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.