Tutorial sobre *Fuzzy-c-Means* e *Fuzzy Learning Vector Quantization*: Abordagens Híbridas para Tarefas de Agrupamento e Classificação

Thiago Rocha [1] Sarajane Marques Peres [2]
Helton H. Bíscaro [2] Renata Cristina B. Madeo [2] Clodis Boscarioli [3]

Resumo: Neste tutorial é apresentada uma discussão sobre o algoritmo *Fuzzy-c-Means* e sobre as Redes Neurais *Fuzzy*, considerando a proposta de inserção de princípios da Teoria de Conjuntos *Fuzzy* nas abordagens de agrupamento e classificação clássicas: algoritmo *c-Means* e o modelo neural *Learning Vector Quantization*. A motivação para a construção de um modelo híbrido, dessa categoria, é conferir às abordagens clássicas a capacidade de lidar adequadamente com aspectos de incerteza e imprecisão, comumente encontrados em problemas reais.

Abstract: In this tutorial, we present a discussion on the Fuzzy-c-Means algorithm and the Fuzzy-Neural Networks, taking into account the proposal of inclusion of principles of the Fuzzy Set Theory into classical clustering and classification approaches, namely: c-Means algorithm and the Learning Vector Quantization neural model. The goal of building such a hybrid model is to enable the classical neural approach to deal appropriately with uncertainty and imprecision issues, commonly found in real problems.

1 Introdução

Sistemas Híbridos são caracterizados pela uso combinado de características referentes a duas ou mais técnicas da Inteligência Computacional, ou de áreas afins, em um único modelo, com o intuito de aproveitar o que de melhor cada uma das técnicas envolvidas pode oferecer à resolução de um problema. Nesse contexto encontram-se os modelos híbridos construídos a partir de, ou com o uso de, por exemplo, Teoria de Conjuntos *Fuzzy* (TCF) e Redes Neurais Artificiais (RNA).

Os modelos híbridos discutidos neste tutorial tem como base de "hibridização" a inserção de conceitos da Teoria de Conjuntos *Fuzzy* no algoritmo *c-Means*. Essa iniciativa dá

[1]Departamento de Engenharia Elétrica, Faculdade de Tecnologia, UnB, Gleba A, Av. L3 Norte, Brasília, DF {hiagoh@unb.br}
[2]Escola de Artes, Ciências e Humanidades, USP, Av. Arlindo Bettio, 1000, São Paulo, SP {sarajane@usp.br} {heltonhb@usp.br} {renata.si@usp.br}
[3]Universidade Estadual do Oeste do Paraná, Unioeste, R. Universitária, 1069, Cascavel, PR {Clodis.Boscarioli@unioeste.br}

origem ao algoritmo *Fuzzy-c-Means*, proposto por Bezdek, Tsao e Pal [4], o qual é a base para vários outros modelos híbridos aplicáveis a tarefas de agrupamento e classificação *fuzzy*, incluindo os modelos neuro-*fuzzy* aqui discutidos. Modelos híbridos inspirados no *Fuzzy-c-Means* ou relacionados a esse algoritmo são discutidos em [30, 31, 39, 18, 6, 36, 8, 26]. Como exemplos de aplicações recentemente desenvolvidas com o apoio do algoritmo *Fuzzy-c-Means* cita-se processamento de imagens [18, 6, 8, 37, 35, 40, 42], análise em sistemas de energia [36, 26] e reconhecimento de padrões [38]. O algoritmo FCM também foi recentemente usado por Pedrycs, Loia e Senatore [33] na experimentação de uma nova abordagem de introdução de conhecimento de domínio na tarefa de agrupamento.

Uma taxonomia simplificada para organizar os Sistemas Neurais Híbridos é discutida por Wermter e Sun [43], que consiste de três classes de arquitetura de sistemas: arquiteturas neurais unificadas (ou sistemas híbridos unificados), arquiteturas híbridas de transformação e arquiteturas híbridas modulares. A primeira classe constitui-se de representações conexionistas, porém, com a capacidade de conferir interpretações simbólicas para os nós e para as ligações da RNA. Na segunda classe estão as arquiteturas capazes de transformar representações simbólicas em representações conexionistas e vice-versa. E finalmente, arquiteturas que possuem tanto módulos simbólicos quanto módulos conexionistas adequados para resolver determinadas tarefas, ou parte de tarefas, estão inseridas na última classe.

Especificamente da combinação de RNA com características da TCF surgem duas classes de sistemas [27]: Redes Neurais *Fuzzy* (do inglês *fuzzy-neural network*) e Sistemas Neuro-*Fuzzy* (do inglês *neural-fuzzy systems*). A primeira classe é caracterizada por RNA dotadas da capacidade de manusear informações *fuzzy*[4], e a segunda classe é caracterizada por Sistemas *Fuzzy* estendidos pelo uso de RNA para melhorar algumas características como a adaptabilidade do sistema.

Estudos sobre Redes Neurais *Fuzzy* e Sistemas Neuro-*Fuzzy* são motivados pela necessidade de melhorar resultados em problemas complexos de classificação e agrupamento de dados, predição entre outros. Um breve histórico sobre essa área pode ser encontrado no trabalho de Vuorimaa [41]. Segundo esse autor, tais modelos tiveram o início de seu desenvolvimento entre o fim dos anos 80 e início dos anos 90 e, desde então, tem sido aplicados em diferentes problemas. O artigo "*Fuzzy Neural Networks*", publicado na revista *Mathematical Biosciences* [21], é considerado a primeira iniciativa na criação de sistemas híbridos utilizando RNA e TCF, com a proposta de substituir as somas ponderadas realizadas no modelo de neurônio *McCulloc-Pitts* [25] por uma operação *fuzzy* correspondente.

Ainda segundo Vuorimaa [41], uma coleção dos primeiros e mais importantes artigos acerca dos Sistemas Neuro-*Fuzzy* foi editada por Bezdek et al. [3], em 1992. Kosko [17], também em 1992, elaborou um livro abrangente sobre o assunto. Mais recentemente alguns autores tem apresentado trabalhos que discutem diferentes aspectos de sistemas neuro-*fuzzy*

[4]É nessa classe que se encaixa a discussão realizada neste tutorial.

e afins, por exemplo: [22, 29, 28, 20, 1].

O modelo de Rede Neural *Fuzzy* escolhido para ser explorado neste tutorial é o modelo *Fuzzy Learning Vector Quantization* (*Fuzzy* LVQ). Trata-se de um modelo inspirado na rede neural *Leaning Vector Quantization* (LVQ), um modelo de fácil entendimento e efetivo em termos de tarefas de classificação e agrupamento[5]. Existem várias iniciativas que usam o modelo *Fuzzy Learning Vector Quantization* em diferentes áreas de aplicação como as recentes aplicações em Geociências [11], processamento de dados sensoriais (*processing sensor-array data*) [13] e reconhecimento de padrões [7, 24]; e também algum esforço em propor novas formas de "hibridização", como os trabalhos [44, 45].

É nesse contexto que se encontra a motivação para dar atenção aos fundamentos de tal hibridização, pois percebe-se que o potencial desta área não está totalmente explorado, e é a partir do entendimento de suas bases que surgem as novas linhas de desenvolvimento e aplicação na área. Assim, este tutorial é dedicado a introduzir o leitor no estudo de abordagens híbridas para tarefas de agrupamento e classificação. A fim de explorar tal assunto, o algoritmo *Fuzzy-c-Means* e as Redes Neurais *Fuzzy* são as abordagens escolhidas para serem discutidas sob uma arquitetura de sistema híbrido unificado. De forma mais específica, foram selecionados dois modelos neurais, baseados na RNA LVQ, propostos por Chung e Lee [5] e por Bezdek, Tsao e Pal [4]. No modelo de Chung e Lee, graus de pertinências de dados a classes são utilizados para "fuzificar" a regra de aprendizado da RNA. No modelo de Bezdek, Tsao e Pal, uma variação não supervisionada da RNA LVQ é acrescida de características *fuzzy*. Ao serem escolhidos esses modelos se está conferindo a este tutorial um caráter de aplicação de análise de dados, especificamente para problemas de classificação e agrupamento.

Para melhor direcionar a leitura deste tutorial, optou-se por organizá-lo da seguinte forma: na Seção 2 é oferecida uma revisão de conceitos necessários ao entendimento do conteúdo aqui apresentado; o algoritmo *Fuzzy-c-Means* é discutido em detalhes na Seção 3 contextualizando-o na área de Agrupamentos *Fuzzy*; na Seção 4 as Redes Neurais *Fuzzy* derivadas da RNA LVQ são apresentadas. A Seção 5 apresenta alguns exemplos didáticos de aplicação dos algoritmos discutidos no tutorial. Um breve levantamento de iniciativas que aplicam os modelos aqui discutidos é apresentado na Seção 6 e, finalmente, na Seção 7 algumas considerações finais são delineadas.

[5] *Learning Vector Quantization* é um dos modelos que, juntamente com *Self-Organizing Maps* e *Adaptive Resonance Theory*, dominam a área de agrupamento de dados baseado em RNA [46].

2 Conceitos Básicos

Nesta seção são discutidos aspectos referentes a representação de dados, aprendizado computacional, tarefas de análise de dados, medidas de similaridade, erro de quantização, algoritmo *c-Means* clássico e modelo de RNA LVQ. O objetivo dessa discussão é contextualizar o leitor no que diz respeito às teorias utilizadas nas demais seções.

2.1 Conjunto de Dados

No contexto deste tutorial, um conjunto de dados X é definido como:

$$X = \begin{matrix} \vec{x_1} \\ \vec{x_2} \\ \cdots \\ \vec{x_n} \end{matrix} = \begin{matrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{matrix}$$

onde $\vec{x_j}$ é um vetor de p coordenadas e n é o número de elementos do conjunto de dados. Cada vetor representa um dado desse conjunto e cada coordenada desse vetor representa um atributo descritivo do dado. O conjunto de dados X reside no espaço $\Re^p$, e este espaço é referenciado pelos algoritmos de análise de dados como "espaço dos dados", "espaço de entrada" ou "espaço vetorial".

Geralmente esses dados apresentam algum tipo de organização que pode ser explicada por meio de partições do conjunto de dados, e o estudo dessas partições é uma tarefa de análise de dados. Essas partições podem ser caracterizadas como *classes* ou *grupos* de dados (também comumente chamados de *clusters*), a depender, respectivamente, da existência ou não de algum tipo de informação discriminatória explícita sobre elas.

2.2 Aprendizado Computacional

Na teoria da área de Inteligência Artificial existem diferentes tipos de aprendizado computacional que podem ser implementados por diferentes técnicas. É de interesse, neste tutorial, um tipo específico de aprendizado - o indutivo - nas modalidades supersionada e não supervisionada, implementados por algoritmos de otimização (*c-Means*, *Fuzzy-c-Means*, RNA LVQ e suas variações *fuzzy*).

No aprendizado indutivo, padrões e regras (partições e funções) são determinados a partir de um conjunto de dados ou representação de experiências. Na modalidade supervisionada, o algoritmo ajusta parâmetros de uma função a partir do erro medido entre as respostas obtidas com o uso de tal função e as respostas desejadas. Já na modalidade não supervisio-

nada, os parâmetros de uma função são ajustados com base na maximização de medidas de qualidade aplicadas ao resultado apresentado com o uso de tal função[6].

2.3 Tarefas de Análise de Dados: Classificação e Agrupamento

O termo *classe* deve ser usado quando existe informação sobre quantas e quais são as partições presentes no conjunto de dados, bem como qual dado pertence a qual partição. Nesse caso, o trabalho de análise de dados é denominado *classificação* e se propõe a determinar uma função que seja capaz de realizar o mapeamento entre os dados e suas classes, bem como mapear corretamente novos dados que venham a compor o conjunto de dados, trazendo ou não a informação sobre sua classificação. Formalmente, na tarefa de classificação, o mapeamento de um conjunto de vetores de dados de entrada ($\vec{x} \in \Re^p$) para um conjunto C finito de rótulos onde a cardinalidade[7] de C é c, é modelado em termos de alguma função:

$$\mathcal{F} : \Re^p \times W \rightarrow C$$

onde W é um espaço de parâmetros ajustáveis por meio de um algoritmo de aprendizado supervisionado.

O termo *grupo* deve ser usado quando não existe qualquer informação sobre como é a organização dos dados. Nesse caso, o trabalho de análise de dados é denominado *agrupamento*[8] e tem por objetivo estudar as relações de similaridade entre os dados, determinando quais dados formam quais grupos. Os grupos são formados de maneira a maximizar a similaridade entre os elementos de um grupo (similaridade intra-grupo) e minimizar a similaridade entre elementos de grupos diferentes (similaridade inter-grupos). Formalmente, dado um conjunto de dados de entrada ($\vec{x} \in \Re^p$), é encontrada uma função:

$$\mathcal{G} : \Re^p \times W \rightarrow C$$

onde W é um vetor de parâmetros ajustáveis, por meio de um algoritmo de aprendizado supervisionado ou não supervisionado, que determina c-classes ou grupos em X, $C = C_1, ..., C_c (c \leq n)$ tal que [46]:

[6]Para mais detalhes sobre os diferentes tipos de aprendizado e sobre sua relação com a teoria de RNA, veja [34], [12] e [10].

[7]De acordo com [3], c pode ser definido dentro do intervalo $]1, n[$ com o objetivo de rejeitar a hipótese de que o conjunto de dados X apresenta dados agrupados quando $c = 1$ ou $c = n$. Se $c = 1$ tem-se a situação em que os dados se encontram organizados em um único grupo ou classe. Já quando $c = n$ tem-se a situação em que cada dado do conjunto de dados representa um grupo ou classe.

[8]A tarefa de agrupamento também é conhecida como "clusterização".

1. $C_i \neq \emptyset, i = 1, ..., c$;
2. $\bigcup_{i=1}^{c} C_i = X$;
3. $C_i \cap C_j = \emptyset, i, j = 1, ..., c$ and $i \neq j$, assumindo a abordagem de classificação ou agrupamentos clássica.

2.4 Teoria de Conjuntos *Fuzzy*

A TCF foi introduzida por Loft Zadeh [48] na década de 60, época em que a visão tradicional sobre como a informação científica deveria ser trabalhada (evitando incertezas) estava começando a ser revista. A essência dessa teoria está na aceitação da "incerteza" como um fato das situações reais, que deve ser apropriadamente modelado e tratado por métodos matemáticos. A ideia de "incerteza" passa pelas dificuldades em lidar com informações imprecisas, falta de especificidade, conceitos vagos entre outras características que podem fazer com que uma situação não pareça clara ou nítida o suficiente para ser analisada pela teoria de conjuntos *crisp* (ou clássica).

O objetivo da TCF é tratar os fenômenos naturais ou as situações reais como são, para que se tenha uma modelagem muito mais próxima do que é real. Segundo Klir e Yuan [15], a possibilidade de estudar a incerteza tende a reduzir a complexidade dos modelos e a aumentar a credibilidade dos mesmos.

A TCF é uma generalização da teoria de conjuntos clássica que, ao liberar os estudos dos conjuntos da dicotomia imposta na teoria clássica, aumenta o poder de expressividade dos conjuntos (veja a Figura 1), de forma que um elemento pode pertencer ou não a um conjunto com determinados graus de pertinência e, mais do que isso, pode pertencer, com graus apropriados, a diferentes conjuntos definidos sobre um mesmo universo de discurso.

O que caracteriza a pertinência de um elemento a um conjunto é a função característica que o define, associando, no caso de conjuntos *crisp*, os valores 0 ou 1 a cada elemento de um universo de discurso, de forma a discriminá-lo como pertencente ou não ao conjunto em questão. A função f que define um conjunto *crisp* A é do tipo:

$$f_A : \mathcal{X} \rightarrow \{0, 1\}$$

em que $\mathcal{X}$ é o universo de discurso (um conjunto de elementos) e $\{0, 1\}$ define um conjunto de dois estados: 0 - não pertence; 1 - pertence.

No caso da TCF, uma função característica generalizada associa a cada elemento do universo de discurso um valor de um intervalo, entre 0 e 1, indicando o grau de pertinência dos elementos ao conjunto. Essa função assume o papel de **função de pertinência** que define

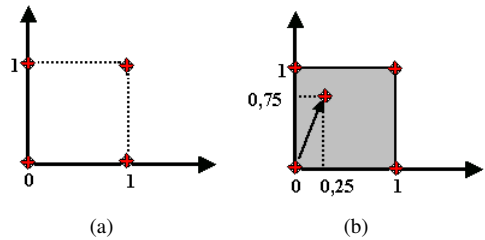


Figura 1. Interpretação geométrica do poder de expressividade da teoria de conjuntos (a) *crisp* e (b) *fuzzy* [32]. Em (a) existem apenas quatro vetores representados: (0;0), (0;1), (1;0) e (1;1). Em (b) todos os vetores contidos no espaço unitário [0,1] estão representados. O vetor (0,25;0,75) não pode ser representado na interpretação geométrica da teoria de conjuntos *crisp*.

um **conjunto fuzzy**. Assim, define-se a função de pertinência μ_A de um conjunto *fuzzy* A como sendo do tipo:

$$\mu_A : \mathcal{X} \rightarrow [0, 1]$$

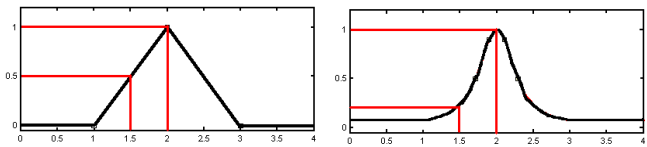
em que $\mathcal{X}$ é o universo de discurso (um conjunto de elementos *crisp*) e $[0, 1]$ define um intervalo infinito de estados cujas extremidades significam a não pertinência de um elemento a um conjunto (0) e a pertinência total de um elemento a um conjunto (1). Os demais números no intervalo significam diferentes graus de pertinência de um elemento a um conjunto.

Diferentes funções de pertinência podem ser definidas no mesmo universo de discurso. Na Figura 2 são exibidos os gráficos de duas funções que modelam a pertinência do conjuntos dos números reais ao conjunto *fuzzy* 2, ou seja, elas modelam o *conjunto de números reais próximos ao número 2*. No eixo das abscissas está representado o universo de discurso, neste caso, o conjunto dos números reais; e no eixo das ordenadas estão os graus de pertinência de cada número real ao conjunto *fuzzy* 2.

Segundo o gráfico da função exibido na Figura 2(a), o número real $1,5$ pertence ao conjunto *fuzzy* 2 com grau de pertinência $0,5$. Em outras palavras, $1,5$ é "igual a 2 com grau de certeza $0,5$". Já na Figura 2(b) o número real $1,5$ possui um grau de pertinência $0,3$ em relação ao referido conjunto fuzzy.

Os conjuntos *fuzzy* definidos por tais funções de pertinência possuem diferenças, mas como representam o mesmo conceito, apresentam algumas similaridades (baseado em [15]):

- em todos eles o número real 2 tem grau de pertinência 1 (pertinência total);



(a) (b)

Figura 2. Duas funções de pertinência diferentes para definir o *conjunto de números reais próximos ao número 2*, usando a noção de conjuntos *fuzzy*. (Adaptada de [15]). Observe-se que funções de pertinência diferentes definem conjuntos *fuzzy* distintos.

- para todos os outros números reais o grau de pertinência é < 1;
- a função de pertinência que os definem é simétrica em relação à reta $x = 2$;
- o função de pertinência é monotonicamente decrescente, de 1 para 0, de acordo com o aumento de $|2 - x|$; e os números reais que estão fora do intervalo $[1, 3]$ (no universo de discurso) não pertencem aos conjuntos ou pertencem com graus de pertinência desprezíveis.

2.5 Medidas de Similaridade

Os algoritmos que executam tarefas de análise de dados geralmente utilizam alguma medida de similaridade entre vetores em seu processo de execução. Essas medidas servem para guiar o processo de construção da superfície de decisão que determinará qual é a região de abrangência de uma classe de dados no caso da tarefa de classificação, ou quais dados pertencem a quais grupos no caso da tarefa de agrupamento.

A similaridade entre dois vetores corresponde a uma medida que compara a igualdade dos mesmos, podendo também ser utilizada como uma forma de medir a distinção entre eles (a dissimilaridade). Normalmente, a similaridade é calculada com base em uma medida de distância entre dois vetores.

Existem diferentes medidas de distância que podem ser aplicadas nos processos de análise de dados, como Distância *Euclidiana*, Distância de *Hamming*, Distância de *Chebyschev*, Distância *City-Block*, Métrica de *Tanimoto*, Métrica de *Mahalanobis* etc. Cada uma delas é mais ou menos adequada a depender do tipo de tarefa a ser resolvida e do tipo de dados utilizado. Para uma revisão detalhada sobre medidas de similaridade veja [9].

Neste tutorial, a medida de similaridade é sempre aplicada visando medir o quão similares são um vetor de dados e um vetor que representa uma classe ou grupo. A Distância Euclidiana é a métrica escolhida para ser a base do cálculo de similaridade entre vetores, que

é calculada como:

$$d(\vec{v}_i, \vec{v}_j) = \left(\sum_{l=1}^{p} (v_{il} - v_{jl})^2 \right)^{\frac{1}{2}} \quad (1)$$

onde p é a dimensão do espaço dos vetores e $\vec{v}_i$ e $\vec{v}_j$ são os vetores sobre os quais se deseja calcular a similaridade.

2.6 Erro de Quantização

Segundo [10] a tarefa de agrupamento (e também de classificação) de dados pode ser entendida como um processo de quantização vetorial. Os vetores protótipos dos grupos (ou classes), são também chamados de vetores quantizadores ou vetores de reconstrução, e representam dados de uma determinada região do espaço.

Uma das formas de avaliar a quantização do espaço obtida mediante a aplicação de um algoritmo de agrupamento (ou classificação) é usando a medida do Erro de Quantização. Essa medida está baseada no cálculo da média das distâncias entre os dados e o vetor que representa a região onde os dados estão localizados, e é calculada como:

$$E_q = \frac{1}{n} \sum_{j=1}^{n} \left\| \vec{x}_j - \vec{C}_i \right\|$$

onde n é o número de dados, $\vec{x}_j$ é um dado e $\vec{C}_i$ é o vetor representante do grupo ou classe à qual o dado $\vec{x}_j$ pertence, sendo que $i = \arg\min_{i=1}^{c} d(\vec{C}_i, \vec{x}_j)$.

2.7 *c-Means*

O algoritmo *c-Means* foi um dos primeiros algoritmos propostos para análise de agrupamento[9]. Nesse algoritmo, c agrupamentos são representados como um conjunto $\mathcal{C} = \{\vec{C}_1, ..., \vec{C}_c\}$ de vetores chamados "protótipos". Cada vetor protótipo sempre está associado à representação de uma classe ou grupo do conjunto de dados e, para isso, deve residir no mesmo espaço $\Re^p$ que os dados do conjunto. O conjunto $\mathcal{C}$ é representado por uma matriz de dimensão $c \times p$.

Para alcançar seu objetivo, o algoritmo realiza várias iterações na busca de uma configuração ótima de parâmetros para minimizar $J_C M(U_h, C)$, que é dado por:

[9] Veja um dos primeiros estudos sobre esse algoritmo, por MacQueen em 1967 [23].

$$J_{CM}(U_h, \mathcal{C}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} d(\vec{C}_i, \vec{x}_j)^2 \tag{2}$$

onde $d(\vec{C}_i, \vec{x}_j)$, é a distância entre o vetor de dados $\vec{x}_j$ e o protótipo do grupo $\vec{C}_i$, c é o número de grupos a ser determinado pelo algoritmo, n é o número de dados no conjunto de dados e U_h é uma matriz binária chamada "matriz de partição", de dimensões $c \times n$, definida como:

$$U_h = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ u_{2,1} & u_{2,2} & \cdots & u_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ u_{i+1,1} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ u_{c,1} & u_{c,2} & \cdots & u_{c,n} \end{bmatrix}$$

Nessa matriz de partição, cada elemento $u_{ij} \in \{0, 1\}$ indica a associação de um dado a um grupo. Um dado $\vec{x}_j$ está associado ao grupo representado pelo protótipo $\vec{C}_i$ se $u_{ij} = 1$, caso contrário, se $u_{ij} = 0$, o dado $\vec{x}_j$ não está associado ao grupo i. Com o processo de minimização, os dados são associados aos grupos de forma que, quanto menores forem as distâncias entre o dado $\vec{x}$ e o vetor protótipo $\vec{C}$ associado a ele, menor é o valor da Equação (2).

Visto que o objetivo de um processo de agrupamento de dados é associar um elemento a um grupo, e que no caso do *c-Means* a base é a teoria de conjuntos *crisp*, é necessário garantir que cada dado esteja associado a exatamente um grupo. Assim, o processo de minimização da Equação (2) deve obedecer à condição:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j \in 1, ..., n.$$

garantindo que a soma das pertinências de um dado $\vec{x}_j$ a todos os grupos em C seja igual a 1, ou seja, cada coluna da matriz de partição deve possuir o valor 1 em **uma e somente uma** célula.

Segundo Kruse, Döfing e Lesot [19], esta restrição garante partições exaustivas e evita a solução trivial para a minimização de J_{CM}, a qual consiste em não associar os dados aos grupos ($u_{ij} = 0 \; \forall i, j$).

Existe uma segunda restrição que precisa ser adicionada ao processo de otimização de J_{CM} para garantir que todos os c grupos tenham, ao menos, um dado associado. Essa restrição é modelada por:

$$\sum_{j=1}^{n} u_{ij} \geq 1, \forall i \in 1, ..., c.$$

tal que cada linha da matriz de partição deve possuir o valor 1 em **pelo menos** uma célula.

As duas próximas equações são implementadas no processo de minimização de J_{CM} que constitui o Algoritmo *c-Means*. A atualização de U_h, dada por:

$$u_{ij}^{t+1} = \begin{cases} 1, & \text{se } i = \arg\min_{i=1}^{c} d(\vec{C}_i, \vec{x}_j) \\ 0, & \text{caso contrário.} \end{cases} \quad (3)$$

onde t é o contador de iterações do processo de otimização e u_{ij}^{t+1} é o valor da pertinência do dados j ao grupo i na iteração $t+1$, faz com que cada dado seja associado ao grupo cujo protótipo é o mais próximo a ele (possui a distância mínima) dentre todos os protótipos. A atualização de $\mathcal{C}$, calculada como:

$$\vec{C}_i^{t+1} = \frac{\sum_{j=1}^{n} u_{ij} \vec{x}_j}{\sum_{j=1}^{n} u_{ij}} \quad (4)$$

estabelece novos vetores protótipos para os grupos de acordo com a média de todos os vetores de dados associados a eles. O numerador da Equação 4 soma, para cada grupo, os vetores de dados associados a eles.

Assumindo a Distância Euclidiana (Equação (1)), o processo de minimização de J_{CM} pode ser definido como no Algoritmo 1[10].

O erro ϵ usado como condição de parada do algoritmo é um limite inferior para a alteração dos vetores protótipos representantes dos grupos: alterações muito pequenas indicam que as mudanças ocorridas nos grupos em formação no algoritmo não alteram, significativamente, a formação de cada grupo, indicando que o processo encontrou um ponto de mínimo.

[10]Na notação do tipo $\mathcal{C}_t$ entenda-se t como o indicativo da versão da variável $\mathcal{C}$ que deve ser considerada, ou seja, a versão da variável $\mathcal{C}$ criada na iteração t. Esta observação vale para todas as ocorrências desta notação nos algoritmos apresentados neste texto.

Algoritmo 1 *c-Means*

Determine a quantidade de partições c;
Determine um valor pequeno e positivo para um erro máximo, ϵ, permitido no processo;
Inicialize o conjunto de protótipos $\mathcal{C}$ aleatoriamente, escolhendo c vetores protótipos dentro do menor intervalo que contém todos os dados do conjunto; ou inicialize tais vetores escolhendo aleatoriamente c dados do conjunto de dados;
Inicialize o contador de iterações t como $t = 0$;
repita
  $t++$;
  Atualize U_h de acordo com a Equação (3);
  Atualize $\mathcal{C}$ de acordo com a Equação (4);
até que $\left\| \mathcal{C}^{(t)} - \mathcal{C}^{(t-1)} \right\| < \epsilon$

O resultado apresentado pelo Algoritmo *c-Means* é fortemente dependente da inicialização do parâmetro c e da inicialização do conjunto de vetores protótipos $\mathcal{C}$. Sendo assim, não é garantido que a otimização realizada pelo algoritmo atingiu um mínimo global. Mínimos locais são frequentemente encontrados nesse processo e, a fim de melhorar esse aspecto do algoritmo, é aconselhável a execução de diferentes instâncias desse processo, com variações na inicialização dos dois parâmetros citados.

2.8 *Learning Vector Quantization*

LVQ é uma arquitetura de RNA, proposta por Teuvo Kohonen, caracterizada pelo uso de um algoritmo de aprendizado competitivo e supervisionado baseado no conceito de distância vetorial como forma de análise de similaridade entre vetores (dados e protótipos)[11]. Seu objetivo é analisar o espaço dos dados dividindo-o em regiões distintas e definindo um vetor protótipo (um neurônio ou um vetor de reconstrução) para cada região. Esse processo também é conhecido como Quantização Vetorial, daí o nome da *Learning Vector Quantization*. As coordenadas dos vetores protótipos são denominadas *pesos sinápticos*.

Basicamente, o algoritmo de aprendizado desta RNA usa a informação de classe como um guia para movimentar os vetores protótipos no espaço dos dados a fim de definir a superfície de decisão de um modelo classificador. Para dar início ao processo de aprendizado desta arquitetura, um conjunto de vetores protótipos é inicializado e a cada um deles é atribuído um rótulo, correspondendo aos rótulos das classes presentes no conjunto de dados. A intenção é que cada um desses vetores já seja previamente associado a uma classe, cabendo ao algoritmo de treinamento a função de encontrar o melhor posicionamento para que cada

[11] Eventualmente, é possível encontrar uma variação do algoritmo de aprendizado desta RNA que não utiliza a informação de classe dos dados, caracterizando-se como aprendizado não supervisionado.

vetor protótipo represente a classe a ele associada. É importante ressaltar que a uma classe é possível associar vários vetores protótipos.

No processo de aprendizado, cada dado do conjunto de dados (vetores de entrada) é comparado aos vetores protótipos por meio da uma medida de similaridade. Se o rótulo de classe do dado é igual ao rótulo do vetor protótipo mais próximo a ele, este protótipo é deslocado na direção da localização vetorial do dado. Caso contrário, se as classes são diferentes, o movimento se dá para a direção contrária à localização vetorial do dado. O uso da informação de classe para direcionar o aprendizado caracteriza a supervisão. A escolha do vetor protótipo mais próximo ao vetor do dado de entrada, dentre todo o conjunto de vetores protótipo, caracteriza o aprendizado competitivo e é formalmente descrita como:

$$I_v = \arg\min_{i=1}^{k} d(\vec{C}_i, \vec{x}_j) \tag{5}$$

onde I_v é o índice do protótipo vencedor (o protótipo mais próximo do dado).

Seja X o conjunto de dados para treinamento da LVQ; $\mathcal{C}_{LVQ} = \{\vec{C}_1, ..., \vec{C}_k\}$ o conjunto de vetores protótipos, também conhecido como neurônios de saída da LVQ; $R_{LVQ} = \{r_1, ..., r_c\}$, com $c \leq k$, o conjunto de rótulos aos quais estão associados cada um dos vetores de dados e dos vetores protótipos. A regra que rege a competição é apresentada na Equação (5). As regras de aprendizado são:

$$\text{if } \vec{x}_j\ (R_{LVQ}) = \vec{C}_{I_v}\ (R_{LVQ}) \text{ então } \vec{C}_{I_v}^{t+1} = \vec{C}_{I_v}^{t} + \alpha[\vec{x}_j - \vec{C}_{I_v}^{t}]; \tag{6}$$

$$\text{if } \vec{x}_j\ (R_{LVQ}) \neq \vec{C}_{I_v}\ (R_{LVQ}) \text{ então } \vec{C}_{I_v}^{t+1} = \vec{C}_{I_v}^{t} - \alpha[\vec{x}_j - \vec{C}_{I_v}^{t}]; \tag{7}$$

onde t é um contador de iterações e α é o coeficiente de aprendizado, $\vec{x}_j\ (R_{LVQ})$ é o rótulo associado ao dado $\vec{x}_j$ e $\vec{C}_{I_v}\ (R_{LVQ})$ é o rótulo associado ao protótipo vencedor. O Algoritmo 2 descreve os passos do processo de treinamento da LVQ.

O coeficiente de aprendizado desempenha um papel importante no processo de aprendizado da LVQ. Para entendê-lo é útil ter uma visão geométrica sobre como se comportam os vetores envolvidos nas regras de aprendizado. Na Figura 3 estão ilustrados cada um dos passos das operações vetoriais executadas nas regras de aprendizado:

- Na Figura 3(a) está representada a operação de subtração ($\vec{x}_j - \vec{C}_{I_v}$, na iteração t), que determina o deslocamento máximo que o vetor protótipo pode realizar.

Algoritmo 2 Algoritmo de treinamento da LVQ

Inicialize o conjunto de protótipos (neurônios) $\mathcal{C}_{LVQ}$ aleatoriamente (ou use algum conhecimento *a priori*);
Determine o rótulo r de cada vetor protótipo (conjunto de rótulos R_{cX1});
Determine o coeficiente de aprendizado α;
enquanto condição de parada é falsa **faça**
  para Todos os dados j, com $j = 1..., n$ **faça**
    Encontre o protótipo vencedor $\vec{C}_{I_v}$ de acordo com a Equação (5);
    Adapte os pesos sinápticos de $\vec{C}_{I_v}$ de acordo com as Equações (6) ou (7) ;
  fim para
  Atualize o coeficiente de aprendizado α (se necessário e desejável);
fim enquanto

- Sobre esse vetor resultante atua o coeficiente de aprendizado determinando o tamanho do deslocamento que o vetor protótipo efetivamente sofrerá (Figura 3(b)). Note que quanto menor for esse coeficiente, menor será o deslocamento do vetor protótipo.

- Nas Figuras 3(c) e 3(d) é mostrado o efeito da terceira operação vetorial (adição ou subtração). No caso da adição, que ocorre quando a classificação sugerida pela LVQ é correta, o vetor resultante da operação está situado entre o vetor protótipo e o vetor de dados, ou seja, houve a movimentação do vetor protótipo na direção do dado[12]. No caso da subtração, que ocorre quando a classificação sugerida pela LVQ é incorreta, o efeito é o contrário.

Durante o treinamento de uma LVQ, o valor do coeficiente de aprendizado deve diminuir, de forma que no início do treinamento os movimentos dos vetores protótipos sejam amplos, maximizando a exploração do espaço dos dados e, ao final do processo, os movimentos realizados pelos vetores protótipos sejam sutis, caracterizando um ajuste fino nos seus posicionamentos. Assim, a atualização do coeficiente de aprendizado pode ser realizada por meio de uma função monotônica decrescente, definida no intervalo [0,1]. Exemplos de funções para atualização do coeficiente de aprendizado são:

$$\alpha^t = \alpha^{(0)} * 1 - (\frac{t}{t_{max}})$$

onde t é o contador da iteração atual do algoritmo de treinamento; $\alpha^{(0)}$ é o coeficiente de aprendizado inicial; t_{max} é o número máximo de iterações do algoritmo; e

[12]Se o coeficiente de aprendizado é ≈ 1 o vetor protótipo se torna $\approx$ vetor de dados. Esta situação pode não ser benéfica por contribuir para a ocorrência do fenômeno de sobreajuste (*overfitting*).

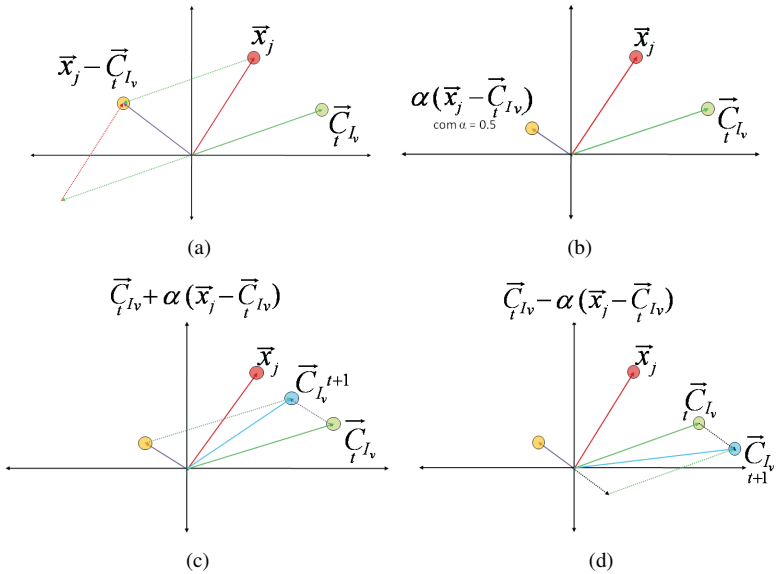


Figura 3. Interpretação geométrica das regras de aprendizado da LVQ.

$$\alpha^t = \alpha^{(0)} * e^{-\frac{t}{\lambda}}$$

onde λ é um parâmetro responsável pela taxa de redução desejada. Como critério de parada para o processo de aprendizado pode-se considerar o número de iterações máxima, o valor mínimo para o coeficiente de aprendizado ou o erro de quantização máximo, entre outros.

A Figura 4 ilustra, no espaço bi-dimensional, alguns passos do processo de treinamento de uma LVQ. A Figura 4(a) mostra o contexto do problema no início do processo de treinamento. O treinamento inicia com um vetor de dados e o vetor protótipo vencedor da competição para este dado. Ambos estão destacados pela ligação feita por uma flecha na Figura 4(b). Na figura seguinte, a flecha indica a direção de deslocamento do vetor protótipo. Nesse caso trata-se de um afastamento pois o vetor protótipo não representa a classe do vetor de dados. Na sequência é apresentado o novo contexto do problema, com o vetor protótipo já deslocado. As duas últimas figuras (Figuras 4(e) e (f)) representam, respectivamente, um contexto do problema em uma iteração intermediária do processo e o contexto do problema na iteração final. Nessa última, observa-se que todos os vetores protótipos se deslocaram até a posição onde representam bem os vetores de dados das classes às quais foram associados.

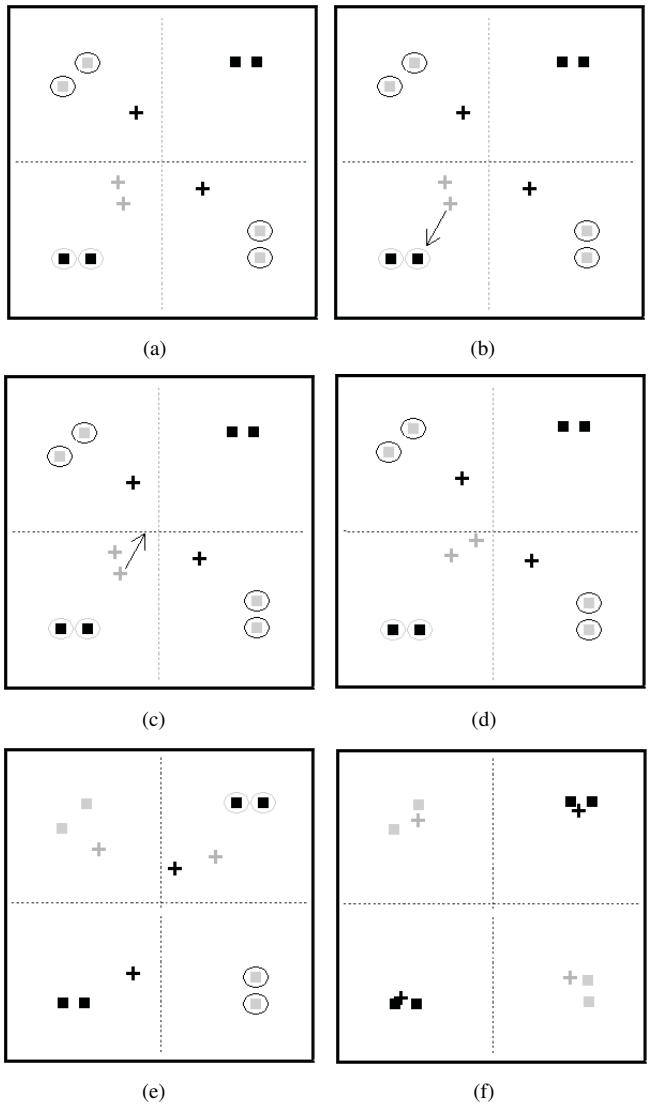


Figura 4. Passos do processo de treinamento de uma LVQ (Adaptada da execução da demonstração "nnd14lv1.m" do Matlab®). Legenda: quadrado: vetor de dados; cruz: vetor protótipo; cor preta: classe 1; cor cinza: classe 2; quadrado circulado: classificação incorreta; quadrado não circulado: classificação correta.

3 Agrupamento *Fuzzy* e *Fuzzy c-Means*

Conforme definido na Seção 2, a tarefa de análise de dados por agrupamento é o processo de agrupar dados de um conjunto de forma que a similaridade entre os dados de um grupo é maximizada enquanto a similaridade entre dados de grupos diferentes é minimizada. Entretanto, na prática, separar dados em grupos pode exigir a consideração de fatores de incerteza e imprecisão, já que alguns dados podem se caracterizar como similares a dados de um ou mais grupos. O tratamento desses fatores pode ser realizado por meio da TCF, transformando o processo de agrupamento clássico em um processo *fuzzy*. Entre os algoritmos existentes para a realização de agrupamento *fuzzy* está o *Fuzzy-c-Means* (FCM), originalmente proposto em Bezdek, em 1981 [2], como uma alternativa ao algoritmo de agrupamento clássico *c-Means*. Esse algoritmo é a base para a inserção de características *fuzzy* aos modelos neurais discutidos neste tutorial e o entendimento de seus princípios é essencial para promover o uso adequado de tais modelos. Esta seção é dedicada a discutir a tarefa de agrupamento *fuzzy* e o algoritmo FCM.

3.1 Agrupamento *Fuzzy*

Formalmente, considerando o conjunto de dados X com n elementos, o número de agrupamentos c e a definição de conjuntos *fuzzy*, como apresentados na Seção 2, no agrupamento *fuzzy* os c grupos *fuzzy* formam uma partição *fuzzy* C_f de X. C_f é uma matriz $c \times n$ cujos valores são por:

$$\mu_{ij} = \mu_i(\vec{x}_j), 1 \geq i \geq c, 1 \geq j \geq n$$

satisfazendo três condições:

$$0 \leq \mu_{ij} \leq 1, \forall i, j \tag{8}$$

$$\sum_{i=0}^{c} \mu_{ij} = 1, \forall j \tag{9}$$

$$0 < \sum_{j=0}^{n} \mu_{ij} < n, \forall i. \tag{10}$$

onde μ_{ij} é o grau de pertinência de $\vec{x}_j$ ao i-ésimo grupo *fuzzy* de X.

A primeira restrição trata da questão de normalidade de conjuntos *fuzzy*. A segunda restrição trata da divisão da pertinência do dado em pertinências a um ou vários grupos. A terceira restrição impede que qualquer grupo seja vazio (sem dados associados a ele); e que um grupo tenha associado a ele todos os dados do conjunto com pertinência máxima (1). Se todos os valores de μ_{ij} pertencerem ao conjunto $\{0, 1\}$, então os grupos são definidos como "grupos *crisp*" do conjunto X [3], [4].

Bezdek et al. [3] exemplificou o processo de agrupamento *fuzzy* por meio de objetos, como os contidos na Figura 5(a). Para agrupar esse conjunto de objetos, três grupos foram criados: das MAÇÃS, das PERAS e das LARANJAS. A Figura 5(b) ilustra um processo de agrupamento gráfico (ou espacial) desses objetos, cuja metodologia consiste em: colocar sobre cada objeto um rótulo que indica a qual grupo natural ele pertence, onde **P** = Pera (objeto com a marca **P**, pertencente ao grupo natural das Peras), **M**=Maçã, **L**=Laranja, com exceção do objeto oval **O3**, um LIMÃO, que representa uma anomalia nos dados, frente a rotulação definida neste exemplo.

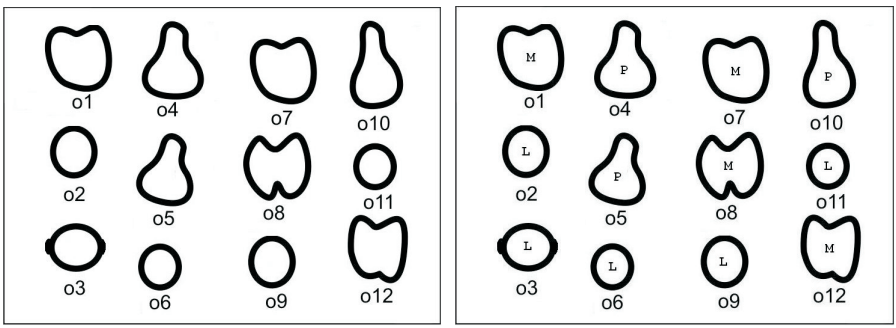


Figura 5. Objetos separados em grupos naturais: (a) Objetos físicos para agrupamento. (b) Agrupamento Gráfico. Adaptado de [3].

As Tabelas 1 e 2 representam, respectivamente, as matrizes U_h (matriz de partições clássica) e U_f (matriz de partições *fuzzy*, discutida na Seção 3.2), resultantes dos processos de agrupamento *crisp* e do agrupamento *fuzzy*. Note que no agrupamento *crisp* o objeto **O3** foi definido como sendo uma Laranja (objeto pertence ao grupo das Laranjas com grau de pertinência 1). No agrupamento *fuzzy*, esse mesmo objeto foi mais bem agrupado como Laranja (o maior grau de pertinência para este objeto é 0,85), entretanto, ele também possui características de Maçã e de Pera, visto que pertence a esses grupos com graus de pertinência 0,09 e 0,06, respectivamente.

Diferentes graus de pertinência de um dado a diversos grupos permitem diferentes

	O1	O2	**O3**	O4	O5	O6	O7	O8	O9	O10	O11	O12
P	0	0	**0**	1	1	0	0	0	0	1	0	0
L	0	1	**1**	0	0	1	0	0	1	0	1	0
M	1	0	**0**	0	0	0	1	1	0	0	0	1
Total	1	1	**1**	1	1	1	1	1	1	1	1	0

Tabela 1. Resultado do Processo de Agrupamento *Crisp*.

	O1	O2	**O3**	O4	O5	O6	O7	O8	O9	O10	O11	O12
P	0,05	0	**0,06**	0,93	0,92	0	0	0	0,09	0,75	0,10	0,1
L	0	0,97	**0,85**	0	0	0,99	0,21	0,19	0,82	0,13	0,80	0,25
M	0,95	0,03	**0,09**	0,07	0,08	0,01	0,79	0,81	0,09	0,12	0,10	0,65
Total	1	1	**1**	1	1	1	1	1	1	1	1	1

Tabela 2. Resultado do Processo de Agrupamento *Fuzzy*.

interpretações de agrupamento, o que não acontece no processo de agrupamento *crisp*. Além disso, essa modelagem *fuzzy* produziu uma solução com estrutura mais rica e flexível que a solução dada pelo modelo *crisp*.

3.2 *Fuzzy-c-Means*

O algoritmo FCM tem como objetivo encontrar grupos *fuzzy* para um conjunto de dados. Para alcançar este objetivo, o algoritmo precisa minimizar uma função que diz respeito à minimização das distâncias entre os dados e os centros dos grupos aos quais tais dados pertencem com algum grau de pertinência [46]. A minimização da função citada deve produzir grupos mais adequados (melhores ou mais naturais) do que aqueles produzidos pelo algoritmo *c-Means* clássico. Tal função é dada por:

$$J_{FCM}\left(U_f, \mathcal{M}\right) = \sum_{i=1}^{c}\sum_{j=1}^{n}\left(\mu_{ij}\right)^m d(\vec{m}_i, \vec{x}_j)^2 \tag{11}$$

onde,

- n, c e μ_{ij} foram definidos anteriormente neste tutorial;
- m é um número real positivo, tal que $m \in (1, \infty)$, utilizado como parâmetro de "fuzificação" e discutido mais à frente neste texto;

- $d(\vec{m}_i, \vec{x}_j)$ é a distância entre o vetor de dados $\vec{x_j}$ e o vetor protótipo representante da i-ésima partição *fuzzy* $\vec{m_i}$;
- U_f é uma matriz de pertinências *fuzzy*, de dimensões $c \times n$, definida de acordo com:

$$U_f = \begin{bmatrix} \mu_{1,1} & \mu_{1,2} & \cdots & \mu_{1,n} \\ \mu_{2,1} & \mu_{2,2} & \cdots & \mu_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{i,1} & \mu_{i,2} & \cdots & \mu_{i,n} \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{i+1,1} & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ \mu_{c,1} & \mu_{c,2} & \cdots & \mu_{c,n} \end{bmatrix}$$

A construção e atualização desta matriz acontecem durante o processo de minimização da Equação (11), seguindo o disposto em[13]:

$$\mu_{ij}^{t+1} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C_l}, \vec{x}_j)}{d(\vec{C_i}, \vec{x}_j)} \right)^{\frac{1}{1-m}}} \tag{12}$$

sendo que duas exceções devem ser tratadas:

$$\mu_{ij}^{t+1} = \begin{cases} 1 & \text{se } d(\vec{C_i}, \vec{x}_j) = 0 \\ 0 & \text{se } d(\vec{C_l}, \vec{x}_j) = 0, (l \neq i \text{ e } 1 \leq l, i \leq c) \end{cases}$$

- $\mathcal{M}$ é uma matriz de protótipos de grupos. $\vec{m_i}$ são vetores protótipos. A matriz possui dimensões $c \times p$ e é definida por:

$$\mathcal{M} = \begin{bmatrix} \vec{m_1} \\ \vec{m_2} \\ \cdots \\ \vec{m_c} \end{bmatrix} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,p} \\ m_{i,1} & m_{i,2} & \cdots & m_{i,p} \\ m_{i+1,1} & \cdots & \cdots & \cdots \\ m_{c,1} & m_{c,2} & \cdots & m_{c,p} \end{bmatrix}$$

A inicialização dessa matriz é aleatória e sua atualização ocorre durante o processo de minimização da Equação (11), seguindo o disposto em[14]:

[13] Esta atualização está discutida em detalhes na Seção 3.3.
[14] Esta atualização está discutida em detalhes na Seção 3.4.

$$\vec{m}_i^{t+1} = \frac{\sum_{j=1}^{n} \left(\mu_{ij}^{t+1}\right)^m \vec{x}_j}{\sum_{j=1}^{n} \left(\mu_{ij}^{t+1}\right)^m} \tag{13}$$

onde $i = 1, ..., c$ e $j = 1, ..., n$, t é um contador de iterações inicializado em 0 e $\vec{x}_j$ é o vetor representante do dado x_j que é analisado em relação às suas pertinências aos grupos.

Assumindo a Distância Euclidiana (Equação (1)), o processo de execução do FCM pode ser definido, segundo [3], como descrito no Algoritmo 3.

Algoritmo 3 *Fuzzy-c-Means*

Determine o valor do parâmetro de "fuzificação" m;
Determine a quantidade de partições *fuzzy* c;
Determine um valor pequeno e positivo para o erro máximo, ϵ, permitido no processo;
Inicialize a matriz de protótipos $\mathcal{M}$ aleatoriamente;
Inicialize o contador de iterações t como $t = 0$;
repita
  $t++$;
  Atualize U_f de acordo com a Equação (12);
  Atualize $\mathcal{M}$ de acordo com a Equação (13);
até que $\left\| \mathcal{M}^{(t)} - \mathcal{M}^{(t-1)} \right\| < \epsilon$

A definição dos valores iniciais para os parâmetros m, c e $\mathcal{M}$ influencia o resultado a ser obtido ao final do processo e, devido ao caráter aleatório ou subjetivo de definição dos mesmos, várias execuções do algoritmo devem ser realizadas para que se tenha um panorama da sua robustez na resolução de um problema específico. Além disso, entender o processo de atualização da matriz de pertinências *fuzzy* U_f e da matriz de protótipos $\mathcal{M}$ é essencial para que se tenha condições de trabalhar adequadamente com o algoritmo *Fuzzy c-Means*. Os processos de atualização de tais matrizes são discutidos na sequência.

3.3 Estudo da atualização da Matriz de Pertinências *Fuzzy* U_f

O processo de atualização de cada elemento μ_{ij} da matriz U_f (Equação (12)) mantém o índice i no grupo ao qual se quer calcular a pertinência do dado j, enquanto varia o índice l em todos os grupos. Isso significa que a pertinência de um dado j a um grupo i é dependente das relações desse dado com todos os demais grupos. A influência dessas relações no cálculo das pertinências μ_{ij} é ponderada por meio do parâmetro m.

De acordo com [46], a definição de $m = 2$ permite a seguinte simplificação da Equação (12):

$$\mu_{ij}^{t+1} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)}\right)^{\frac{1}{(1-2)}}} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)}\right)^{-1}} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_i, \vec{x}_j)}{d(\vec{C}_l, \vec{x}_j)}\right)}$$

resultando em:

$$\mu_{ij}^{t+1} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_i, \vec{x}_j)}{d(\vec{C}_l, \vec{x}_j)}\right)} \tag{14}$$

Desta forma, se o parâmetro m é valorado com 2, o grau de pertinência μ_{ij} é obtido unicamente em função das razões entre as distâncias e o dado j e os centros de grupos.

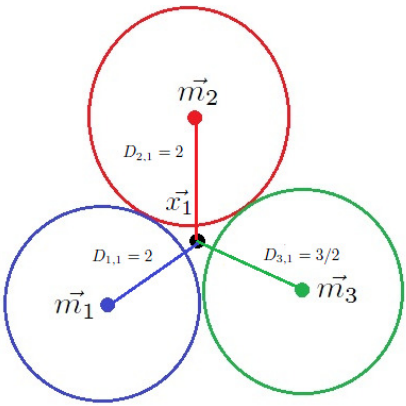
Caso a distância entre o dado j e o centro do grupo i esteja próxima de zero, o grau de pertinência μ_{ij} tenderá a 1, que significa que o dado j tem alta pertinência àquele grupo, como mostrado por:

$$\lim_{d(\vec{C}_i, \vec{x}_j) \to 0} \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_i, \vec{x}_j)}{d(\vec{C}_l, \vec{x}_j)}\right)} = \lim_{d(\vec{C}_i, \vec{x}_j) \to 0} \frac{1}{\sum_{l=1 \wedge l \neq i}^{c} \left(\frac{d(\vec{C}_i, \vec{x}_j)}{d(\vec{C}_l, \vec{x}_j)}\right) + \frac{d(\vec{C}_i, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)}} =$$

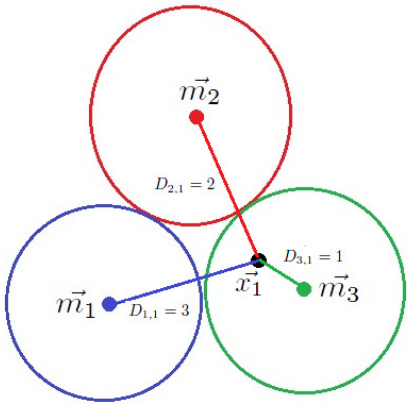
$$= \frac{1}{0+1} = 1$$

Para o caso em que $d(\vec{C}_i, \vec{x}_j) \neq 0$, é interessante analisar o efeito das razões entre $d(\vec{C}_i, \vec{x}_j)$ e $d(\vec{C}_l, \vec{x}_j)$ no valor dos termos da somatória (Equação (14)) e consequentemente nos valores das pertinências μ_{ij}. Quanto maior for o resultado de uma razão dentro da somatória, maior será sua influência sobre a soma total, contribuindo para a redução do grau de pertinência μ_{ij}. Isso é equivalente a afirmar que, quanto maior for a distância de um dado j ao centro de um grupo i menor será o seu grau de pertinência a esse grupo.

Estas situações são ilustradas na Figura 6. Nota-se que nas situações em que a distância entre o dado j e o centro do grupo i é **menor** que a distância entre o dado j e outros centros dos outros grupos, a pertinência μ_{ij} é **maior**. O caso contrário também é válido.



(a)



(b)

$$\mu_{1,1} = \frac{1}{\frac{D_{1,1}}{D_{1,1}} + \frac{D_{1,1}}{D_{2,1}} + \frac{D_{1,1}}{D_{3,1}}} = \frac{1}{\frac{2}{2} + \frac{2}{2} + \frac{2}{\frac{3}{2}}} = \frac{1}{\frac{10}{3}} = \frac{3}{10} = 0,300$$

$$\mu_{2,1} = \frac{1}{\frac{D_{2,1}}{D_{1,1}} + \frac{D_{2,1}}{D_{2,1}} + \frac{D_{2,1}}{D_{3,1}}} = \frac{1}{\frac{2}{2} + \frac{2}{2} + \frac{2}{\frac{3}{2}}} = \frac{1}{\frac{10}{3}} = \frac{3}{10} = 0,300$$

$$\mu_{3,1} = \frac{1}{\frac{D_{3,1}}{D_{1,1}} + \frac{D_{3,1}}{D_{2,1}} + \frac{D_{3,1}}{D_{3,1}}} = \frac{1}{\frac{\frac{3}{2}}{2} + \frac{\frac{3}{2}}{2} + \frac{\frac{3}{2}}{\frac{3}{2}}} = \frac{1}{\frac{10}{4}} = \frac{4}{10} = 0,400$$

(c)

$$\mu_{1,1} = \frac{1}{\frac{D_{1,1}}{D_{1,1}} + \frac{D_{1,1}}{D_{2,1}} + \frac{D_{1,1}}{D_{3,1}}} = \frac{1}{\frac{3}{3} + \frac{3}{2} + \frac{3}{1}} = \frac{1}{\frac{33}{6}} = \frac{6}{33} = 0,182$$

$$\mu_{2,1} = \frac{1}{\frac{D_{2,1}}{D_{1,1}} + \frac{D_{2,1}}{D_{2,1}} + \frac{D_{2,1}}{D_{3,1}}} = \frac{1}{\frac{2}{3} + \frac{2}{2} + \frac{2}{1}} = \frac{1}{\frac{22}{6}} = \frac{6}{22} = 0,273$$

$$\mu_{3,1} = \frac{1}{\frac{D_{3,1}}{D_{1,1}} + \frac{D_{3,1}}{D_{2,1}} + \frac{D_{3,1}}{D_{3,1}}} = \frac{1}{\frac{1}{3} + \frac{1}{2} + \frac{1}{1}} = \frac{1}{\frac{11}{6}} = \frac{6}{11} = 0,546$$

(d)

Figura 6. Exemplo do cálculo de μ_{ij} no FCM, com $m = 2$: (a) e (b) são gráficos representando as distâncias similares (a) e discrepantes (b), entre o dado e os centros dos clusters; (c) e (d) são os cálculos referentes aos gráficos (a) e (b), respectivamente.

Por outro lado, assumindo $m \neq 2$, observa-se que existe uma alteração na influência das relações entre as distâncias dos dados aos centros dos grupos, de acordo com o valor resultante no expoente $\frac{1}{(1-m)}$. Para $(m \to \infty)$, μ_{ij} não é calculado em função das distâncias entre dados e os centros dos grupos, e sim em função da quantidade de grupos c, como demostrado em:

$$\lim_{m\to\infty} \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)} \right)^{\frac{1}{1-m}}} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)} \right)^{0}} = \frac{1}{\sum_{l=1}^{c} 1} = \frac{1}{c} \tag{15}$$

Consequentemente, o dado terá o mesmo grau de pertinência a todos os grupos, conforme comentado em [4].

A fim de melhor entender o comportamento do algoritmo mediante a parametrização de m, um exemplo numérico é aqui desenvolvido. A Tabela 3 mostra o efeito causado pela variação de m na formação dos grupos *fuzzy* da Figura 6(b). Observe que para uma mesma situação, a pertinência do dado $\vec{x}_1$ aos grupos representados pelos protótipos $\vec{m}_1$, $\vec{m}_2$, $\vec{m}_3$ varia.

Parâmetro de fuzificação m	$\mu_{1,1}$	$\mu_{2,1}$	$\mu_{3,1}$
3	$0,253$	$0,309$	$0,438$
2	$0,182$	$0,273$	$0,545$
$3/2$	$0,081$	$0,184$	$0,735$

Tabela 3. Influência do parâmetro de fuzificação m nos graus de pertinência de um dado aos grupos. Com valores de m mais altos, as diferenças entre os graus de pertinências são atenuadas.

Assim, analisando o intervalo dos possíveis valores de m ($(1, \infty)$), observa-se que o valor 2 atua como um ponto de referência. À medida que o valor de m aumenta, o módulo do denominador da fração $\frac{1}{1-m}$ também aumenta, diminuindo, portanto, a diferença entre os valores de cada parcela do somatório e consequentemente diminuindo a diferença entre os valores de μ_{ij}. Inversamente, a medida que m se aproxima de 1, cada parcela do somatório passa a ter um expoente e não um radical, aumentando o valor das parcelas e por consequência a diferença entre os valores de μ_{ij}.

Na Seção 3.4 é mostrado como tal variação tem influência direta sobre a atualização da matriz $\mathcal{M}$ e por isso tem também influência sobre a formação final dos grupos ao final do processo de agrupamento.

3.4 Estudo da atualização da Matriz de Protótipos $\mathcal{M}$

O processo de atualização de cada elemento da matriz de protótipos $\mathcal{M}$ (Equação (13)) é baseada nos graus de pertinência dos dados a um grupo, especificamente aquele cujo vetor protótipo relacionado está sendo alterado. Nesse processo, o índice i é fixo em um grupo enquanto o índice j varia em todos os dados do conjunto de dados. Isso significa que a posição de um vetor $\vec{m}_i$ é dependente das posições dos vetores de dados $\vec{x}_j$ e de seus graus de pertinências aos grupos i (μ_{ij})[15]. A influência destas relações no cálculo da posição final de $\vec{m}_i$ é ponderada pelo parâmetro de fuzificação m.

Para entender a influência dos graus de pertinência μ_{ij} na atualização dos vetores protótipo $\vec{m}_i$, a Tabela 3 é estendida (Tabela 4) de forma a ilustrar que, quanto maior for o grau de pertinência do dado ao grupo, maior é a sua contribuição na operação vetorial (maior é a influência/peso exercida(o) pelo dado no deslocamento do vetor protótipo) realizada na Equação (13), que reposiciona o vetor protótipo $\vec{m}_i$ sob alteração.

Parâmetro de fuzificação m	$\mu_{1,1}$	**Peso no deslocamento de** $\vec{m}_3$	$\mu_{2,1}$	**Peso no deslocamento de** $\vec{m}_3$	$\mu_{3,1}$	**Peso no deslocamento de** $\vec{m}_3$
3	$0,253$	$0,016$	$0,309$	$0,030$	$0,438$	$0,084$
2	$0,182$	$0,033$	$0,273$	$0,075$	$0,545$	$0,297$
$3/2$	$0,081$	$0,023$	$0,184$	$0,079$	$0,735$	$0,630$

Tabela 4. Influência dos graus de pertinência no reposicionamento do vetor $\vec{m}_3$.

Já a influência do parâmetros de fuzificação m sobre o reposicionamento do vetor $\vec{m}_3$ é melhor entendida considerando diferentes possibilidades de graus de pertinência de um dado a um grupo, como na Tabela 5.

De acordo com os dados apresentados na Tabela 5, quanto maior é o valor do parâmetro m mais suave será o reposicionamento dos vetores na matriz $\mathcal{M}$, visto que a ponderação de m faz com que diminua a influência dos dados sobre tal reposicionamento.

A Tabela 6 mostra o resultado da execução do FCM sobre um conjunto de dados didático (veja a representação gráfica na Figura 7). Observe que existe uma tendência a aumentar o erro de quantização quando se aumenta o valor do parâmetro m e, para altos valores de m, os protótipos tendem a se aproximar do centro do conjunto de dados, como ilustrado na Figura 7(d) e (e). Portanto, é plausível afirmar que para valores altos de m

[15] A operação vetorial executada nesse processo tem uma interpretação geométrica similar àquela executada no processo de aprendizado da LVQ (veja Figura 3).

Parâmetro de fuzificação m	$\mu_{1,1}$ $0,253$	**Peso no deslocamento de** $\vec{m}_3$	$\mu_{2,1}$ $0,309$	**Peso no deslocamento de** $\vec{m}_3$	$\mu_{3,1}$ $0,438$	**Peso no deslocamento de** $\vec{m}_3$
7		$0,000$		$0,000$		$0,003$
3		$0,016$		$0,030$		$0,084$
2		$0,064$		$0,095$		$0,192$
$1,5$		$0,127$		$0,172$		$0,290$
$1,01$		$0,250$		$0,305$		$0,434$

Tabela 5. Influência do parâmetro de fuzificação m no reposicionamento do vetor $\vec{m}_3$.

o algoritmo apresenta resultados com grupos menos bem definidos. Mais exemplos sobre resultados obtidos com a variação desses parâmetros são mostrados na Seção 5.

Parâmetros de fuzificação m	**Número de Iterações**	**Erro de Quantização**	**Parâmetros de fuzificação** m	**Número de Iterações**	**Erro de Quantização**
1	0	Indefinido	4	18	$0,5136$
1.1	2	$0,0838$	4.5	16	$0,5137$
1.5	4	$0,0839$	5	15	$0,5139$
2	7	$0,0852$	5.5	14	$0,5139$
2.5	11	$0,0904$	6	14	$0,5141$
3	38	$0,1018$	6.5	13	$0,5141$
3.5	22	0.5135	7	13	$0,5141$

Tabela 6. Execução do algoritmo FCM, com variação no parâmetro de fuzificação m. A condição de parada para estas execuções é uma movimentação mínima na autalização da matriz $\mathcal{M}$.

O comportamento discutido acima pode ser formalmente verificado. De acordo com Bezdek et al. [3], para atualização da matriz $\mathcal{M}$, executada por meio da Equação (13), caso o valor de m tenda a infinito, os novos valores de $\vec{m}_i$ serão a posição média dos dados pertencentes àquele grupo, conforme demonstrado por:

$$\lim_{m\to\infty} \frac{\sum_{j=1}^{n} \left(\mu_{ij}^{t+1}\right)^m \vec{x}_j}{\sum_{j=1}^{n} \left(\mu_{ij}^{t+1}\right)^m} = \lim_{m\to\infty} \frac{\sum_{j=1}^{n} \left(\frac{1}{c}\right)^m \vec{x}_j}{\sum_{j=1}^{n} \left(\frac{1}{c}\right)^m} = \lim_{m\to\infty} \frac{\left(\frac{1}{c}\right)^m \sum_{j=1}^{n} \vec{x}_j}{\left(\frac{1}{c}\right)^m \sum_{j=1}^{n} 1} = \frac{\sum_{j=1}^{n} \vec{x}_j}{n} \quad (16)$$

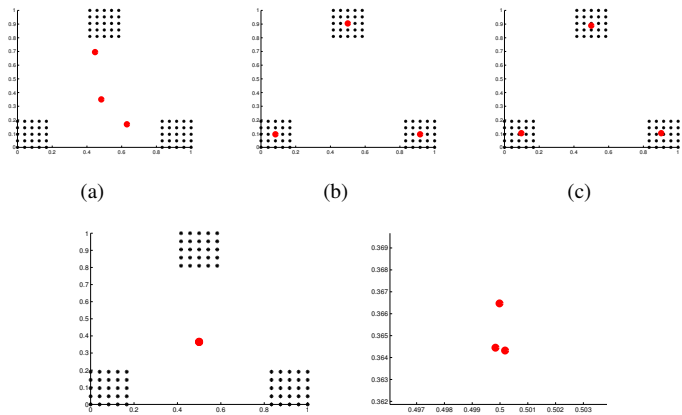


Figura 7. Plotagem do conjunto de dados (cruzes) e os protótipos representantes de grupos (círculos): (a) Condições iniciais; (b) Resultado para $m = -1, 1$; (c) Resultado para $m = 2$; (d) Resultado para $m = 7$; (e) Visão mais detalhada para resultado para $m = 7$.

onde $\left(\mu_{ij}^{t+1}\right)^m = \frac{1}{c}$ de acordo com a Equação (15).

4 *Fuzzy Learning Vector Quantization*

Nesta seção são discutidos dois modelos de Redes Neurais *Fuzzy* baseados na LVQ. A primeira, proposta em [5], caracteriza-se como um modelo de aprendizagem supervisionada cujo processo é bastante similar ao processo da LVQ clássica. A segunda, proposta em [4], caracteriza-se por um processo de aprendizagem não supervisionada, tornando o processo de aprendizado da LVQ similar ao próprio FCM ou mesmo a um modelo simplificado de uma rede neural auto-organizável como SOM[16].

4.1 *Fuzzy* LVQ com aprendizagem supervisionada

Conforme discutido em [5], identificam-se dois objetivos principais na execução de uma LVQ: (a) maximização da taxa de classificações corretas e; (b) minimização do erro de quantização.

[16]Do inglês *Self-Organizing Map* (SOM), rede neural artificial baseada em aprendizado auto-organizável competitivo, também proposta por Teuvo Kohonen [16].

Motivados pelo alcance efetivo desses objetivos e pelas facilidades oferecidas pelo algoritmo FCM, Chung e Lee definem em [5] uma função objetivo para o modelo *Fuzzy* LVQ. Esta função é dada por:

$$Q^m(U_f, \mathcal{M}) = \sum_{j=1}^{n} Q_j^m(U_f, \mathcal{M}) = \sum_{j=1}^{n}\sum_{i=1}^{c} [\tau_{ij} - (\mu_{ij})^m]\, d(\vec{C}_i, \vec{x}_j)^2$$

onde:

- os termos m, U_f, $\mathcal{M}$, n, c e $d(\vec{C}_i, \vec{x}_j)$ foram definidos anteriormente neste tutorial;
- $i = 1, ..., c$ e $j = 1, ..., n$;
- as condições de agrupamento *fuzzy*, expressas nas Equações (8), (9) e (10), devem ser obedecidas;
- τ_{ij} é o grau de pertinência esperado do dado x_j à classe i, representando o rótulo de classe que será utilizado no modelo de aprendizado supervisionado. Tais graus de pertinência esperados estão organizados em uma matriz Γ, de dimensões $c \times n$.

A otimização dessa função considera a minimização das a) diferenças entre os graus de pertinência esperados (τ_{ij}) e os graus de pertinência obtidos (μ_{ij}) e; b) distâncias entre os dados e os centros das classes procuradas; que correspondem, respectivamente, aos objetivos que Chung identifica para a LVQ.

O processo de execução da *Fuzzy LVQ* (minimização do função objetivo) está definido no Algoritmo 4. Comparativamente aos algoritmos FCM e de treinamento da LVQ, esse algoritmo, respectivamente:

- realiza as atualizações das matrizes de pertinência U_f e de protótipos $\mathcal{M}$, de maneira similar ao algoritmo FCM.
- faz uso de classificações de dados, conhecidas *a priori*, num processo de aprendizado supervisionado como no algoritmo da LVQ;

Para definição da matriz Γ, é considerado que um dado pertença a apenas uma classe, então o respectivo grau de pertinência deve ser valorado com 1, enquanto os demais referentes a esse dado devem ser valorados com 0. O critério de parada é definido segundo o projetista do modelo, que poderá aplicar, assim como no FCM, o conceito do erro máximo permitido

Algoritmo 4 Algoritmo de treinamento da *Fuzzy* LVQ supervisionada

```
Determine o valor do parâmetro de "fuzificação" m;
Determine o valor do coeficiente de aprendizado inicial da rede α^(0);
Determine a quantidade de classes c;
Determine o critério de parada a ser utilizado;
Defina a matriz de pertinências esperadas Γ;
Inicialize a matriz de protótipos M aleatoriamente ou com algum conhecimento a priori;
Inicialize o contador de iterações t;
enquanto condição de parada é falsa faça
  para Cada dado j, com j = 1..., n faça
    para Cada vetor protótipo i, com i = 1, ..., c faça
      Atualize U_f, segundo a Equação (17);
      Aualiza M, segundo a Equação (18);
    fim para
  fim para
  Atualize a taxa de aprendizado α;
  Incremente o contador de iterações t;
fim enquanto
```

ϵ, ou como definido no processo de aprendizado da LVQ, um valor mínimo para a taxa de aprendizado α, ou ainda, um valor máximo para o número de iterações t.

Após estas definições, inicia-se o processo de minimização que constitui o algoritmo da *Fuzzy LVQ*, por meio da atualização das matrizes U_f e $\mathcal{M}$. Para atualização da matriz de pertinências U_f aplica-se (adaptado de [5]):

$$u_{ij}^{t+1} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_i, \vec{x}_j)}{d(\vec{C}_l, \vec{x}_j)} \right)^{\frac{1}{(m-1)}}} \tag{17}$$

para $i = 1, ..., c$ e $j = 1, ..., n$.

A equivalência das Equações (17) e (12) é mostrada por:

$$u_{ij}^{t+1} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_i, \vec{x}_j)}{d(\vec{C}_l, \vec{x}_j)} \right)^{\frac{1}{(m-1)}}} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)} \right)^{\frac{1}{(-1)(m-1)}}} =$$

$$= \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)} \right)^{\frac{1}{(-m+1)}}} = \frac{1}{\sum_{l=1}^{c} \left(\frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)} \right)^{\frac{1}{(1-m)}}}$$

portanto, os estudos já apresentados sobre o parâmetro m também se aplicam no presente caso.

A atualização da matriz $\mathcal{M}$, que corresponde à regra de aprendizado do modelo, ocorre por meio da movimentação de todos os vetores protótipos representantes de classes $\vec{m}_i$, seguindo o disposto em:

$$\vec{m}_i^{\,(t+1)} = \vec{m}_i^{\,(t)} + \alpha^t \left[\tau_{ij} - (u_{ij})^m \right] \left[\vec{x}_j - \vec{m}_i^{\,(t)} \right] \tag{18}$$

onde α é a taxa de aprendizado, a qual sofrerá atualizações de formas análogas àquelas aplicadas no processo clássico da LVQ. Por meio da Equação (18) nota-se que a atualização de $\mathcal{M}$ é calculada em função do termo $[\tau_{ij} - (u_{ij})^m]$. Semelhantemente ao processo de aprendizado da LVQ, caso esse termo seja negativo, o centro da classe (o protótipo) será afastado do dado, caso contrário, será atraído pelo dado. A inclusão deste termo "fuzificou" a regra de aprendizado da LVQ, considerando que:

- a direção de movimento dos vetores protótipos é determinada de acordo com o sinal da diferença entre os graus de pertinência atuais dos dados às classes;
- a intensidade do movimento dos vetores protótipos é ponderada tanto pelo coeficiente de aprendizado quanto pela diferença entre as pertinências verificadas e as pertinências esperadas;
- as pertinências verificadas são ponderadas pelo parâmetro de fuzificação m e as discussões já realizadas em relação a estes parâmetros podem ser mapeadas para o presente caso: quanto maior é o valor de m mais suave é a influência das pertinências verificadas no processo de aprendizado.

4.2 *Fuzzy* LVQ com aprendizagem não supervisionada

Bezdek et al. [3] definiram um algoritmo para a *Fuzzy LVQ* que não utiliza os conceitos do aprendizado supervisionado, pois não utiliza o rótulo de classe em sua execução. Esse algoritmo possui duas importantes características: (a) utiliza os protótipos não vencedores da competição na determinação da taxa de aprendizado, introduzindo no processo uma ideia de vizinhança similar à existente no algoritmo SOM; (b) automatiza, até certo ponto, a escolha do valor do parâmetro de fuzificação m.

Diferentemente do que acontece no FCM e na *Fuzzy* LVQ supervisionada, o valor do parâmetro de fuzificação é alterado durante o processo de execução do algoritmo de treinamento da *Fuzzy* LVQ não supervisionada. A maneira como ele é alterado (de forma crescente ou descrescente, ou sem alteração) determina em qual família das *Fuzzy* LVQ o algoritmo será classificado. Tais famílias são nomeadas da seguinte forma:

- *Fuzzy* LVQ Decrescente - $\downarrow$ *Fuzzy LVQ*, onde $m^{(0)} > m^{(f)}$;
- *Fuzzy* LVQ Crescente - $\uparrow$ *Fuzzy LVQ*, onde $m^{(0)} < m^{(f)}$;
- *Fuzzy* LVQ - *Fuzzy* LVQ constante, onde $m^{(0)} = m^{(f)}$.

onde $m^{(0)}$ e $m^{(f)}$ denotam, respectivamente, os valores inicial e final definidos para o parâmetro de fuzificação m, antes da execução do algoritmo. Após inspeções experimentais, Bezdek et al. [3] recomendam que os valores de $m^{(0)}$ e $m^{(f)}$ sejam valorados no intervalo]1,01 , 7[. Dessa forma, no caso da $\downarrow$ *Fuzzy LVQ*, o seguinte critério deve ser obedecido: $7 > m^{(0)} > m^{(f)} > 1,01$. A regra proposta como forma de atualizar esse parâmetro em função do tempo de execução do algoritmo de treinamento, é expressa por:

$$m^{(t)} = m^{(0)} + t\frac{m^{(f)} - m^{(0)}}{T}; \tag{19}$$

onde: t é o contador de iterações do algoritmo; T é o número máximo de iterações; $m^{(f)}$,$m^{(0)}$, $m^{(t)}$, representam, respectivamente, os valores final, inicial e na iteração t, do parâmetro de fuzificação m. Note que esta regra é aplicável em qualquer uma das famílias da *Fuzzy* LVQ não supervisionada.

Na *Fuzzy* LVQ proposta por Bezdek et al. [3] não existe atualização explícita da Matriz de Pertinência U_f, como ocorre no FCM e na *Fuzzy* LVQ supervisionada. Na *Fuzzy* LVQ não supervisionada, o valor do coeficiente de aprendizado corresponde ao valor do grau de pertinência do dado ao grupo, ponderado pelo parâmetro m, conforme demonstrado em:

$$\alpha Fuzzy_{ij}^{(t)} = \mu_{ij}^{m(t)} = \frac{1}{\sum_{l=1}^{c} \frac{d(\vec{C}_l, \vec{x}_j)}{d(\vec{C}_i, \vec{x}_j)}^{\frac{1}{1-m}}}^{m(t)} \tag{20}$$

Note que no algoritmo FCM, os graus de pertinência do dado ao grupo também são ponderados pelo parâmetro m (conforme pode ser visto na Equação(13)).

Assim como no processo da *Fuzzy* LVQ supervisionada, a atualização dos vetores protótipos (Matriz $\mathcal{M}$) $\vec{m_i}$ ocorre em função da posição atual do vetor, das taxas de aprendizado calculadas anteriormente, e das distâncias entre vetores representantes dos dados $\vec{x_j}$ e centros dos grupos $\vec{m_i}$ no instante $t-1$, conforme demonstrado em:

$$\vec{m_i}^{(t)} = \vec{m_i}^{(t-1)} + \frac{\sum_{j=1}^{n} \alpha Fuzzy_{ij}^{(t)} (\vec{x_j} - \vec{m_i})^{(t-1)}}{\sum_{s=1}^{n} \alpha Fuzzy_{ij}^{(t)}} \tag{21}$$

Analisando a Equação (21), referente à atualização dos protótipos da *Fuzzy* LVQ não supervisionada, e a Equação (13), referente à atualização dos protótipos do FCM, nota-se que o diferencial entre os dois algoritmos está pautado em duas questões:

- na consideração ou não da posição atual do protótipo no cálculo da nova posição. Na realidade, a *Fuzzy* LVQ não supervisionada, seguindo os princípios da LVQ, executa um deslocamento no vetor protótipo, enquanto que o FCM realiza uma substituição do vetor protótipo.
- na possibilidade de atualizar o parâmetro de fuzificação, conferindo à *Fuzzy* LVQ não supervisionada um caráter mais autônomo em relação a este parâmetro.

O processo de execução da $\downarrow$ *Fuzzy LVQ* pode ser definido conforme descrito no Algoritmo 5 [3].

Considerando as discussões sobre o parâmetro de fuzificação m e sobre as taxas de aprendizado α e $\alpha Fuzzy$ já realizadas nesse texto, infere-se que:

- as redes pertencentes à família das $\uparrow$ *Fuzzy LVQ* possuem melhores condições para encontrar agrupamentos bem definidos, principalmente quando o valor inicial para o parâmetro de fuzificação é baixo ($\lim_{m_0 \to 1}$), pois nas iterações iniciais, tanto a taxa de aprendizado da rede neural quanto o parâmetro de fuzificação permitem a ordenação

Algoritmo 5 Algoritmo de treinamento da *Fuzzy* LVQ não supervisionada

Determine a quantidade de partições *fuzzy* c;
Inicialize o contador de iterações t;
Determine o erro máximo permitido ϵ;
Determine os valores dos parâmetros de fuzificação inicial $m^{(0)}$ e final $m^{(f)}$;
Inicialize a matriz de protótipos $\mathcal{M}$ aleatoriamente;
Inicialize o contador de iterações t como $t = 0$;
Inicialize a medida de término $E^{(0)}$ com o maior valor possível;
repita
  Atualize o valor de m, conforme Equação (19);
  para Cada dado j ($j = 1, ..., n$) **faça**
    Calcule o novo valor do coeficiente de aprendizado, o qual será específico para cada par (dado, grupo), conforme Equação (20)
  fim para
  Atualize M conforme Equação (21);
  Incremente t;
  Atualize $E^{(t)} =\| M^{(t)} - M^{(t-1)} \|$;
até que $t > T$ ou $E^{(t-1)} \leq \epsilon$

inicial dos protótipos, através de deslocamentos grandes, na direção dos grupos de dados. Com a evolução do algoritmo, tanto a taxa de aprendizado quando o parâmetro de fuzificação levam a rede a uma situação de estabilização;

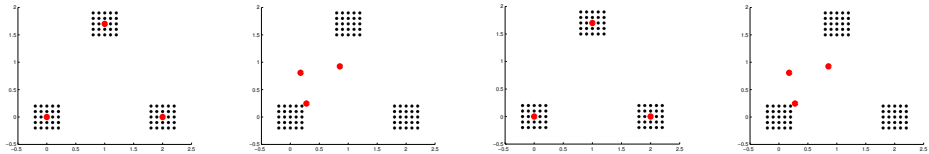
- na família das $\downarrow$ *Fuzzy LVQ*, onde $m^{(0)} > m^{(f)}$, essa fase inicial de ordenação (deslocamentos grandes) está combinada a valores altos do parâmetro de fuzificação, levando os protótipo para média dos dados (como discutido na Equação (16));

- no caso da família das LVQ constante, o comportamento se assemelha àquele discutido no algoritmo FCM, com baixos valores do parâmetro de fuzificação levando a um estado de melhor definição de grupos, e com altos valores desse parâmetros, levando os protótipos na direção do centro dos dados.

A Figura 8 ilustra cada um dos casos comentados.

5 Experimentos

A fim de ilustrar os processos de execução dos algoritmos aqui discutidos, nesta seção são apresentados alguns testes realizados sobre conjuntos de dados de cunho didático. São realizadas algumas comparações entre tais processos com o intuito de ilustrar como a combinação das técnicas levou à concepção de modelos robustos e eficientes. Para fins de



(a) (b) (c) (d)

Figura 8. Execuções da *Fuzzy* LVQ não supervisionada para o mesmo conjunto de dados e com os mesmos pesos iniciais. Condição de parada: $\epsilon = 0,0001$. (a) LVQ constante - $m^{(0)} = m^{(f)} = 1,1$, $E_q = 0,1874$ e 5 épocas; (b) LVQ constante - $m^{(0)} = m^{(f)} = 7$, $E_q = 0,8925$ e 28 épocas; (c) ↑ *Fuzzy LVQ* - $m^{(0)} = 1,1$ e $m^{(f)} = 7$, $E_q = 0,1874$ e 5 épocas; (d) ↓ *Fuzzy LVQ* - $m^{(0)} = 7$, $m^{(f)} = 1,1$, $E_q = 0,8925$ e 28 épocas.

comparação, em cada conjunto de dados fixou-se os valores dos pesos iniciais para todos os algoritmos e o critério de parada para todas as execuções foi $\epsilon = 0,0001$[17]. O valor 2 foi escolhido para o parâmetro de fuzificação m, sendo que para os casos em que existe variação no m o intervalo estabelecido foi $[1,1;2]$.

O primeiro exemplo, ilustrado na Figura 9 e Tabela 7, é a aplicação dos algoritmos estudados nesse tutorial em um conjunto de dados sugerido por Yager e Filev [47]. Este conjunto de dados possui três grupos bem definidos e bem separados, com formato retangular e todos com a mesma densidade de dados[18].

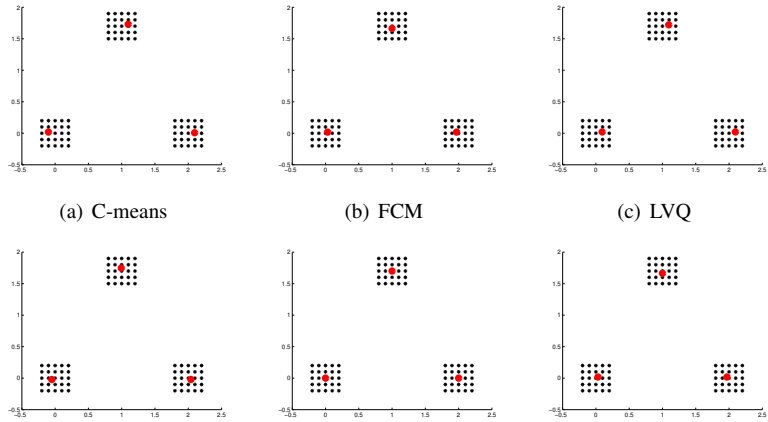
	C-Means	FCM	LVQ	*Fuzzy* LVQ Supervisionada	↑ *Fuzzy* LVQ	↓ *Fuzzy* LVQ
Erro de Quantização	0,2058	0,1906	0,2046	0,1929	**0,1874**	0,1907
Número de épocas	50	11	4	41	**5**	36

Tabela 7. Informações sobre a execução para o conjunto de dados de fácil resolução.

Todos os modelos desse primeiro exemplo chegaram a resultados de classificação ou agrupamento satisfatórios. Contudo, percebe-se claramente a superioridade dos modelos híbridos, com destaque para a ↑ *Fuzzy* LVQ, que além de chegar ao erro de quantização mais baixo, também se apresenta como uma abordagem bastante eficiente em termos de número

[17]Uma exceção foi estabelecida para a execução do *c-Means*, cujo critério de parada foi $\epsilon = 0,01$ para os dois primeiros conjuntos e $\epsilon = 0,001$ para o terceiro conjunto. Essa estratégia foi adotada para impedir que o *c-Means* executasse por muito mais tempo do que as demais técnicas, visto que este algoritmo tem um tempo de convergência mais longo. Execuções mais longas do que as apresentadas aqui não levam a melhores resultados.

[18]Características que permitem verificar a corretude do algoritmo, dado que se trata de um conjunto de dados com grupos bem definidos e "fáceis" de encontrar.



(a) C-means (b) FCM (c) LVQ

(d) *Fuzzy* LVQ supervisionada (e) ↑ *Fuzzy LVQ* (f) ↓ *Fuzzy LVQ*

Figura 9. Gráficos dos resultados de aplicação dos algoritmos estudados neste tutorial: conjunto de dados de fácil resolução. Os dados estão representados pelas "cruzes" e os protótipos de grupos estão representados pelos "círculos".

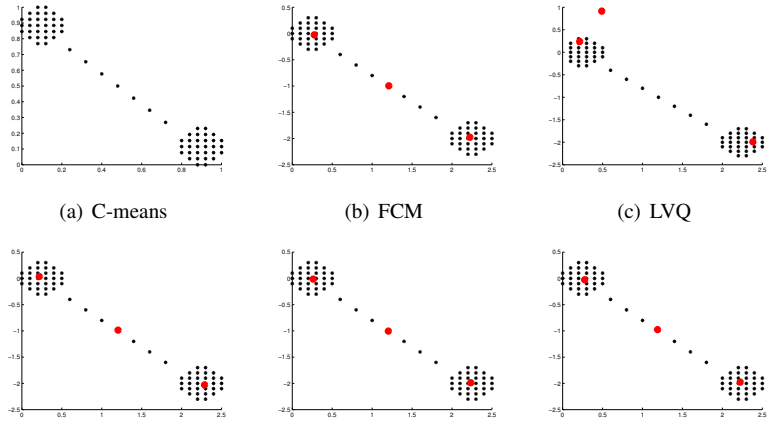
de épocas executadas.

A Figura 10 e a Tabela 8 trazem informações referentes à execução dos algoritmos em um segundo conjunto sugerido por Yager e Filev [47]. Esse conjunto proporciona interpretações subjetivas quanto ao número de grupos existentes, podendo ser interpretado como um conjunto com dois ou três grupos, a depender de onde os sete pontos localizados entre os dois grupos das extremidades serão agrupados. Tais pontos podem ser considerados um grupo, pontos pertencentes aos outros dois grupos, ou podem ser considerados como ruído.

	C-Means	FCM	LVQ	*Fuzzy* LVQ Supervisionada	↑ *Fuzzy* LVQ	↓ *Fuzzy* LVQ
Erro de Quantização	0,2976	0,2278	0,3494	0,2317	**0,2269**	0,2281
Número de épocas	não convergiu	36	5	85	**11**	66

Tabela 8. Informações sobre a execução para o conjunto de dados com dupla interpretação de número de grupos.

Os resultados do segundo exemplo confirmam a superioridade dos modelos híbridos. Nesse caso, os modelos híbridos consumiram mais épocas de execução, porém se mostra-



(a) C-means (b) FCM (c) LVQ

(d) *Fuzzy* LVQ supervisionada (e) ↑ *Fuzzy LVQ* (f) ↓ *Fuzzy LVQ*

Figura 10. Gráficos dos resultados de aplicação dos algoritmos estudados neste tutorial: conjunto de dados com dupla interpretação de número de grupos. Os dados estão representados pelas "cruzes" e os protótipos de grupos estão representados pelos "círculos".

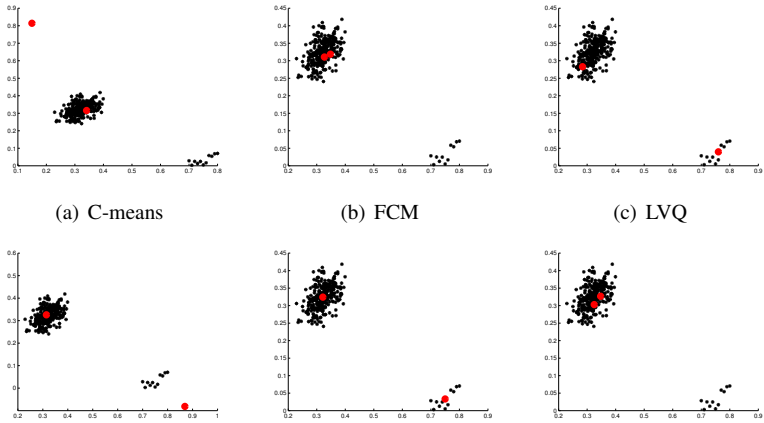
ram factíveis para resolução do problema considerando a existência de três grupos, enquanto o *c-Means* e a LVQ não são capazes de "entender" o conjunto de dados sob esse perspectiva. Considerando a existência de apenas dois grupos, os resultados dos modelos clássicos melhoram, pois sua principal dificuldade está em "encontrar o terceiro grupo".

Finalmente, o último exemplo está ilustrado na Figura 11 e na Tabela 9. Trata-se de um conjunto de dados com variação de densidade de dados entre os grupos, adaptado de [14]. O conjunto possui 312 pontos, divididos em 301 pontos para o primeiro grupo e 11 pontos para o segundo.

	C-Means	FCM	LVQ	*Fuzzy* LVQ Supervisionada	↑ *Fuzzy* LVQ	↓ *Fuzzy* LVQ
Erro de Quantização	0,0616	0,0541	0,0637	0.0455	**0,0403**	0,0516
Número de épocas	19	7	6	55	**11**	16

Tabela 9. Informações sobre a execução para o conjunto de dados com densidade de dados diferentes nos grupos.

No caso desse último exemplo, há que se destacar o ótimo desempenho do modelo ↑ *Fuzzy* LVQ. O conjunto de dados usado não favorece técnicas não supervisionadas (observe



(a) C-means (b) FCM (c) LVQ

(d) *Fuzzy* LVQ supervisionada (e) ↑ *Fuzzy LVQ* (f) ↓ *Fuzzy LVQ*

Figura 11. Gráficos dos resultados de aplicação dos algoritmos estudados neste tutorial: conjunto de dados com densidade de dados diferentes nos grupos. Os dados estão representados pelas "cruzes" e os protótipos de grupos estão representados pelos "círculos".

que as técnicas supervisionadas resolvem o problema), visto a grande disparidade da densidade de dados em cada grupo. Entretanto, mesmo em condições desfavoráveis o modelo ↑ *Fuzzy* LVQ apresentou o melhor erro de quantização[19]. Em termos de classes/grupos obtidos, a LVQ apresenta resultados superiores, ainda que seu erro de quantização seja mais alto. Esse comportamento se deve ao fato da grande quantidade de dados no grupo onde os dois protótipos do algorimto FCM se posicionaram, usando um pequeno número de épocas.

6 Aplicações - Visão Geral

Os modelos explorados neste trabalho (FCM e *Fuzzy* LVQ) foram apresentados no início da década de 90. Atualmente, eles tem sido aplicados na execução de tarefas de agrupamento e também utilizados como base para o desenvolvimento de novos modelos com o objetivo de minimizar deficiências observadas durante suas aplicações, como a sensibilidade a ruídos, o alto custo computacional exigido e a adequabilidade para aplicação em problemas que apresentam necessidades particulares.

[19]O erro de quantização é usado como uma medida de qualidade para os resultados de tarefas de classificação e agrupamento, como discutido na Seção 2.6, contudo, a depender da situação, esse erro não indica o melhor resultado obtido. Um exemplo é o caso do algoritmo FCM e do modelo LVQ.

A modelagem original do modelo FCM tem sido aplicada, por exemplo, para o processamento de imagens médicas [37, 35], para a análise e compressão de imagens [40] e outros problemas de reconhecimento de padrões [38]. Em [37], o FCM é utilizado na segmentação de imagens de ressonância magnética para detecção de exsudatos [20] - o primeiro sinal de Retinopatia Diabética - com precisão de $99,11\%$. Em [35], variações do FCM são utilizadas para segmentar lesões em imagens de dermatoscopia. Nesse caso, a versão clássica do algoritmo apresenta $0,74$ de sensibilidade e $0,99$ de especificidade, enquanto a versão proposta apresenta $0,77$ de sensibilidade e $0,99$ de especificidade, além de ter menor custo computacional. O FCM também é utilizado para identificação online de um usuário por meio de sua escrita a mão com $98,3\%$ de precisão em [38]. Em [40], uma versão modificada *in batch* do algoritmo *Fuzzy* LVQ, baseada na função objetivo do FCM, foi utilizado para compressão de imagens, obtendo resultados superiores a própria *Fuzzy* LVQ clássica e a outros algoritmos, como versões modificadas da rede neural *Linde Buzo Gray* (LBG) e do algoritmo clássico *c-Means*.

Em [42] os autores avaliam quatro diferentes métodos de agrupamento *fuzzy*, desenvolvidos com base no FCM padrão, a fim de identificar pontos em comum e limitações específicas de cada método. Ao final, é proposto um algoritmo de agrupamento otimizado que foi aplicado com sucesso na classificação de posturas humanas.

O algoritmo FCM também foi recentemente usado na construção de uma prova de conceito para uma nova abordagem de inserção de conhecimento de domínio na resolução da tarefa de agrupamento. Em [32], os autores introduzem o conceito chamado *"viewpoints"*, o qual é uma forma pela qual o usuário pode introduzir algum conhecimento sobre os dados que estão sendo agrupados por um processo de otimização qualquer. Apesar dos autores terem escolhido o FCM em sua prova de conceito, a abordagem introduzida neste trabalho não está restrita a essa classe de algoritmos de agrupamento.

O algoritmo *Fuzzy* LVQ tem sido utilizado em Geociências [11], processamento de dados sensoriais (*processing sensor-array data*) [13] e reconhecimento de padrões [7, 24]. Os autores em [11] aplicaram diferentes arquiteturas de redes neurais (*Multilayer Perceptron* (MLP) *Self Organizing Maps* (SOM) e *Fuzzy* LVQ) no problema de classificação no contexto de mapeamento geológico. Diferentes experimentos foram executados e foi observado a superioridade da *Fuzzy* LVQ tanto em relação a acuidade dos resultados alcançados quanto em relação a velocidade de execução e estabilidade do modelo. A *Fuzzy* LVQ também foi alvo de comparação com outras arquiteturas de redes neurais em aplicações de reconhecimento de padrões de odores. Em [13] os autores afirmam que a *Fuzzy* LVQ apresenta alta capacidade de reconhecimento neste tipo de aplicação. Nesse trabalho, o desempenho dessa arquitetura neural foi melhorado pela combinação de seu processo de aprendizado com o

[20] matéria resultante de processo inflamatório : fonte - FERREIRA, Aurélio Buarque de Holanda. Dicionário da língua portuguesa. Rio de Janeiro (RJ), Brasil: 2000, verbete exsudato

método de otimização *Particle Swarm*. Em [7], as arquiteturas neurais SOM, LVQ e *Fuzzy* LVQ supervisionada e não supervisionada foram testadas para resolver um problema de reconhecimento de padrões de movimentos gestuais. Nesse trabalho, a rede neural *Fuzzy* LVQ não supervisionada obteve os melhores resultados dentre os modelos testados, apresentando taxas de reconhecimento altas com um número reduzido de neurônios em relação aos demais modelos. Em [24], redes neurais *Fuzzy* LVQ não supervisionadas e supervisionadas foram utilizadas para compor um comitê de máquinas visando reconhecer imagens estáticas e movimentos dinâmicos representando as configurações de mão e movimentos da Língua Brasileira de Sinais. O comitê foi capaz de reconhecer as 20 configurações de mão e os cinco movimentos utilizados para representar as letras do alfabeto na Língua Brasileira de Sinais com precisão de, respectivamente, 85% e $91,7\%$.

Versões híbridas, que combinam o FCM com outros algoritmos, são exploradas em diferentes contextos de aplicação. Com o objetivo de analisar a eficiência de companhias de distribuição de energia, os autores do trabalho apresentado em [36] aplicaram o FCM em conjunto com os métodos *Principal Component Analysis* (PCA) e *Data Envelopment Analysis* (DEA). Nesse trabalho, o FCM foi a técnica escolhida para agrupar companhias de distribuição de energia com base em variáveis ambientais, de forma que cada companhia seja comparada apenas com aquelas atuantes em contextos similares, evitando comparações de organizações com realidades muito diferentes. Também no contexto de sistemas de energia, os autores do trabalho apresentado em [26] constróem um sistema híbrido usando FCM, *Quantum-Inspired Particle Swarm Optimization* (QPSO) e redes neurais *Radial Basis Function* (RBF). Nesse sistema híbrido, o algoritmo FCM atua na busca dos centros das funções kernel da rede neural RBF e o cálculo dos pesos das conexões da camada de saída da rede neural é suportado por uma técnica de otimização global, a QPSO, também introduzida nesse artigo. Em [8] é proposto um algoritmo baseado no FCM e no princípio do Filtro *Sigma*, o *Improved Fuzzy C-means* (IFCM), visando a segmentação de imagens de ressonância magnética. O uso do novo algoritmo diminuiu o erro médio na segmentação de $2,5\%$, utilizando o FCM tradicional, para $0,74\%$.

Em termos de evolução, é perceptível que a principal preocupação em relação aos modelos FCM e *Fuzzy* LVQ é a sensibilidade a ruídos. Visando fornecer soluções para esse problema, diversos trabalhos foram desenvolvidos. No caso do FCM, o modelo *Fuzzy Possibilistic C-Means* (FPCM) é proposto em [30] visando superar o problema de sensibilidade a ruídos, presente no FCM, e um problema presente no modelo *Possibilistic C-Means* (PCM), que apresenta uma tendência a convergir para clusters coincidentes, através da consideração de graus de pertinência e valores de possibilidade. Posteriormente, uma evolução desse algoritmo é apresentada em [31]: o modelo *Possibilistic Fuzzy C-Means* (PFCM) é um modelo híbrido baseado nos modelos FCM e PCM que pretende evitar os problemas já citados, além de evitar certa dificuldade do FPCM em lidar com grandes conjuntos de dados. O algoritmo FPCM também embasou a criação de uma abordagem para implementação de *co-clustering*,

apresentada em [39].

Alguns algoritmos, como o algoritmo *Fuzzy Local Information C-means* (FLICM) apresentado em [18], visam superar as desvantagens do FCM objetivando uma área de aplicação específica - neste caso, a segmentação de imagens. O FLICM utiliza informações locais de níveis de cinza e espaciais para segmentar imagens. O algoritmo apresenta robustez a ruídos e pode ser utilizado diretamente nas imagens originais, ao invés de imagens pré-processadas. O problema do ruído também foi notado em [6], em uma aplicação de segmentação de imagens. Como no modelo FCM não são consideradas as informações de vizinhança espacial, que são fundamentais para o alcance de bons resultados na segmentação de imagens, em [6, 18], os autores propõem outras modificações na medida de similaridade do FCM padrão, com vistas a melhorar seu desempenho neste tipo de aplicação.

Já no caso da *Fuzzy* LVQ, algums evoluções são representadas pelos trabalhos descritos em [44, 45]. O modelo *Possibilistic Fuzzy Learning Vector Quantization* (PFLVQ) proposto em [44] integra o modelo de agrupamento PFCM no modelo *Fuzzy* LVQ para lidar com os ruídos presentes nos dados. Outro modelo proposto recentemente em [45] consiste em integrar os princípios da LVQ ao modelo *Generalized Noise Clustering* (GNC), ao invés do modelo FCM - que é sensível a ruídos - para criar o *Noise Fuzzy Learning Vector Quantization* (NFLVQ), que utiliza os valores de pertinência do algoritmo GNC como taxas de aprendizado e apresenta melhores resultados no agrupamento de conjuntos de dados com ruídos.

7 Considerações Finais

Este tutorial cobriu os principais conceitos referentes ao algoritmo *Fuzzy-c-Means* e aos modelos neurais-*fuzzy Fuzzy Learning Vector Quantization*. Além disso, conceitos básicos normalmente necessários para o entendimento de modelos que se aplicam a tarefas de agrupamento e classificação foram apresentados, juntamente com as suas versões clássicas (*C-Means* e *Learning Vector Quantization*).

A teoria discutida, bem como os exemplos didáticos apresentados, tiveram o objetivo de colocar o leitor em condições de melhor entender o comportamento dos algoritmos e, com isso, otimizar o uso dos mesmos. A explanação teórica, sobretudo sobre o algoritmo *Fuzzy-c-Means*, ilustrou a importância da correta parametrização dos modelos, visto que diferentes valores para diferentes parâmetros, comprovadamente, levam a comportamentos e resultados diferenciados. O algoritmo *Fuzzy-c-Means*, que serviu de base para a elaboração do modelo *Fuzzy* LVQ, apresentado neste trabalho, tem sido utilizado também como base para o desenvolvimento de diversas novas modelagens. Conhecer em detalhes o comportamento do algoritmo FCM diante dos parâmetros de entrada torna-se fundamental para o pleno entendimento das abordagens evoluídas do algoritmo padrão.

A adequabilidade do tratamento da incerteza e imprecisão, suportados pela Teoria *Fuzzy*, foi evidenciada pelos exemplos presentes na seção de experimentos. Fica evidente também nos exemplos didáticos a superioridade dos métodos híbridos sobre suas versões clássicas. Não obstante, o referencial bibliográfico que suporta a discussão aqui desenvolvida, mostra que os métodos híbridos aqui tratados estão em constante aplicação e evolução.

8 Referências

[1] ABE, S. *Support Vector Machines for Pattern Classification (Advances in Pattern Recognition)*, 1 ed. Springer, July 2005.

[2] BEZDEK, J. *Pattern Recognition with Fuzzy Objective Function Algorithms*, 1 ed. Kluwer Academic Publishers, 1981.

[3] BEZDEK, J. C. E PAL, S. K. *Fuzzy models for pattern recognition : methods that search for structures in data. Methods that search for structures in data*. IEEE Press, New York, 1992.

[4] BEZDEK, J. C., TSAO, E. C. E PAL, N. R. Fuzzy kohonen clustering networks. In *IEEE International Conference on Fuzzy Systems* (1992), pp. 1035–1043.

[5] CHUNG, F. E LEE, T. Fuzzy learning vector quantization. In *IEEE International Joint Conference on Neural Networks* (1993), pp. 2739 –2742.

[6] DESPOTOVIC, I., GOOSSENS, B., VANSTEENKISTE, E. E PHILIPS, W. An improved fuzzy clustering approach for image segmentation. In *17th IEEE International Conference on Image Processing* (September 2010), pp. 249 –252.

[7] DIAS, D. B., MADEO, R. C. B., ROCHA, T., BÍSCARO, H. H. E PERES, S. M. Hand movement recognition for Brazilian Sign Language: A study using distance-based neural networks. In *International Joint Conference on Neural Networks* (2009), IEEE, pp. 697–704.

[8] DU, R. E LEE, H. J. A modified-FCM segmentation algorithm for brain MR images. In *International Conference on Hybrid Information Technology (ICHIT '09)* (New York, NY, USA, 2009), ACM, pp. 25–27.

[9] EVERITT, B. S. E RABE-HESKETH, S. *The Analysis of Proximity Data*. Hodder Arnold Publishers, Londres, 1997.

[10] FAUSETT, L. *Fundamentals of Neural Netoworks: architectures, algorithms, and applications*. Prentice-Hall, Inc., New Jersey, USA, 1994.

[11] FILIPPI, A. E JENSEN, J. Effect of continuum removal on hyperspectral coastal vegetation classification using a fuzzy learning vector quantizer. *IEEE Transactions on Geoscience and Remote Sensing, 45*, 6 (June 2007), 1857–1869.

[12] HAYKIN, S. *Neural Networks: a comprehensive foundation*, 2nd ed. Prentice Hall Inc, New Jersey, USA, 1998.

[13] JATMIKO, W., ROCHMATULLAH, KUSUMOPUTRO, B., SANABILA, H., SEKIYAMA, K. E FUKUDA, T. Visualization and statistical analysis of fuzzy-neuro learning vector quantization based on particle swarm optimization for recognizing mixture odors. In *International Symposium on Micro-NanoMechatronics and Human Science, 2009. MHS 2009* (9-11 2009), pp. 420–425.

[14] KAYMAK, U. E SETNES, M. Extended fuzzy clustering algorithms. Research Paper ERS-2000-51-LIS, Erasmus Research Institute of Management (ERIM), November 2000.

[15] KLIR, G. J. E YUAN, B. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, 1995.

[16] KOHONEN, T. *Self Organizing Maps*, 3rd ed. Springer, 2000.

[17] KOSKO, B. *Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence*. Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[18] KRINIDIS, S. E CHATZIS, V. A robust fuzzy local information c-means clustering algorithm. *IEEE Transactions on Image Processing 19*, 5 (May 2010), 1328 –1337.

[19] KRUSE, R., DÖRIGN, C. E LESOT, M.-J. *Fundamentals of Fuzzy Clustering*. John Wiley & Sons Ltd, 2007, ch. Advances in Fuzzy Clustering and its Applications, pp. 3–30.

[20] LEE, R. S. T. *Fuzzy-Neuro Approach to Agente Applications*, 1 ed. Springer, December 2005.

[21] LEE, S. E LEE, E. Fuzzy neural networks. *Mathematical Biosciences 23* (1975), 151–177.

[22] LIU, X. E PEDRYCZ, W. *Axiomatic Fuzzy Set Theory and Its Applications*, 1 ed. Springer, April 2009.

[23] MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability* (1967), Berkeley, University of California Press, pp. 1:281–297.

[24] MADEO, R. C. B., PERES, S. M., BÍSCARO, H. H., DIAS, D. B. E BOSCARIOLI, C. A committee machine implementing the pattern recognition module for fingerspelling applications. In *25th Annual ACM Sumposium on Applied Computing* (2010), ACM.

[25] MCCULLOCH, W. S. E PITTS, W. A logical calculus of the ideas immanent in nervous activity. 15–27.

[26] MENG, K., DONG, Z. Y., WANG, D. H. E WONG, K. P. A self-adaptive rbf neural network classifier for transformer fault analysis. *IEEE Transactions on Power Systems PP*, 99 (2010), 1 –1. IEEE Early Access.

[27] MITRA, S. E HAYASHI, Y. Neuro-fuzzy rule generation: Survey in soft computing framework. *IEEE Transactions on Neural Network 11*, 3 (May 2000), 748–768.

[28] MIYAMOTO, S., ICHIHASHI, H. E HONDA, K. *Algorithms for Fuzzy Clustering: Methods in c-Means Clustering with Applications*, 1 ed. Springer, April 2008.

[29] OLIVEIRA, J. V. E PEDRYCS, W., Eds. *Advances in Fuzzy Clustering and its Applications*. John Wiley & Sons, Ltd, 2007.

[30] PAL, N. R., PAL, K. E BEZDEK, J. C. A mixed c-means clustering model. In *Proceedigns of IEEE International Conference on Fuzzy Systems* (Barcelona, Spain, July 1997), pp. 11–21.

[31] PAL, N. R., PAL, K., KELLER, J. M. E BEZDEK, J. C. A possibilistic fuzzy c-means clustering algorithm. *IEEE T. Fuzzy Systems 13*, 4 (2005), 517–530.

[32] PEDRYCS, W. E GOMIDE, F. *An Introduction to Fuzzy Sets: Analysis and Design*. A Bradford Books, 1998.

[33] PEDRYCZ, W., LOIA, V. E SENATORE. Fuzzy clustering with viewpoints. *IEEE Transactions on Fuzzy Systems 18*, 2 (April 2010), 274 –284.

[34] RUSSEL, S. E NORVIG, P. *Artificial Inteligence: A Modern Approach*, 2nd ed. Pearson, December 2002.

[35] SCHAEFER, H. Z., SADKA, G. E CELEBI, M. E. Anisotropic mean shift based fuzzy c-means segmentation of dermoscopy images. *IEEE Journal of Selected Topics in Signal Processing 3*, 1 (2009), 26–34.

[36] SIMAB, M. E HAGHIFAM, M.-R. Using integrated model to assess the efficiency of electric distribution companies. *IEEE Transactions on Power Systems PP*, 99 (2010), 1–1. IEEE Early Access.

[37] SOPHARAK, A., UYYANONVARA, B. E S., B. Automatic exudate detection from non-dilated diabetic retinopathy retinal images using fuzzy c-means clustering. *Sensors 9*, 3 (2009), 2148–2161.

[38] TAN, G. X., VIARD-GAUDIN, C. E KOT, A. Online writer identification using fuzzy c-means clustering of character prototypes. In *Proceedings of 11th International Conference on Frontiers in Handwriting Recognition* (Montreal, Canadá, 2008), pp. 475–480.

[39] TJHI, W.-C. E CHEN, L. Dual fuzzy-possibilistic coclustering for categorization of documents. *IEEE Transactions on Fuzzy Systems 17*, 3 (2009), 532–543.

[40] TSEKOURAS, G. E., ANTONIOS, M., ANAGNOSTOPOULOS, C., GAVALAS, D. E ECONOMOU, D. Improved batch fuzzy learning vector quantization for image compression. *International Journal of Information Sciences 178* (October 2008).

[41] VUORIMAA, P. Fuzzy self-organizing map. *Fuzzy Sets and Systems 66* (1994), 223–231.

[42] WANG, Z. Comparison of four kinds of fuzzy c-means clustering methods. In *Third International Symposium on Information Processing* (Octorber 2010), pp. 563–566.

[43] WERMTER, S. E SUN, R. An overview of hybrid neural systems. In *Hybrid Neural Systems* (2000), vol. 1778 of *Lecture Notes in Computer Science*, Springer, pp. 1–13.

[44] WU, X.-H., FU, H., WU, B. E ZHAO, J.-W. Possibilistic fuzzy learning vector quantization. *Journal of Information and Computational Science 7*, 3 (2010), 777–783.

[45] WU, X.-H., WU, B. E ZHAO, J.-W. Noise fuzzy learning vector quantization. *Key Engineering Materials 439-440* (2010), 367–371.

[46] XU, R. E WUNSCH, D. II survey of clustering algorithms. *IEEE Transactions on Neural Networks 16*, 3 (may 2005), 645–678.

[47] YAGER, R.R., F. D. Approximate clustering via the mountain method. In *IEEE Transactions on Systems, Man and Cybernetics* (1994), pp. 1279–1284.

[48] ZADEH, L. A. Fuzzy sets. *Information and Control 8*, 3 (1965), 338–353.