UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
Departamento de Computação

Aprendizado não supervisionado em domínios *fuzzy* – algoritmo *fuzzy c-means*

Frank Sussumu Yonamine
Lucia Specia
Veronica Oliveira de Carvalho
Maria do Carmo Nicoletti

São Carlos, 2002

Índice

Figuras

Tabelas

Introdução

A tarefa de agrupar ou classificar objetos em categorias é uma das atividades mais comuns e primitivas do homem e vem sendo intensificada em função do grande volume de informações disponíveis atualmente, sobre as mais diversas áreas (Backer, 1995).

Para realizar essa tarefa, pode ser empregado um mecanismo chamado análise de *cluster*, ou clusterização, o qual é definido como um processo pelo qual procura-se classificar objetos em categorias. Esse processo pode ser utilizado em várias aplicações, por exemplo, no reconhecimento de padrões, no processamento de imagens, na construção de taxonomias, na classificação de documentos para a recuperação de informações, em agrupamentos sociais baseados em vários critérios, na representação e compressão de base de dados, etc.

O problema de clusterização pode ser tratado segundo diferentes abordagens, entre elas, a abordagem convencional (*crisp*), na qual cada objeto deve ser classificado única e totalmente em uma determinada categoria, e a abordagem *fuzzy*, mais flexível, na qual um objeto pode ser classificado em várias categorias, com diferentes graus de associação a cada uma delas. O uso da teoria de conjuntos *fuzzy* torna-se conveniente, uma vez que grande parte das categorias comumente encontradas e empregadas na clusterização possui limites vagos.

No caso específico da clusterização *fuzzy* (ou *fuzzy clustering*), diferentes métodos podem ser utilizados, por exemplo, o algoritmo *fuzzy c-means*, criado por J. C. Bezdek e descrito neste trabalho.

1 Clusterização

A **clusterização** é um método para a análise exploratória de dados utilizado para auxiliar a resolução de problemas de classificação (Backer, 1995). O objetivo do processo de clusterização é agrupar um conjunto de dados (ou objetos[1]) em diferentes grupos; esse agrupamento é tal que o grau de associação entre elementos do mesmo grupo é alto e entre elementos de grupos diferentes é baixo. Procura-se encontrar a estrutura intrínseca dos dados, organizando-os em grupos (também chamados classes ou *clusters*). O uso desse processo é apropriado quando se conhece pouco ou nada sobre a estrutura de um conjunto de dados.

A partir de um conjunto finito de objetos ou padrões X, cada um deles descritos por um conjunto de pares **atributo-valor_atributo**, um algoritmo de clusterização deve atribuir rótulos aos objetos que identifiquem subgrupos naturais no conjunto. Isso é feito por meio da introdução de uma medida de distância (ou de similaridade) entre os objetos. Usando essa medida, determina-se um critério que expressa a idéia da clusterização, isto é, que particiona o conjunto de objetos em c *clusters* individuais e homogêneos, nos quais elementos de um *cluster* são tão similares entre si quanto possível, e tão diferentes quanto possível, dos elementos dos outros *clusters*. Segundo Brailovsky (1991)[2], a minimização desse critério resulta em *clusters* ótimos.

No método descrito neste trabalho, o número de classes c deve ser definido previamente, no entanto, outras abordagens permitem derivar esse número a partir da imposição de restrições matemáticas ou físicas ao conjunto de dados, como os métodos descritos em Gath e Geva (1992); Tamura et al. (1971); e Zahid et al. (2001).

A clusterização é considerada uma técnica de aprendizado de máquina não supervisionado, onde são aprendidas as classes de cada um dos elementos a partir da descrição de cada um deles como um vetor de pares atributo-valor_atributo. Os algoritmos de clusterização tentam particionar o conjunto de objetos baseados em certas suposições e/ou critérios; conseqüentemente, as saídas do algoritmo podem ou não produzir interpretações significativas e úteis da estrutura nos dados.

A clusterização pode ser realizada de acordo com a abordagem clássica (*crisp*), na qual cada elemento pertence totalmente a uma única classe, ou de acordo com abordagens alternativas, como a *fuzzy*, onde um elemento pode pertencer a várias classes, com diferentes valores de pertinência. Segue um exemplo de aplicação da clusterização clássica.

Considere X = {bob, lia, ari, ana} um conjunto de elementos descritos por seis pares atributo-valor_atributo, como mostra a Tabela 1.

Tabela 1. Quatro objetos descritos por seis pares atributo-valor_atributo

	Olho	***Cabelo***	***Tipo de sangue***	***Sexo***	***Idade***	***Estado civil***
bob	azul	castanho	a	m	44	c
lia	castanho	castanho	a	f	42	c
ari	azul	castanho	o	m	21	s
ana	azul	castanho	b	f	18	s

[1] Um objeto em um espaço n-dimensional pode ser caracterizado como um vetor de n pares atributo-valor_atributo.

[2] Brailovsky, V. L. A Probabilistic Approach to Clustering. In *Pattern Recognition Letters*, vol. 12, pp. 193-198, 1991. Apud (Backer, 1995).

Dependendo do critério selecionado para particionar o conjunto X, pode-se obter diferentes agrupamentos de indivíduos de X, representando diferentes *clusters*, como mostra a Figura 1.

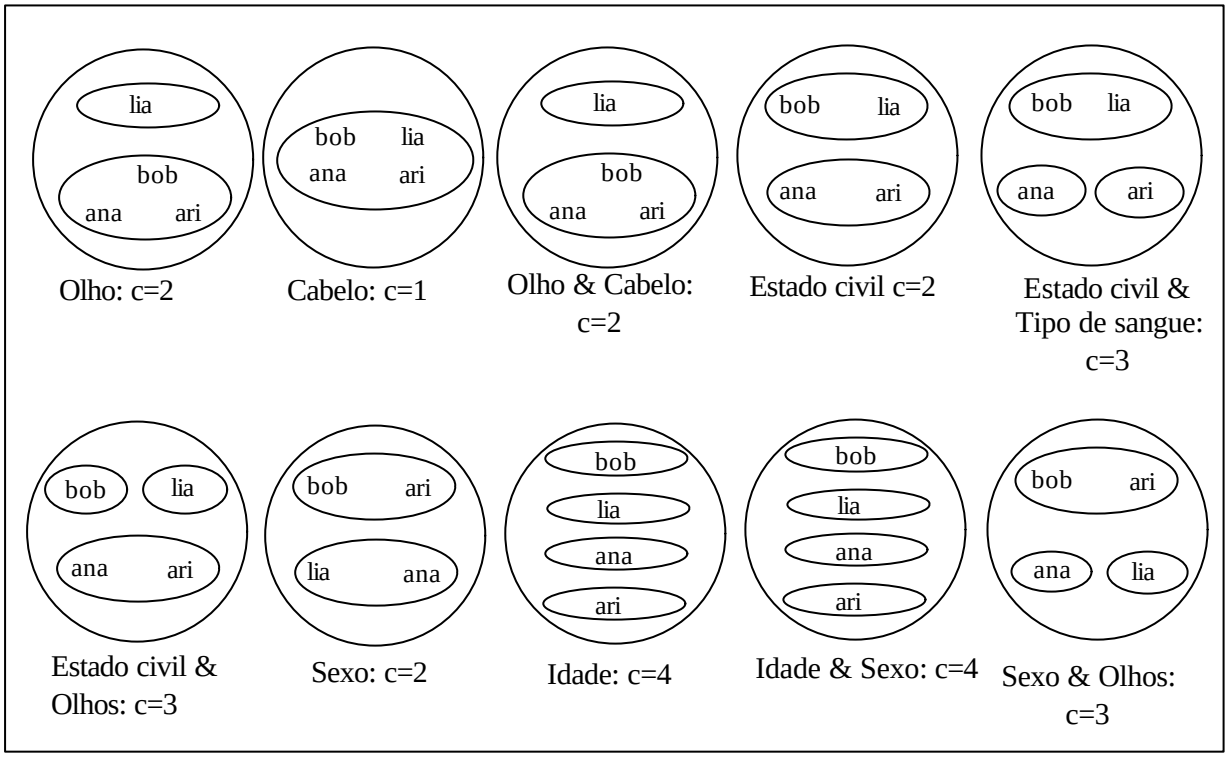

Figura 1. Diferentes agrupamentos dos elementos do conjunto {bob, lia, ari, ana}

2 Clusterização *Fuzzy*

Como visto anteriormente, dado um conjunto finito de elementos X, o problema de clusterização em X consiste em atribuir classes aos elementos de maneira que essas classes identifiquem subgrupos naturais em X. Na análise de *cluster* clássica (*crisp*), essas classes devem formar uma partição[3] de X; indivíduos que pertencem ao mesmo bloco (classe) da partição estão totalmente relacionados (são indistingüíveis) e indivíduos que pertencem a blocos distintos não estão relacionados. Em outras palavras, indivíduos estão totalmente relacionados entre si ou não estão relacionados.

Em muitas aplicações práticas, no entanto, uma partição *crisp* pode ser muito restritiva e inviável, devido, muitas vezes, à imprecisão ou à não completeza dos dados. Segundo Bezdek e Pal (1992), a imprecisão nos dados pode surgir de diversas fontes. Por exemplo, erros em instrumentos ou ruídos no experimento podem levar a valores parcialmente confiáveis de determinados atributos. Em alguns casos, o custo envolvido na extração de valores muito precisos de um atributo pode ser alto. Em outros casos, pode ser difícil decidir quais são os atributos mais relevantes que caracterizam um elemento. Por essas razões, torna-se conveniente o uso de variáveis lingüísticas e limitações para descrever os valores de atributos, em vez de tentar fornecer uma

[3] Uma partição P de um conjunto X é uma família de subconjuntos distintos e não vazios de X, tal que $P(X) = \{X_i \mid i \in I, X_i \subseteq X\}$, onde $X_i \neq \emptyset$, $X_i \cap X_j \neq \emptyset$ para $i, j \in I, i \neq j$ e $\bigcup_{i \in I} X_i = X$.

representação numérica exata para os dados com valores incertos dos atributos. Isso pode ser feito por meio do uso da Teoria de Conjuntos *Fuzzy* (TCF) para representar valores imprecisos.

A incerteza na classificação de elementos pode provir também da sobreposição de várias classes. Nas técnicas de classificação convencionais, geralmente assume-se que um elemento pertence a apenas uma classe, o que nem sempre se verifica em domínios de dados reais, nos quais elementos pertencem a mais de uma classe, com diferentes graus.

Uma alternativa viável para a classificação de informações imprecisas em grupos é a utilização da TCF. Segundo Klir e Yuan (1995), a TCF pode ser utilizada em pelo menos dois níveis no problema de clusterização: 1) no nível de atributos, para representar os elementos do conjunto como um vetor de graus de pertinência, sendo que cada um desses graus representa o grau de "posse" do atributo em questão, por parte desses elementos; e 2) no nível de classificação, para representar a pertinência desses elementos às classes, bem como para prover uma estimativa das informações incompletas em termos de valores de pertinência.

Com a utilização da abordagem *fuzzy*, o problema passa então a ser caracterizado como um problema de clusterização *fuzzy*, cujo objetivo é a obtenção de uma **partição *fuzzy*** ou então de uma **pseudopartição *fuzzy*** em um conjunto de dados X. Em ambos os casos, permite-se diferentes graus de relacionamento entre elementos do conjunto, sendo que um elemento pode pertencer a mais de uma classe.

Pseudopartições *fuzzy* (também chamadas de *c-partitions fuzzy*, onde c representa o número de classes da partição) diferem das partições *fuzzy* regulares no sentido de que estas últimas são obtidas a partir de associações de relações de equivalência *fuzzy*, o que não ocorre com as primeiras (Klir e Yuan, 1995). O método apresentado neste trabalho, descrito na Seção 3, gera somente pseudopartições *fuzzy*. A seguir, é descrito um exemplo segundo o qual se pode verificar os diferentes resultados decorrentes da aplicação das abordagens de clusterização *crisp* e *fuzzy*.

2.1 Exemplo de Clusterização *Fuzzy*

As Figuras 2 e 3 (adaptadas de Bezdek e Pal, 1992, p. 15) mostram o problema da clusterização graficamente. Na primeira, os objetos do conjunto não estão classificados. Na segunda, estão classificados de acordo com alguma técnica de clusterização *crisp*.

Nesse exemplo, o conjunto universo é um conjunto de objetos X={o1,o2,…,o17,o?} e o objetivo é classificar esses objetos como um dos três tipos de frutas, a saber, M = maçã, L= laranja, P = pêra. Note que, na Figura 3, os rótulos atribuídos aos objetos são *crisp*, sendo que o objeto de forma elíptica rotulado como "o?" representa uma anomalia nos dados.

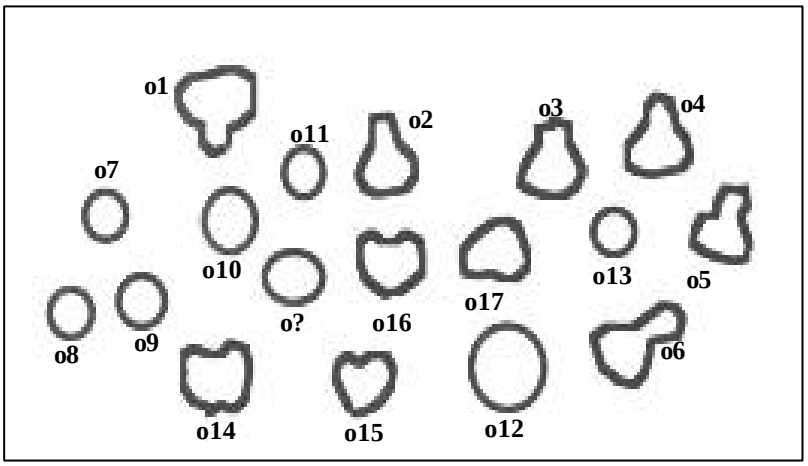

Figura 2. O problema de clusterização: objetos não agrupados

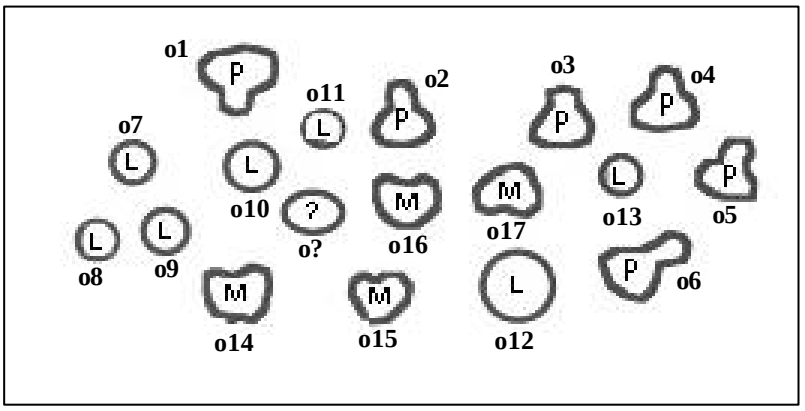

Figura 3. O problema de clusterização: elementos agrupados após a aplicação de alguma técnica de clusterização

A Tabela 2 mostra uma partição *crisp* dos 17 elementos, em três classes, P, L e M, para os dados da Figura 2. Note na tabela que cada elemento pertence a apenas uma das três classes, com pertinência total (=1). A linha **Total** representa a soma dos valores de pertinência de cada um dos elementos a cada uma das classes. Na tabela todos os objetos da mesma classe estão em colunas adjacentes.

Tabela 2. Partição crisp para os objetos da Figura 2

Partição Crisp																		
	o1	***o2***	***o3***	***o4***	***o5***	***o6***	***o7***	***o8***	***o9***	***o10***	***o11***	***o12***	***o13***	***o14***	***o15***	***o16***	***o17***	***o?***
P	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	1
M	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
Total	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

A Tabela 3 mostra o mesmo conjunto de dados, agora agrupados por uma pseudopartição *fuzzy* de três classes, P, L e M, para os dados da Figura 2.

Tabela 3. Pseudopartição fuzzy para os elementos da Figura 2

Partição Crisp																		
	o1	***o2***	***o3***	***o4***	***o5***	***o6***	***o7***	***o8***	***o9***	***o10***	***o11***	***o12***	***o13***	***o14***	***o15***	***o16***	***o17***	***o?***
P	0.9	1	0.7	0.7	0.9	1	0	0	0.1	0.1	0.1	0.1	0	0.1	0.1	0.2	0.2	0.15
L	0	0	0.1	0	0	0	0.8	0.8	0.8	0.7	0.8	0.6	0.8	0.1	0.1	0.1	0.1	0.60
M	0.1	0	0.2	0.3	0.1	0	0.2	0.2	0.1	0.2	0.1	0.3	0.2	0.8	0.8	0.7	0.7	0.25
Total	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

No caso da classificação *crisp* mostrada na Tabela 2, ao elemento “o?” é atribuída (erroneamente) a pertinência completa à classe "laranja" – provavelmente porque tal objeto tem a forma parecida com a de uma laranja. O uso de uma pseudopartição *fuzzy* como a mostrada na Tabela 3 permite evitar este tipo de erro. A última coluna dessa tabela atribui a maior pertinência (0.60) de “o?” à classe "laranja"; entretanto, admite a possibilidade do elemento em questão não ser uma laranja atribuindo, assim, graus de pertinências menores do elemento às classes "pêra" (0.15) e "maçã" (0.25). Dessa forma, um modelo *fuzzy* provê uma estrutura de solução mais rica e flexível, que modela objetos do mundo com um alto grau de detalhe. Note que algumas colunas das pseudopartições *fuzzy* (Tabela 3) são *crisp* (a segunda coluna, por exemplo); elas correspondem a elementos que podem ser associados unicamente e com certeza a uma determinada classe.

Vários métodos têm sido desenvolvidos para obter tanto *clusters* convencionais (*crisp*) quanto *fuzzy* a partir de um determinado conjunto de dados. Para o caso da abordagem *fuzzy*, foco deste trabalho, na seção seguinte é apresentado o algoritmo *fuzzy c-means*, dada a sua relevância e aplicabilidade.

3 Método de Clusterização *Fuzzy C-means*

O método de clusterização *fuzzy c-means* foi proposto em Bezdek (1981)[4] e é descrito a seguir.

Dado um conjunto de dados $X = \{x_1,x_2,\ldots,x_n\}$ onde x_k, em geral, é um vetor de características $x_k = [x_{k_1},x_{k_2},\ldots,x_{k_p}] \in R^p$ para todo $k \in \{1,2,\ldots,n\}$ sendo R^p o espaço p-dimensional, o problema de clusterização *fuzzy* é encontrar uma pseudopartição *fuzzy* que representa a estrutura dos dados da melhor forma possível.

Uma pseudopartição *fuzzy* de X é uma família de c subconjuntos *fuzzy* de X, denotada por $P = \{A_1,A_2,\ldots,A_c\}$ que satisfaz as equações (1) e (2).

$$\sum_{i=1}^{c} A_i(x_k) = 1 \qquad (1)$$

para todo $k \in \{1,2,\ldots,n\}$ sendo que n representa o número de elementos do conjunto X. Ou seja, a soma dos graus de pertinência de um elemento em todas as famílias deve ser igual a um.

$$0 < \sum_{k=1}^{n} A_i(x_k) < n \qquad (2)$$

[4] Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981. Apud (Klir e Yuan, 1995).

para todo $i \in \{1,2,\ldots,c\}$ sendo que c representa o número de classes. Ou seja, a soma dos graus de pertinência de todos os elementos de uma família deve ser menor que o número de elementos existentes no conjunto universo X. Por exemplo, dado $X = \{x_1, x_2, x_3\}$ e

$$A_1 = 0.6/x_1 + 1/x_2 + 0.1/x_3$$
$$A_2 = 0.4/x_1 + 0/x_2 + 0.9/x_3$$

note que $0.6 + 0.4 = 1$, $1 + 0 = 1$ e $0.1 + 0.9 = 1$, o que satisfaz (1) e que $0.6 + 1 + 0.1 = 1.7 < 3$ e $0.4 + 0 + 0.9 = 1.3 < 3$, o que satisfaz (2). Portanto $\{A_1,A_2\}$é uma pseudopartição *fuzzy* de *X*.

O método *fuzzy c-means* pode ser equacionado por meio de um algoritmo iterativo, baseado na minimização de um índice de desempenho, que indica a adequabilidade da pseudopartição gerada. O desempenho do algoritmo é influenciado pela escolha do número de classes c, dos centros de *cluster* iniciais, da ordem na qual os vetores são processados, da medida de distância, do critério de parada e pelas propriedades geométricas dos dados. Os conjuntos que apresentam *clusters* compactos, bem separados e com formas hiper-esféricas, são apropriados para este método, mas para encontrar agrupamentos adequados é necessário realizar extensivos testes com vários valores de c, distâncias, critérios de parada, centros de *cluster* iniciais e diferentes ordens de amostras.

Assim, o algoritmo assume como entrada os seguintes parâmetros: o número desejado de *clusters* c; uma medida de distância $m \in (1,\infty)$, que define a distância permitida entre os pontos e os centros de *cluster*; e um número pequeno $\varepsilon > 0$, utilizado como um critério de parada, além da instanciação inicial dos graus de pertinência dos objetos a cada uma das classes e dos centros de *cluster* dessas classes. A seguir, são descritos os passos do algoritmo.

Passo 1: Considere t = 0 representando a iteração 0. Defina $P^{(0)}$ como uma pseudopartição $P = \{A_1,A_2,\ldots,A_c\}$, atribuindo os graus de pertinência dos elementos às classes dessa pseudopartição, os quais podem ser informados ou calculados de forma aleatória.

Passo 2: No caso de t = 0, forneça os c centros de *cluster* $v_1^{(t)},\ldots v_c^{(t)}$ iniciais[5]. Para t > 0, calcule os c centros de *cluster* $v_1^{(t)},\ldots v_c^{(t)}$ por (3) para $P^{(t)}$ e o valor de m fornecido.

$$v_i = \frac{\sum_{k=1}^{n} [A_i(x_k)]^m x_k}{\sum_{k=1}^{n} [A_i(x_k)]^m} \qquad (3)$$

A variável real m > 1 é chamada de **índice de fuzzificação** e é usada para definir a distância permitida entre os pontos e o centro que está sendo calculado. Quanto maior o valor de m, mais elementos do conjunto são considerados como pertencentes a uma

[5] Segundo Bezdek (2002), os centros de *cluster* iniciais (em t = 0) podem também ser calculados de acordo com a fórmula (3), em vez de fornecidos. De qualquer forma, deve-se evitar valores idênticos para centros de *cluster* de diferentes classes, uma vez que este tipo de inicialização pode fazer com que esses valores sejam mantidos durante todas as iterações do algoritmo, causando influência negativa nos resultados.

pseudopartição. Esse parâmetro é escolhido de acordo com o problema considerado. Não existe nenhuma base teórica para uma escolha ótima do valor de m.

O vetor ν_i calculado por (3), visto como o centro do *cluster* A_i, é a média ponderada dos dados em A_i. O peso do dado x_k é a m-ésima potência do seu grau de pertinência ao conjunto *fuzzy* A_i.

Passo 3: Atualize $P^{(t)}$ para $P^{(t+1)}$ usando o seguinte procedimento: para cada $x_k \in X$ e para todo $i \in \{1,2,\ldots,c\}$, se $\|x_k - \nu_i^{(t)}\|^2 > 0$, calcule o grau de pertinência do elemento x_k à classe A_i, pela fórmula (4), sendo as classes A_i podem ser selecionadas em uma ordem pré-definida ou aleatoriamente.

$$A_i^{(t+1)}(x_k) = \left[\sum_{j=1}^{c} \left(\frac{\| x_k - v_i^{(t)} \|^2}{\| x_k - v_j^{(t)} \|^2} \right)^{\frac{1}{m-1}} \right]^{-1} \qquad (4)$$

Assume-se que $\| \cdot \|$ é alguma norma indicada para o produto interno no espaço R^p e $\|x_k - \nu_i^{(t)}\|^2$ representa a distância entre x_k e ν_i.

Por meio da fórmula (4), a distância de cada elemento x_k ao centro de *cluster* ν_i da pseudopartição atual A_i é comparada à distância do mesmo elemento ao centro de *cluster* ν_j de todas as demais pseudopartições A_j, de modo a atribuir um grau de pertinência ao elemento x_k no *cluster* A_i que seja proporcional ao seu grau de pertinência aos demais *clusters* A_j.

Quando $\|x_k - \nu_i^{(t)}\|^2 = 0$ para algum $i \in I \subseteq \{1,2,\ldots,c\}$, o que indica que x_k corresponde ao centro de *cluster* ν_i, defina $A_i^{(t+1)}(x_k)$ como um número real não negativo que satisfaz (5) e defina $A_i^{(t+1)}(x_k) = 0$ para $i \in \{1,2,\ldots,c\} - I$.

$$\sum_{i \in I} A_i^{(t+1)}(x_k) = 1 \qquad (5)$$

Em casos como este, quando o elemento coincide com o centro de *cluster* de uma classe A_i, o ideal seria que o algoritmo atribuísse a tal elemento o grau de pertinência máximo (1) àquela classe e o grau de pertinência mínimo (0) às demais classes. No entanto, o teste realizado pela fórmula (5) se torna necessário, uma vez que esse elemento pode ter sido associado a outras classes com um grau de pertinência maior que zero, se A_i não for a primeira classe selecionada pelo algoritmo. Nota-se, portanto, a influência da ordem na qual as pseudopartições são escolhidas.

Passo 4: Como critério de parada, compare $P^{(t)}$ e $P^{(t+1)}$. Se $|P^{(t)} - P^{(t+1)}| \leq \varepsilon$, então páre; caso contrário, faça $t = t + 1$ e retorne ao Passo 2.

Neste teste, $|P^{(t)} - P^{(t+1)}|$ denota a distância entre $P^{(t+1)}$ e $P^{(t)}$, dada pela fórmula (6):

$$|P^{(t)} - P^{(t+1)}| = \max_{i \in Nc, k \in Nn} |A_i^{(t+1)}(x_k) - A_i^{(t)}(x_k)| \qquad (6)$$

Essa fórmula verifica se a máxima diferença entre o grau de pertinência de qualquer elemento x_k em uma dada classe A_i na iteração atual e na iteração anterior é menor que o erro definido ε, sendo isso feito para todas as classes. Em caso positivo, o

algoritmo deve parar e retornar os graus de pertinências atuais de todos os elementos em todas as classes como resultado. Note que, quanto menor o valor de ε, maior o número de passos e, conseqüentemente, mais refinada é a pseudopartição final obtida.

Ao final da última iteração do algoritmo, a pseudopartição obtida deve ser analisada segundo algum critério que expresse a idéia geral de que as associações são fortes dentro do *cluster* e fracas entre *clusters*. Este critério pode ser definido, por exemplo, em termos de índices de desempenho, tal como o índice $J_m(P)$, dado em função dos centros de *cluster* e de m pela fórmula (7).

$$J_m(P) = \sum_{k=1}^{n}\sum_{i=1}^{c}[A_i(x_k)]^m \parallel x_k - v_i \parallel^2 \qquad (7)$$

Este índice de desempenho mede, para todos os elementos, a soma das distâncias ponderadas de cada elemento a cada um dos centros de *cluster* da pseudopartição. Quanto menor o valor de $J_m(P)$, melhor a pseudopartição *fuzzy* P. Conseqüentemente, o objetivo do método de clusterização *fuzzy c-means* é encontrar uma pseudopartição P que minimize o índice de desempenho $J_m(P)$. Assim, o problema de clusterização pode ser equacionado como um problema de otimização. Vários testes, utilizando diferentes valores para os parâmetros do algoritmo, podem ser feitos de forma a se obter índices de performance menores, ou seja, resultados mais adequados.

3.1 Exemplo de Aplicação do Método *Fuzzy C-means*

Para ilustrar o método *fuzzy c-means*, considere o exemplo adaptado de Klir e Yuan, (1995) e Zimmermann (1991). Nesse exemplo, o conjunto de dados X consiste de 15 pontos em R^2, listados na Tabela 4, e representados graficamente na Figura 4.

Tabela 4. Exemplo de conjunto de dados

k	x_{k1}	x_{k2}
1	0	0
2	0	2
3	0	4
4	1	1
5	1	2
6	1	3
7	2	2
8	3	2
9	4	2
10	5	1
11	5	2
12	5	3
13	6	0
14	6	2
15	6	4

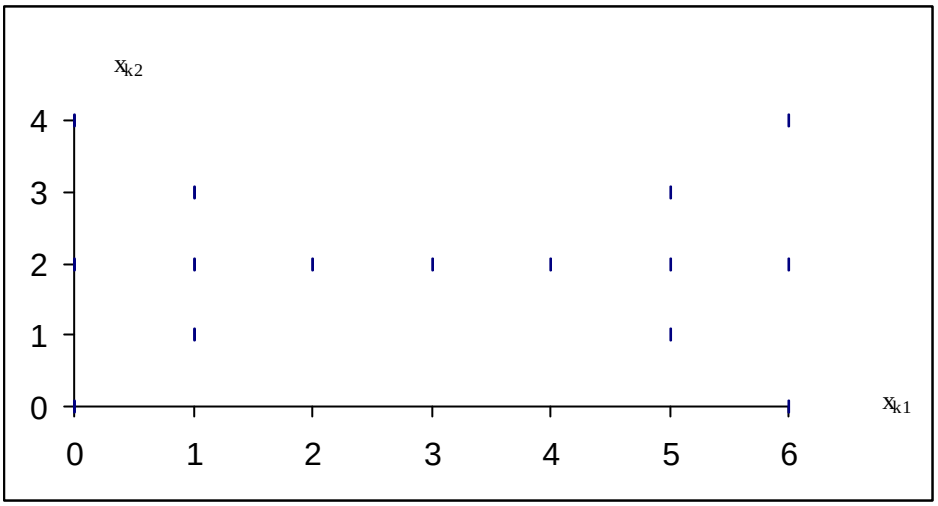

Figura 4. Gráfico do exemplo do conjunto de dados da Tabela 4

Assuma que se queira determinar uma pseudopartição *fuzzy* de X com dois *clusters* (c = 2). Assuma também que os parâmetros escolhidos foram: m = 1.25, ε = 0.01 e $\|\ \|$ = distância Euclidiana. De acordo com esses parâmetros, a execução do algoritmo *fuzzy c-means* ocorre como descrito a seguir.

Passo 1: Considere t = 0 e a pseudopartição *fuzzy* inicial $P^{(0)} = \{A_1, A_2\}$ dada pela Tabela 5.

Tabela 5. Pseudopartição fuzzy $P^{(0)} = \{A_1, A_2\}$

Iteração 0		
k	$A_1(x_k)$	$A_2(x_k)$
1	0.854	0.146
2	0.854	0.146
3	0.854	0.146
4	0.854	0.146
5	0.854	0.146
6	0.854	0.146
7	0.854	0.146
8	0.854	0.146
9	0.854	0.146
10	0.854	0.146
11	0.854	0.146
12	0.854	0.146
13	0.854	0.146
14	0.854	0.146
15	0.854	0.146

Passo 2: Como t = 0, os centros de *cluster* iniciais são dados: c_1 = (0.1,2) e c_2 = (4.65,2).

Passo 3: A pseudopartição é atualizada para $P^{(1)}$, calculando-se, para tanto, os graus de pertinência de todos os elementos x_k a todas as classes A_i pela fórmula (4). Os valores obtidos são mostrados na Tabela 6.

Tabela 6. Pseudopartição fuzzy $P^{(1)} = \{A_1, A_2\}$

Iteração 0		
k	$A_1(x_k)$	$A_2(x_k)$
1	0.9760923628	0.0239076372
2	0.9999997861	0.0000002139
3	0.9760923628	0.0239076372
4	0.9842805128	0.0157194872
5	0.9963170473	0.0036829527
6	0.9842805128	0.0157194872
7	0.7909767386	0.2090232614
8	0.0948553388	0.9051446612
9	0.0007710100	0.9992289900
10	0.0020103485	0.9979896515
11	0.0000260301	0.9999739699
12	0.0020103485	0.9979896515
13	0.0220122841	0.9779877159
14	0.0027336204	0.9972663796
15	0.0220122841	0.9779877159

Passo 4: A diferença máxima entre os graus de pertinência dos elementos nas duas classes em $P^{(0)}$ e $P^{(1)}$ é 0.853973969857095, que é maior que o erro ($\varepsilon = 0.01$). Portanto, t é incrementado de 1, ou seja, t = 1 e o algoritmo continua, retornando para o Passo 2. Na próxima iteração, os passos executados do algoritmo são:

Passo 2: Como t > 0, os centros de *cluster* são calculados pela fórmula (3), sendo obtidos os seguintes valores: $c_1 = (0.7024792624,2)$ e $c_2 = (4.9512886483,2)$.

Passo 3: A pseudopartição é atualizada para $P^{(2)}$, calculando-se, para tanto, os graus de pertinência de todos os elementos x_k a todas as dasses A_i pela fórmula (4). Os valores obtidos são mostrados na Tabela 7.

Tabela 7. Pseudopartição fuzzy $P^{(2)} = \{A_1, A_2\}$

Iteração 1		
k	$A_1(x_k)$	$A_2(x_k)$
1	0.9757697569	0.0242302431
2	0.9995949719	0.0004050281
3	0.9757697569	0.0242302431
4	0.9957250442	0.0042749558
5	0.9999678560	0.0000321440
6	0.9957250442	0.0042749558
7	0.9639852128	0.0360147872
8	0.3422322271	0.6577677729
9	0.0068786220	0.9931213780
10	0.0026438426	0.9973561574
11	0.0000000165	0.9999999835
12	0.0026438426	0.9973561574
13	0.0246733233	0.9753266767
14	0.0015334398	0.9984665602
15	0.0246733233	0.9753266767

Passo 4: A diferença máxima entre os graus de pertinência dos elementos nas duas classes em $P^{(1)}$ e $P^{(2)}$ é 0.853999983493736, que é maior que o erro ($\varepsilon = 0.01$). Portanto, t é incrementado de 1, ou seja, t = 2 e o algoritmo continua, retornando para o Passo 2.

Nas demais iterações, os valores obtidos para cada pseudopartição são mostrados nas Tabelas de 8 a 11.

Tabela 8. Pseudopartição fuzzy $P^{(3)} = \{A_1, A_2\}$

Iteração 2		
Centros	c_1 = (0.8116230105,2)	c_2 = (5.0806193406,2)
k	$A_1(x_k)$	$A_2(x_k)$
1	0.9761627979	0.0238372021
2	0.9993491693	0.0006508307
3	0.9761627979	0.0238372021
4	0.9965704637	0.0034295363
5	0.9999954584	0.0000045416
6	0.9965704637	0.0034295363
7	0.9783353083	0.0216646917
8	0.4496764048	0.5503235952
9	0.0130232448	0.9869767552
10	0.0029377357	0.9970622643
11	0.0000001373	0.9999998627
12	0.0029377357	0.9970622643
13	0.0239684585	0.9760315415
14	0.0009849799	0.9990150201
15	0.0239684585	0.9760315415

Diferença máxima = 0.107444177751615

Tabela 9. Pseudopartição fuzzy $P^{(4)} = \{A_1, A_2\}$

Iteração 3		
Centros	c_1 = (0.8470023789,2)	c_2 = (5.1188334748,2)
k	$A_1(x_k)$	$A_2(x_k)$
1	0.9761847418	0.0238152582
2	0.9992509185	0.0007490815
3	0.9761847418	0.0238152582
4	0.9967652072	0.0032347928
5	0.9999980961	0.0000019039
6	0.9967652072	0.0032347928
7	0.9816638690	0.0183361310
8	0.4840100305	0.5159899695
9	0.0156075774	0.9843924226
10	0.0030792083	0.9969207917
11	0.0000006704	0.9999993296
12	0.0030792083	0.9969207917
13	0.0238564768	0.9761435232
14	0.0008543209	0.9991456791
15	0.0238564768	0.9761435232

Diferença máxima = 0.0343336256076534

Tabela 10. Pseudopartição fuzzy $P^{(5)} = \{A_1, A_2\}$

Iteração 4		
Centros	$c_1 = (0.8585081034, 2)$	$c_2 = (5.1306412129, 2)$
k	$A_1(x_k)$	$A_2(x_k)$
1	0.9761776714	0.0238223286
2	0.9992166593	0.0007833407
3	0.9761776714	0.0238223286
4	0.9968209792	0.0031790208
5	0.9999986232	0.0000013768
6	0.9968209792	0.0031790208
7	0.9826320302	0.0173679698
8	0.4949204140	0.5050795860
9	0.0165017149	0.9834982851
10	0.0031296164	0.9968703836
11	0.0000009901	0.9999990099
12	0.0031296164	0.9968703836
13	0.0238354060	0.9761645940
14	0.0008167417	0.9991832583
15	0.0238354060	0.9761645940

Diferença máxima = 0.0109103835384644

Tabela 11. Pseudopartição fuzzy $P^{(6)} = \{A_1, A_2\}$

Iteração 5		
Centros	$c_1 = (0.8621945829, 2)$	$c_2 = (5.1343585874, 2)$
k	$A_1(x_k)$	$A_2(x_k)$
1	0.9761739226	0.0238260774
2	0.9992054318	0.0007945682
3	0.9761739226	0.0238260774
4	0.9968381054	0.0031618946
5	0.9999987657	0.0000012343
6	0.9968381054	0.0031618946
7	0.9829313006	0.0170686994
8	0.4983863829	0.5016136171
9	0.0167935358	0.9832064642
10	0.0031462011	0.9968537989
11	0.0000011117	0.9999988883
12	0.0031462011	0.9968537989
13	0.0238302311	0.9761697689
14	0.0008051776	0.9991948224
15	0.0238302311	0.9761697689

Diferença máxima = 0.00346596888136857

Dado que na iteração t = 5 a diferença entre graus de pertinência de elementos nas duas classes com relação aos graus de pertinência em $P^{(5)}$ e $P^{(6)}$ é igual a 0.00346596888136857, que é menor que o erro ($\varepsilon = 0.01$), o algoritmo pára e a pseudopartição *fuzzy* obtida é aquela mostrada na Tabela 11.

Ao final dessa última iteração (t = 5), o desempenho da pseudopartição obtida é analisado pela fórmula (7) e o valor resultante de $J_m(P)$ é 18.7541165880264. Conforme mencionado, esse valor poderia ser minimizado com a utilização de outros parâmetros de entrada para o algoritmo. Na próxima seção será apresentada uma implementação do método *fuzzy c-means* – a partir da qual os valores desse exemplo foram obtidos – que permite a variação dos parâmetros de entrada, de modo que se possam realizar testes exaustivos para buscar a otimização dos resultados.

4 Uma Implementação do Método *Fuzzy C-means*

A implementação do método *fuzzy c-means* realizada neste trabalho segue a abordagem clássica descrita na Seção 3, exceto por duas limitações estabelecidas em função do tempo disponível para tal implementação: 1) a ordem fixa (crescente) de análise dos *clusters*, ou seja, não é possível variar a ordem de seleção dos *clusters* para o cálculo dos graus de pertinência dos objetos a esses *clusters*; e 2) o tipo dos objetos do conjunto universo que vão ser agrupados pelo método, pois são permitidos somente objetos de duas dimensões, ou seja, pontos no espaço bi-dimensional, como os do exemplo da Seção 3.1. Portanto, na descrição dessa implementação, usaremos o termo "ponto" para referenciar os objetos do conjunto universo. A implementação foi realizada em Object Pascal Delphi 6.0. Na Tabela 12 estão descritos os principais arquivos que fazem parte do sistema.

Tabela 12. Arquivos do sistema Fuzzy C-means

Arquivo	***Descrição***
C_Means.dpr	código fonte do projeto Delphi
C_Means.exe	código executável do projeto Delphi
main.pás	código fonte do programa principal, que manipula todas as funções implementadas e que gerencia os formulários utilizados pelo sistema
main.dfm	especificação da tela do programa principal
inserir.pas	código fonte para inserção dos valores dos pontos
inserir.dfm	especificação da tela para inserção dos valores dos pontos
pertinencia.pas	código fonte para inserção dos graus de pertinência dos pontos
pertinencia.dfm	especificação da tela para a inserção dos graus de pertinência dos pontos
cluster.pas	código fonte para inserção dos centros de *cluster* dos grupos
cluster.dfm	especificação da tela para inserção dos centros de *cluster* dos grupos
relatorio.pas	código fonte para exibição dos dados dos cálculos em um relatório
relatório.dfm	especificação da tela para exibição dos dados dos cálculos

As Tabelas 13, 14, 15 e 16 descrevem os dados contidos em cada uma das principais telas do programa.

Tabela 13. Descrição da tela principal do programa

Objeto da tela	***Função do objeto***
Número de Pontos	quantidade de pontos do conjunto universo (n) – especificado pelo

	usuário
Número de Grupos	quantidade de grupos (*clusters*) para a clusterização (c) – especificado pelo usuário
Erro Associado	erro máximo (å) – especificado pelo usuário
Expoente m	índice de fuzzificação (m) – especificado pelo usuário
Inserir Pontos	mostra o formulário para a inserção das coordenadas dos pontos
Coordenadas dos Pontos	mostra os pontos inseridos pelo usuário
Inserir Graus	mostra o formulário para a inserção dos graus de pertinência dos pontos
Graus de Pertinência dos Pontos	mostra os graus de pertinência inseridos pelo usuário
Inserir Centros	mostra o formulário para a inserção de centros de *cluster*
Centros de *Cluster*	mostra os centros de *cluster* inseridos pelo usuário
Número de Iterações	número de iterações realizadas pelo sistema para obter a clusterização
Erro Calculado	erro calculado pelo sistema na última iteração
Valor de J	índice de desempenho ($J_m(P)$) – calculado pelo sistema ao fim da última iteração
Novo Cálculo	limpa todos os campos para que o usuário possa fornecer novos valores
Calcular	realiza a clusterização para os parâmetros de entrada informados pelo usuário
Relatório	mostra formulário com os valores intermediários do cálculo da clusterização
Exemplo	inicializa os parâmetros com os dados do exemplo da Seção 3.1

Tabela 14. Descrição da tela para a inserção das coordenadas dos pontos

Objeto da Tela	***Função do Objeto***
Coordenada X	valor da coordenada X – especificada pelo usuário
Coordenada Y	valor da coordenada Y – especificada pelo usuário

Tabela 15. Descrição da tela para a inserção dos graus de pertinência

Objeto da Tela	***Função do Objeto***
Geral	valor do grau de pertinência para os pontos de todo o grupo – especificado pelo usuário
Específico	valor do grau de pertinência para cada ponto do grupo – especificado pelo usuário
Aleatório	valores aleatórios de graus de pertinência para todos os pontos de um grupo – gerados pelo sistema

Conforme ilustrado na Tabela 15, para fornecer os graus de pertinência iniciais dos pontos, o sistema disponibiliza três métodos, referenciados como Geral, Específico e Aleatório. A opção **Geral** permite que o usuário atribua um mesmo grau de pertinência a todos os pontos de um grupo.

A opção **Específico** permite que o usuário insira graus de pertinência específicos a cada ponto de um grupo, desde que respeite as restrições para partições *fuzzy* definidas anteriormente neste trabalho.

A opção **Aleatório** permite a criação automática de graus de pertinência aleatórios para cada grupo de pontos.

Tabela 16. Descrição da tela para a inserção dos c centros de cluster iniciais

Objeto da Tela	*Função do Objeto*
Coordenada X	valor da coordenada X do centro – especificado pelo usuário
Coordenada Y	valor da coordenada Y do centro – especificado pelo usuário

O algoritmo que implementa o método *fuzzy c-means* foi convertido em procedimentos e funções na linguagem *Object Pascal*, referenciados conforme mostra a Tabela 17.

Tabela 17. Principais procedimentos e funções do sistema

Procedimento/Função	*Descrição*
procedure CalculaV	calcula os centros de *cluster* a cada iteração
procedure CalculaU	calcula os graus de pertinência a cada iteração
function CalculaErro	calcula o erro a cada iteração
function CalculaJ	calcula o valor de J na última iteração
function Distancia(PontoA, PontoB:TPonto)	calcula a distância entre dois pontos A e B, usada no cálculo dos graus de pertinência e do $J_m(P)$

As principais variáveis e constantes utilizadas pelo sistema estão descritas na Tabela 18.

Tabela 18. Principais variáveis e constantes do sistema

Variável/Constante	*Descrição*
MAX_PONTOS	constante que define o número máximo de pontos
MAX_GRUPOS	constante que define o número máximo de grupos
PRECISAO	constante que define o precisão de arredondamento
Ponto	vetor que armazena as coordenadas dos pontos
V	vetor que armazena as coordenadas dos centros de *cluster*
U, UAnt	matrizes que armazenam os graus de pertinência dos pontos (U – graus de pertinência da iteração atual, Uant – graus de pertinência da iteração anterior)
Grupos, Pontos, Iteracao	variáveis que armazenam o número de grupos, de pontos e de iterações, respectivamente
M	variável que armazena índice de fuzzificação

Para a execução do programa é necessário especificar o número de pontos do conjunto universo (n), o número de grupos a serem criados (c), o valor do erro máximo como critério de parada ($\varepsilon > 0$) e um valor para o índice de fuzzificação ($m > 1$). Em seguida, é preciso inserir os valores dos n pontos, clicando no botão **Inserir Pontos**, os valores dos graus de pertinência iniciais desses pontos, clicando no botão **Inserir Graus**, e os valores iniciais para os c centros de cluster, clicando no botão **Inserir Centros**. Após a inserção desses parâmetros de entrada pelo usuário e a realização das devidas verificações de consistência pelo sistema, a clusterização pode ser realizada,

clicando no botão **Calcular**. Realizado o cálculo, além dos valores finais (Número de Iterações, Erro Calculado e $J_m(P)$), o sistema exibe em uma nova janela o relatório com todos os valores intermediários do cálculo. Se essa janela for fechada, os valores intermediários podem ser visualizados novamente clicando no botão **Relatório**, desde que um novo cálculo não tenha sido iniciado (botão **Novo Cálculo**). Todos os parâmetros de entrada e resultados da última clusterização podem ser excluídos clicando no botão **Novo Cálculo**. Para testar o exemplo da Seção 3.1, os parâmetros de entrada podem ser inicializados clicando no botão **Exemplo**.

5 Considerações Finais

Dentre as várias propostas de solução para o problema de clusterização, o método *fuzzy c-means* vem sendo bastante utilizado por apresentar resultados satisfatórios. No entanto, a obtenção desses resultados depende de uma série de fatores, entre os quais podem-se destacar a influência dos parâmetros de entrada (ε, m, c), a instanciação inicial dos graus de pertinência dos elementos às c classes, dos c centros de *cluster* iniciais e a ordem na qual as classes são selecionadas.

A definição desses parâmetros é feita, na sua grande maioria, de forma empírica, pois não existe nenhuma base teórica para a escolha de valores adequados, uma vez que esses valores dependem do conjunto de dados e do problema em foco.

Em especial, a necessidade de definição do parâmetro c pode ser considerada como uma desvantagem do método descrito, dada a dificuldade em se especificar, *a priori*, o número de classes desejado. Para tanto, foram desenvolvidas algumas extensões ou generalizações do método, nas quais o parâmetro c é calculado pelo próprio algoritmo, em vez de ser fornecido como parâmetro de entrada. Um desses métodos, por exemplo, é o descrito em Gath e Geva (1992).

A qualidade dos resultados obtidos a partir da aplicação de algoritmos é uma preocupação inerente a várias áreas. No caso do algoritmo *fuzzy c-means*, a avaliação dos resultados, ou seja, da pseudopartição obtida, pode ser realizada por meio de índices que medem o desempenho dessa pseudopartição. O índice de desempenho proposto por Bezdek, além de depender de valores resultantes do algoritmo, sofre a influência do parâmetro m. Além disso, não se dispõe de um valor de referência que possa ser utilizado como base na comparação do desempenho obtido. Diferentes abordagens de avaliação podem ser encontradas, por exemplo, em Windham (1992a).

Como um problema de otimização que visa minimizar o valor desse índice de desempenho, o algoritmo *fuzzy c-means* pode apresentar resultados mais adequados por meio da realização de testes exaustivos com diferentes valores para os parâmetros de entrada.

Bibliografia

Backer, E. *Computer Assisted Reasoning in Cluster Analysis*. Prentice Hall, New York, 1995.

Baraldi, A.; Blonda P. A Survey of Fuzzy Clustering Algorithms for Pattern Recognition – Part II. In *IEEE Transactions on systems, Man, and Cybernetics – Part B: Cybernetics*, vol. 29, n. 6, 1999.

Bezdek, C. J. e Pal, S. K. *Fuzzy Models for Pattern Recognition*. IEEE Press, New York, 1992.

Bezdek, C. J. [jbezdek@cs.uwf.edu], *Fuzzy C-means*, comunicação pessoal, 02 de Maio de 2002.

Bobrowski, L.; Bezdek, J. C. C-Means Clustering with the l_1 and l_∞ Norms. In *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, n. 3, Maio/Junho, 1991.

Gath, I; Geva, B. Unsupervised Optimal Fuzzy Clustering. In James C. Bezdek & Sankar K. Pal (ed) *Fuzzy Models for Pattern Recognition*. IEEE Press, p. 211-218, 1992.

Gustafson, D. E; Kessel, W. C. Fuzzy Clustering with a Fuzzy Covariance Matrix. In James C. Bezdek & Sankar K. Pal (ed) *Fuzzy Models for Pattern Recognition*. IEEE Press, p. 117-122, 1992.

Hartigan, J. A. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.

Keller, A.; Klawonn, F. Fuzzy Clustering with Weighting of Data Variables. In *International Journal of Uncertainty Fuzziness and Knowledge-based Systems*, vol. 8, n. 6, pp. 735-746, 2000.

Klawonn, F.; Keller, A. Fuzzy Clustering Based on Modified Distance Measures. In *Proceedings of Advances in Intelligent Data Analysis*, vol. 1642, pp. 291-301, 1999.

Klir, G. J. e Yuan, B. *Fuzzy Sets and Fuzzy Logic – Theory and Applications*. Prentice Hall, 1995.

Menard, M.; Demko, C.; Loonis, P. The Fuzzy C+2-means: Solving the Ambiguity Rejection in Clustering. In *Pattern Recognition*, vol. 33, n. 7, pp. 1219-1237, 2000.

Nascimento, S.; Mirkin, B.; Moura, F. P. Multiple Prototype for Fuzzy Clustering. In *Advances in Intelligent Data Analysis*, vol. 1642, pp. 269-279, 1999.

Tamura, S.; Higuchi, S.; Tanaka, K. Pattern Classification Based on Fuzzy Relations. In *IEEE Transactions Systems, Man, Cybernetics*, vol. SMC-1, n.1, pp. 61-66, 1971.

TW, C.; DB., G.; LO., H. Fast Fuzzy Clustering. In *Fuzzy Sets and Systems*, vol. 93, n. 1, pp. 49-56, 1998.

Zahid, N.; Abouelala, O.; Limouri, M.; Essaid, A. Fuzzy Clustering Bsed on K-nearest-neighbours Rule. In *Fuzzy Sets and Systems*, vol. 120, n. 2, pp. 239-247, 2001.

Zimmermann, H. J. *Fuzzy Set Theory and its Applications*. Kluwer Academic Publishers, Boston, 1991.

Windham, M. P. Cluster Validity for the Fuzzy C-Means Clustering Algorithm. In James C. Bezdek & Sankar K. Pal (ed) *Fuzzy Models for Pattern Recognition*. IEEE Press, 1992a, p. 195-201.

Windham, M. P. Geometrical Fuzzy Clustering Algorithms. In James C. Bezdek & Sankar K. Pal (ed.) *Fuzzy Models for Pattern Recognition*. IEEE Press, p. 123-129, 1992b.