo do *k-means*, onde a distância Euclidiana é substituída por uma distância generalizada:

$$d^T(e_i, g_{f(e_i)}) = \sum_{r=1}^{m} K^T(\delta(f^T(e_i), r)) d(e_i, g_r) \tag{2.7}$$

esta distância generalizada é a soma dos pesos das distâncias euclidianas entre e_i e todos os protótipos da vizinhança do neurônio $f(e_i)$.

Quando T é mantido fixo, a minimização de J é realizada iterativamente em duas etapas: afetação e representação.

Durante a etapa de afetação, a função f associa cada elemento e_i ao neurônio cujo vetor referência é "mais próximo"a e_i e diminui o valor da função objetivo J. Cada indivíduo e_i é associado ao neurônio mais próximo:

$$c = f^T(e_i) = arg \min_{1 \le r \le m} d^T(e_i, g_r) \tag{2.8}$$

Durante a etapa de representação, novos protótipos representando cada grupo são selecionados. O protótipo g_r^* do grupo C_r, que minimiza a função objetivo J é determinado

pela equação:

$$g_r^* = arg\min_{e \in E} \sum_{i=1}^{n} K^T(\delta(f^T(e_i), r)) d^T(e_i, e_r) \tag{2.9}$$

O algoritmo

1. Inicialização.
   Fixe o número m de neurônios (grupos);
   Fixe δ; Fixe a função *kernel* K^T;
   Fixe o número de iterações N_{iter};
   Fixe T_{min}, T_{max}; Defina $T \leftarrow T_{max}$; Defina $t \leftarrow 0$;
   Selecione aleatoriamente m protótipos distintos $g_c^{(0)} \in E(c = 1, \ldots, m)$;
   Defina o mapa $L(m, G^0)$, onde $G^0 = (g_1^{(0)}, \ldots, g_m^{(0)})$;
   Associe cada objeto e_i ao neurônio (grupo) mais próximo conforme a equação 2.8;

2. Etapa 1: Representação.
   Defina $T = T_{max} * (\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$;
   A função de alocação é mantida fixa.
   Selecione os protótipos $g_c^{(t)}(c = 1, \ldots, m)$ conforme a equação 2.9;

3. Etapa 2: Afetação.
   Os protótipos $g_c^{(t)}(c = 1, \ldots, m)$ permanecem fixos. Associe cada indivíduo $e_i(i = 1, \ldots, n)$ ao neurônio mais próximo conforme a equação 2.8;

4. Critério de parada.
   Se $T = T_{min}$ então PARE; senão defina $t = t + 1$ e vá para 2 (Etapa 1).

3

SOM para dados relacionais baseados em múltiplas tabelas de dissimilaridade

3.1 Introdução

Neste capítulo introduzimos um mapa auto-organizável por lote para dados relacionais baseados em múltiplas matrizes de dissimilaridade. O objetivo do algoritmo apresentado é mapear objetos levando em conta suas descrições relacionais dadas por múltiplas matrizes de dissimilaridade. O modelo aqui proposto é um algoritmo iterativo composto por três etapas, são elas: representação, ponderação e afetação, que serão detalhadas posteriormente. É importante notar que por ser um algoritmo por lote, todo o conjunto de dados é apresentado ao mapa antes que quaisquer alterações sejam realizadas.

O algoritmo SOM por lote para dados relacionais baseados em múltiplas tabelas de dissimilaridade deriva do SOM original por lote e do algoritmo de agrupamento para dados relacionais ((De Carvalho *et al.*, 2011)). Esta abordagem utiliza diferentes pesos para as matrizes de dissimilaridade, com o objetivo de ponderar a relevância de cada matriz na formação dos agrupamentos. Os pesos mudam a cada iteração do algoritmo, portanto não são definidos absolutamente, além disso, são diferentes de um agrupamento para outro.

O algoritmo apresentado neste trabalho é capaz de analisar dados levando em consideração a dissimilaridade entre observações. Os dados analisados são matrizes de dissimilaridade contendo a relação entre cada um dos objetos presentes na base de dados. Cada matriz representa uma variável da base e a dissimilaridade é calculada por uma função fixa que é a distância euclidiana entre os objetos. Para encontrar uma partição dos elementos, o método descrito leva em consideração simultaneamente a descrição relacional dos dados dada por múltiplas matrizes de dissimilaridade.

O modelo aqui apresentado utiliza diferentes pesos adaptativos para cada matriz de dissimilaridades. Esses pesos mudam a cada iteração do algoritmo e, além disso, são diferentes de um agrupamento para outro, ou seja, cada matriz possui uma influência diferente sobre a formação de cada agrupamento. O cálculo dos vetores de pesos neste algoritmo foi inspirado pela abordagem utilizada para calcular pesos para cada variável em cada agrupamento no algoritmo de agrupamento dinâmico baseado em distâncias adaptativas ((Diday and Govaert, 1977)).

3.2 O algoritmo

Seja $E = \{e_1, \ldots, e_n\}$ o conjunto de n objetos e p matrizes de dissimilaridade $\mathbf{D}_j = [d_j(e_i, e_l)]\,(j = 1, \ldots, p)$, onde $d_j(e_i, e_l)$ denota a dissimilaridade entra dois objetos e_i e $e_l\,(i, l = 1, \ldots, n)$ na matriz de dissimilaridade $\mathbf{D}_j$.

Uma característica importante do modelo introduzido é que este assume que o protótipo G_k do agrupamento C_k é um subconjunto de cardinalidade fixa $1 \leq q << n$ do conjunto E (por questão de simplicidade, geralmente $q = 1$), isto é, $G_k \in E^{(q)} = \{A \subset E : |A| = q\}$.

O algoritmo busca minimizar uma função objetivo dada por:

$$J = \sum_{e_i \in E} \sum_{l=1}^{c} K^T(\delta(\chi(e_i), l)) D_{\lambda_l}(e_i, G_l) \tag{3.1}$$

onde D_{λ_l} é a dissimilaridade global entre um objeto $e_i \in P_l$ e o protótipo do agrupamento $G_l \in E^{(q)}$, parametrizada pelo vetor de pesos $\lambda_l = (\lambda_{l1}, \ldots, \lambda_{lp})$ das matrizes de dissimilaridade $\mathbf{D}_j$ no agrupamento $P_l\,(l = 1, \ldots, c)$.

De acordo com a função de alocação, existem diferentes algoritmos SOM por lote. Neste trabalho consideramos funções de alocação com pesos para cada matriz de dissimilaridade, sendo estes pesos estimados localmente ou globalmente.

Através da função de alocação com pesos estimados localmente, é possível comparar agrupamentos e seus protótipos usando diferentes medidas associadas a cada agrupamento que muda a cada iteração, isto é, a distância não é determinada absolutamente e é diferente de um agrupamento para outro.

A função de alocação parametrizada pelo vetor de pesos $\lambda_l = (\lambda_{l1}, \ldots, \lambda_{lp})$, onde $\lambda_{lj} > 0$ e $\prod_{j=1}^{p} \lambda_{lj} = 1$ e associada ao agrupamento $P_l\,(l = 1, \ldots, c)$ é definida pela seguinte expressão:

$$D_{\lambda_l}(e_i, G_l) = \sum_{j=1}^{p} \lambda_{lj} D_j(e_i, G_l) = \sum_{j=1}^{p} \lambda_{lj} \sum_{e \in G_l} d_j(e_i, e) \tag{3.2}$$

onde $D_j(e_i, G_l) = \sum_{e \in G_l} d_j(e_i, e)$ representa a dissimilaridade local entre um exemplo $e_i \in P_l$ e o protótipo do agrupamento $G_l \in E^{(q)}$ na matriz $\mathbf{D}_j\,(j = 1, \ldots, p)$.

O princípio da função de alocação definida por um vetor de pesos estimado globalmente para todos os agrupamentos é que há uma distância para comparar os agrupamentos e seus protótipos, distância esta que muda a cada iteração, mas é a mesma para todos os agrupamentos.

A função de alocação parametrizada pelo vetor de pesos boldmathλ_l $=\lambda = (\lambda_1, \ldots, \lambda_p)\,(l = 1, \ldots, c)$, onde $\lambda_j > 0$ and $\prod_{j=1}^{p} \lambda_j = 1$, é expressa da seguinte forma:

$$D_{\lambda}(e_i, G_l) = \sum_{j=1}^{p} \lambda_j D_j(e_i, G_l) = \sum_{j=1}^{p} \lambda_j \sum_{e \in G_l} d_j(e_i, e) \tag{3.3}$$

onde $D_j(e_i, G_l)$, novamente representa a dissimilaridade local entre um exemplo $e_i \in P_l$ e o protótipo do agrupamento $G_l \in E^{(q)}$ na matriz $\mathbf{D}_j\,(j = 1, \ldots, p)$.

Quando T é fixo, a minimização da função J é realizada iterativamente em três etapas: representação, ponderação e afetação.

3.2.1 Etapa de representação: determinação dos melhores protótipos

Na etapa de representação, a partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e os vetores de pesos ${\lambda_l}^{(t-1)}\,(r = 1, \ldots, c)$ são fixos. A função objetivo J é minimizada de acordo com os protótipos.

Proposição 3.2.1. *O protótipo é atualizado de acordo com a função de alocação usada:*

1. *Se a função de alocação é definida pela equação 3.2, calcule o protótipo* $G_l^{(t)} = G^* \in E^{(q)}$ *do agrupamento* $P_l^{(t-1)}$ $(l = 1, \ldots, c)$ *segundo a expressão:*

$$G^* = argmin_{G \in E^{(q)}} \sum_{e_i \in E} K^T(\delta(\chi^{(t-1)}(e_i), l)) \sum_{j=1}^{p} \lambda_{lj}^{(t-1)} \sum_{e \in G} d_j(e_i, e) \tag{3.4}$$

2. *Se a função de alocação é definida pela equação 3.3, calcule o protótipo* $G_l^{(t)} =$

$G^* \in E^{(q)}$ *do agrupamento* $P_l^{(t-1)}$ $(r = 1, \ldots, c)$ *de acordo com a equação seguinte:*

$$G^* = argmin_{G \in E^{(q)}} \sum_{e_i \in E} K^T(\delta(\chi^{(t-1)}(e_i), l)) \sum_{j=1}^{p} \lambda_j^{(t-1)} \sum_{e \in G} d_j(e_i, e) \tag{3.5}$$

3.2.2 Etapa de ponderação: definição dos melhores vetores de pesos

Durante a etapa de ponderação, a partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e os protótipos $G_l^{(t)} \in E^{(q)}$ $(l = 1, \ldots, c)$ são mantidos fixos. A função objetivo J é minimizada de acordo com os vetores de pesos.

Proposição 3.2.2. *Os vetores de pesos são atualizados de acordo com a função de alocação utilizada:*

1. *Se a função de alocação é definida pela equação 3.2, os vetores de pesos* $\lambda_l{}^{(t)} = (\lambda_{l1}^{(t)}, \ldots, \lambda_{lp}^{(t)})$ $(l = 1, \ldots, c)$, *com* $\lambda_{lj}^{(t)} > 0$ *e* $\prod_{j=1}^{p} \lambda_{lj}^{(t)} = 1$, *têm seus pesos* $\lambda_{lj}^{(t)}$ $(j = 1, \ldots, p)$ *calculados de acordo com a equação:*

$$\lambda_{lj}^{(t)} = \frac{\left\{ \prod_{h=1}^{p} \left[\sum_{e_i \in E} K^T(\delta(\chi^{(t-1)}(e_i), l)) \sum_{e \in G_l^{(t)}} d_h(e_i, e) \right] \right\}^{\frac{1}{p}}}{\left[\sum_{e_i \in E} K^T(\delta(\chi^{(t-1)}(e_i), l)) \sum_{e \in G_l^{(t)}} d_j(e_i, e) \right]} \tag{3.6}$$

2. *Se a função de alocação é definida pela equação 3.3, o vetor de pesos* $\lambda^{(t)} = (\lambda_1^{(t)}, \ldots, \lambda_p^{(t)})$, *com* $\lambda_j^{(t)} > 0$ *e* $\prod_{j=1}^{p} \lambda_j^{(t)} = 1$, *tem seus pesos* $\lambda_j^{(t)}$ $(j = 1, \ldots, p)$ *calculados segundo a equação:*

$$\lambda_j^{(t)} = \frac{\left\{ \prod_{h=1}^{p} \left[\sum_{r=1}^{c} \left(\sum_{e_i \in E} K^T(\delta(\chi^{(t-1)}(e_i), l)) \sum_{e \in G_l^{(t)}} d_h(e_i, e)] \right) \right] \right\}^{\frac{1}{p}}}{\sum_{r=1}^{c} \left[\sum_{e_i \in E} K^T(\delta(\chi^{(t-1)}(e_i), l)) \sum_{e \in G_l^{(t)}} d_j(e_i, e) \right]} \tag{3.7}$$

3.2.3 Etapa de afetação: definição da melhor partição

Durante a etapa de afetação os protótipos $G_l^{(t)} \in E^{(q)}\,(l = 1,\ldots,c)$ e os vetores de pesos ${\lambda_l}^{(t-1)}\,(r = 1,\ldots,c)$ são mantidos fixos. A função objetivo J é minimizada de acordo com a função de afetação.

Proposição 3.2.3. *Cada exemplo $e_i \in E$ é alocado ao neurônio mais próximo de acordo com a função de alocação utilizada:*

1. *Se a função de alocação é definida pela equação (3.2), alocar o exemplo $e_i \in E$ no agrupamento C_m segundo a equação:*

$$m = (\chi^{(t)}(e_i))^{(t)} = argmin_{1\leq r\leq c} \sum_{l=1}^{c} K^T(\delta(r,l)) \sum_{j=1}^{p} \lambda_{lj}^{(t)} \sum_{e\in G_l^{(t)}} d_j(e_i,e) \qquad (3.8)$$

2. *Se a função de alocação é definida pela equação (3.3), alocar o exemplo $e_i \in E$ no agrupamento C_m segundo a expressão:*

$$m = (\chi^{(t)}(e_i))^{(t)} = argmin_{1\leq r\leq c} \sum_{l=1}^{c} K^T(\delta(r,l)) \sum_{j=1}^{p} \lambda_{j}^{(t)} \sum_{e\in G_l^{(t)}} d_j(e_i,e) \qquad (3.9)$$

3.2.4 O algoritmo

O algoritmo SOM em lote para dados relacionais baseados em múltiplas matrizes de dissimilaridade pode ser resumido como segue:

1. Inicialização
   Fixe o número c de agrupamentos;
   Fixe a cardinalidade $1 \leq q << n$ dos protótipos $G_l\,(l = 1,\ldots,c)$;
   Fixe δ; Fixe a função kernel K
   Fixe o número de iterações N_{iter}
   Fixe T_{min}, T_{max}; Determine $T \leftarrow T_{max}$; Determine $t \leftarrow 0$;
   Selecione aleatoriamente c protótipos distintos $G_l^{(0)} \in E^{(q)}\,(l = 1,\ldots,c)$;
   Determine ${\lambda_l}^{(0)} = (1,\ldots,1)\,(l = 1,\ldots,c)$;
   Determine o mapa $L(c,\mathbf{G}^0)$, onde $\mathbf{G}^0 = (G_1^{(0)},\ldots,G_c^{(0)})$
   Aloque cada objeto e_i ao protótipo mais próximo para obter a partição $P^{(0)} = (P_1^{(0)},\ldots,P_c^{(0)})$ de acordo com as equações (3.8) e (3.9)

2. *Etapa de representação: cálculo dos melhores protótipos.*
   Determine $t = t + 1$;
   Calcule $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$
   A partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e $\lambda_l{}^{(t-1)}$ $(l = 1, \ldots, c)$ são mantidos fixos.
   Calcule o protótipo $G_l^{(t)} = G^* \in E^{(q)}$ do agrupamento $P_l^{(t-1)}$ $(l = 1, \ldots, c)$ de acordo com as equações (3.4) e (3.5)

3. *Etapa de ponderação: cálculo dos melhores pesos.*
   A partição $P^{(t-1)} = (P_1^{(t-1)}, \ldots, P_c^{(t-1)})$ e os protótipos $G_l^{(t)} \in E^{(q)}$ $(l = 1, \ldots, c)$ são mantidos fixos.
   Calcule os vetores de pesos λ_l $(l = 1, \ldots, c)$ de acordo com as equações (3.6) e (3.7)

4) *Etapa de afetação: definição da melhor partição.*
   Os protótipos $G_l^{(t)} \in E^{(q)}$ $(l = 1, \ldots, c)$ e $\lambda_l{}^{(t)}$ $(l = 1, \ldots, c)$ são mantidos fixos
   $P^{(t)} \leftarrow P^{(t-1)}$

   para $i = 1$ até n faça
   encontre o agrupamento $C_{m^*}^{(t)}$ ao qual e_i pertence
   encontre o agrupamento vencedor $C_m^{(t)}$ segundo as equações (3.8) e (3.9)
   se $m^* \neq m$
   $C_m^{(t)} \leftarrow C_m^{(t)} \cup \{e_i\}$
   $C_{m^*}^{(t)} \leftarrow C_m^{(t)} \setminus \{e_i\}$

4) *Critério de parada.*
   Se $T = T_{min}$ (ou se $t = N_{iter} - 1$) então PARE; senão vá para 2 (Etapa de representação).

4 Resultados

4.1 Base de dados Íris

Esta base de dados consiste em três tipos (classes) de íris de plantas: iris setosa, iris versicolour and iris virginica. As três classes possuem 50 instâncias (objetos) cada. Uma das classes é linearmente separada das outras duas, estas últimas não são linearmente separadas uma da outra. Cada objeto é descrito por quatro atributos de valores reais: (1) comprimento da sépala, (2) largura da sépala, (3) comprimento da pétala e (4) largura da pétala.

Os três algoritmos de agrupamento (SOM em lote baseado em uma única tabela de dissimilaridade, SOM em lote baseado em múltiplas tabelas de dissimilaridade com ponderação local e sua variante com ponderação global) foram aplicados às matrizes de dissimilaridade representando esta base de dados.

4.2 Base de dados E.coli

4.3 Base de dados Thyroid

4.4 Base de dados Wine

Tabela 4.1 Base de dados Íris: índices *CR*, $F-measure$, e *OERC*

Índices	T_{max}	B-SOM	AB-SOM	global AB-SOM
CR	6	0.4017	0.4847	0.4599
	7	0.3999	0.5067	0.5157
	9	0.3979	0.3978	0.4913
	16	0.3958	0.4653	0.3889
$F-$ *measure*	6	0.4931	0.5785	0.5501
	7	0.5229	0.6071	0.5575
	9	0.5394	0.5383	0.5490
	16	0.5340	0.5742	0.5250
OERC	6	2.67%	4.00%	4.67%
	7	2.67%	4.67%	4.00%
	9	4.67%	3.33%	5.49%
	16	2.00%	4.67%	3.33%

Tabela 4.2 Base Íris: Matrizes relevantes ($T_{max} = 16$)

Modelo	Matriz mais importante	Matriz menos importante
AB-SOM	3-Petal length	1-Sepal length
global AB-SOM	3-Petal length	2-Sepal width

Tabela 4.3 Base Íris: matriz de confusão do algoritmo AB-SOM para mútiplas tabelas de dissimilaridade ($T_{max} = 6$)

	Classes		
Agrupamentos	1-Iris setosa	2-Iris versicolour	3-Iris virginica
0,0	0	16	0
0,1	0	12	0
0,2	0	1	10
0,3	0	4	0
0,4	0	0	12
0,5	0	0	3
0,6	0	0	0
0,7	38	0	0
1,0	0	2	10
1,1	0	9	0
1,2	0	3	0
1,3	0	0	6
1,4	0	0	4
1,5	0	3	3
1,6	0	0	2
1,7	12	0	0

Tabela 4.4 Base Íris: Matriz de pesos final do algoritmo AB-SOM para múltiplas tabelas de dissimilaridade ($T_{max} = 6$)

	Matriz			
Agrupamentos	1-Sepal length	2-Sepal width	3-Petal length	4-Petal width
0,0	0.5878	0.5075	**2.1617**	**1.5507**
0,1	0.1438	0.7925	**1.5381**	**5.7034**
0,2	0.1193	**47.9332**	0.5631	0.3104
0,3	0.7568	0.9139	**1.7698**	0.8169
0,4	0.3581	0.3508	**3.3944**	**2.3451**
0,5	**1.7980**	0.2266	**2.7677**	0.8866
0,6	**1.0037**	0.3404	**1.6919**	**1.7297**
0,7	0.2591	0.0672	**5.2699**	**10.8944**
1,0	0.4403	0.3797	**4.1288**	**1.4484**
1,1	0.2676	0.2180	**4.7063**	**3.6428**
1,2	0.1780	**2.1141**	**1.1214**	**2.3694**
1,3	**1.7573**	0.0686	**3.2489**	**2.5531**
1,4	0.3215	0.1939	**4.4941**	**3.5684**
1,5	**1.4477**	0.1896	**1.0308**	**3.5350**
1,6	**2.0289**	0.7747	0.6483	0.9812
1,7	0.4077	0.1109	**5.3766**	**4.1132**

Tabela 4.5 Base E.coli: índices *CR*, $F-measure$, e *OERC*

Índices	T_{max}	B-SOM	AB-SOM	global AB-SOM
CR	4	0.3035	0.3213	0.2739
	4.5	0.3462	0.2899	0.3165
	6	0.3264	0.2867	0.3555
	10	0.3303	0.2998	0.2925
$F-measure$	4	0.4473	0.4629	0.4082
	4.5	0.5486	0.4153	0.4680
	6	0.4631	0.4036	0.5265
	10	0.5187	0.4385	0.4460
OERC	4	21.43%	17.56%	24.40%
	4.5	16.37%	25.30%	20.54%
	6	16.37%	24.11%	14.88%
	10	15.77%	24.70%	24.40%

Tabela 4.6 Base E.coli: Matrizes relevantes ($T_{max} = 4$)

Modelo	Matriz mais importante	Matriz menos importante
AB-SOM	5	3
global AB-SOM	4	3

Tabela 4.7 Base E.coli: matriz de confusão do algoritmo AB-SOM global para múltiplas tabelas de dissimilaridade ($T_{max} = 10$)

	Classes							
Agrupamentos	1	2	3	4	5	6	7	8
0,0	0	18	0	0	16	1	0	0
0,1	0	10	0	1	24	0	0	0
0,2	0	6	1	0	17	1	0	0
0,3	6	0	0	0	1	4	0	0
0,4	22	0	0	0	2	0	0	0
1,0	1	1	0	1	6	5	5	2
1,1	0	0	0	0	10	0	0	0
1,2	10	0	0	0	0	1	0	0
1,3	15	0	0	0	0	1	0	0
1,4	22	0	0	0	0	0	0	0
2,0	2	0	1	0	0	29	0	9
2,1	0	0	0	0	0	10	0	9
2,2	6	0	0	0	1	0	0	0
2,3	25	0	0	0	0	0	0	0
2,4	34	0	0	0	0	0	0	0

Tabela 4.8 Base E.coli: Matriz de pesos final do algoritmo AB-SOM global para múltiplas tabelas de dissimilaridade ($T_{max} = 10$)

Matriz				
1	2	3	4	5
0.7636	0.7101	0.4876	**2.1144**	**1.7887**

Tabela 4.9 Base de dados Thyroid: índices *CR*, $F-measure$ e *OERC*

Índices	T_{max}	B-SOM	AB-SOM	AB-SOM global
CR	4	0.3662	0.4539	0.5700
	5	0.3642	0.3689	0.3604
	6	0.2618	0.3183	0.5882
	10	0.3784	0.3660	0.3654
$F-$ *measure*	4	0.4961	0.5841	0.5978
	5	0.4870	0.4788	0.4678
	6	0.4019	0.4850	0.6401
	10	0.5023	0.5048	0.4885
OERC	4	9.77%	3.25%	4.65%
	5	9.30%	5.58%	4.19%
	6	11.16%	4.19%	7.44%
	10	9.30%	3.72%	5.12%

Tabela 4.10 Base de dados Thyroid: Matrizes ($T_{max} = 4$)

Modelo	Matriz mais importante	Matriz menos importante
AB-SOM	4-TSH	1-T3-resin uptake test
global AB-SOM	5-maximal absolute difference in TSH	1-T3-resin uptake test

Tabela 4.11 Base de dados Wine: índices *CR*, $F-measure$ e *OERC*

Índices	T_{max}	B-SOM	AB-SOM	global AB-SOM
CR	4	0.2775	0.3560	0.3498
	5	0.2857	0.3449	0.3248
	6	0.2967	0.3589	0.3489
	10	0.2940	0.3773	0.3339
$F-$ *measure*	4	0.4163	0.4897	0.4737
	5	0.4286	0.4735	0.4357
	6	0.4343	0.5245	0.4801
	10	0.4427	0.5332	0.4637
OERC	4	26.40%	7.30%	2.25%
	5	26.97%	3.37%	4.49%
	6	26.97%	4.49%	5.06%
	10	26.97%	6.17%	6.74%

Tabela 4.12 Base de dados Wine: Matrizes relevantes $T_{max} = 4$

Modelo	Matriz mais importante	Matriz menos importante
AB-SOM	2	3
global AB-SOM	7	3

Referências Bibliográficas

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

Bezdek, J. C. (1981). Pattern recognition with fuzzy objective function algorithms. *Plenum Press, New York*.

Chavent, M. (1998). A monotetic clustering method. *Pattern Recognition Letters*, **19**, 989–996.

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

Davenport, J., Hathaway, R., and Bezdek, J. (1989). Relational duals of the c-means algorithms. *Pattern Recognition*, **22**, 205–212.

De Carvalho, F., M.Csernel, and Lechevallier, Y. (2009). Clustering constrained symbolic data. *Pattern Recognition Letters*, **30 (11)**, 1037–1045.

De Carvalho, F., Lechevallier, Y., and de Melo, F. (2011). Partitioning hard clustering algorithms based on multiple dissimilarity matrices. *Pattern Recognition*.

Diday, E. and Govaert, G. (1977). Classification automatique avec distances adaptatives. *R.A.I.R.O. Informatique Comput. Sci.*, **11**, 329–349.

Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, **21**, 768–780.

Frigui, H. and Nasraoui, O. (2004). Unsupervised learning of prototypes and attribute weights. *Pattern Recognition*, **37**, 567–581.

Frigui, H., Hwanga, C., and Rhee, F. C.-H. (2007). Clustering and aggregation of relational data with applications to image database categorization. *Pattern Recognition*, (40 (11)), 3053–3068.

Golli, A. E., Conan-Guez, B., and Rossi, F. (2004). A self organizing map for dissimilarity data. *IFCS 2004 Proceedings*.

Gowda, K. and Krishna, G. (1978). Disaggregative clustering using the concept of mutual nearest neighborhood. *IEEE Transactions on Systems, Man, and Cybernetics*, **8**, 888–895.

Grozavu, N., Bennani, Y., and Lebbah, M. (2009). From variable weighting to cluster characterization in topographic unsupervised learning. *Proceedings of the International Joint Conference on Neural Networks*, pages 1005–1010.

Guenoche, A., Hansen, P., and Jaumard, B. (1991). Efficient algorithms for divisive hierarchical clustering. *Journal of Classification*, **8**, 5–30.

Guha, S., Rastogi, R., and Shim, K. (1998). Cure: An efficient clustering algorithm for large databases. In *Proc. ACM SIGMOD Int. Conf. Management of Data*, pages 73–84.

Guha, S., Rastogi, R., and Shim, K. (2000). Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, **25 (5)**, 345–366.

Hansen, P. and Mladenoviae, N. (2001). J-means: A new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, **34**, 405–413.

Hathaway, R. and Bezdek, J. (1994). Nerf c-means: non-euclidean relational fuzzy clustering. *Pattern Recognition*, **27 (3)**, 429–437.

Hathaway, R., Bezdek, J., and Hu, Y. (2000). Generalized fuzzy c-means clustering strategies using lp norm distances. *IEEE Transactions on Fuzzy Systems*, **8 (5)**, 576–582.

Hoeppner, F., Klawonn, F., and Kruse, R. (1999). Fuzzy cluster analysis: Methods for classification, data analysis, and image recognition. *Wiley, New York*.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, **2**, 283–304.

Hung, M. and Yang, D. (2001). An efficient fuzzy c-means clustering algorithm. *Proc. IEEE Int. Conf. Data Mining*, pages 225–232.

Jain, A. K., Murty, M., and Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, **31 (3)**, 264–323.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2000). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions in Pattern Analysis Machine Intelligence*, **24 (7)**, 881–892.

Karypis, G., Han, E., and Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, **32 (8)**, 68–75.

Kaufman, L. and Rousseeuw, P. (1990). Finding groups in data. *Wiley, New York*.

Kaufman, L. and Rousseeuw, P. J. (1987). Clustering by means of medoids. *Statistical data analysis based on the L1-norm and related methods*, pages 405–416.

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

Kohonen, T. and Somervuo, P. (2002). How to make large self-organizing maps for nonvectorial data. *Neural Networks*, **15**, 945–952.

Kolen, J. and Hutcheson, T. (2002). Reducing the time complexity of the fuzzy c-means algorithm. *IEEE Transactions on Fuzzy Systems*, **10 (2)**, 263–267.

Lance, G. and Williams, W. (1968). Note on a new information statistic classification program. *The Computer Journal*, **11**, 195–197.

Lechevallier, Y. (1974). Optimisation de quelques criteres en classification automatique et application a l'etude des modifications des proteines seriques en pathologie clinique. *ThÃ¨se de 3eme cycle. Universite Paris-VI*.

Qiang Wang, Y. Y. and Huang, J. Z. (2008). Fuzzy k-means with variable weighting in high dimensional data analysis. *The Ninth International Conference on Web-Age Information Management*, pages 365–372.

Sneath, P. and Sokal, R. (1973). Numerical taxonomy. *Freeman, San Francisco*.

Su, M. and Chou, C. (2001). A modified version of the k-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23 (6)**, 674–680.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, **16 (3)**, 645–678.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD Conf. Management of Data*, pages 103–114.