LUCAS BARBOSA GALETE

UM MÉTODO PARA AGRUPAMENTO EM FLUXO DE DADOS UTILIZANDO O ALGORITMO SOM

Dissertação de mestrado submetida ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção do título de Mestre em Informática.

Curitiba PR
Fevereiro de 2012

LUCAS BARBOSA GALETE

UM MÉTODO PARA AGRUPAMENTO EM FLUXO DE DADOS UTILIZANDO O ALGORITMO SOM

Dissertação de mestrado submetida ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção do título de Mestre em Informática.

Área de concentração: *Ciência da Computação*
Linha de Pesquisa: *Descoberta do Conhecimento e Aprendizagem de Máquina*

Orientador: Prof. Emerson Cabrera Paraiso
Co-orientador: Prof. Júlio César Nievola

Curitiba PR
Fevereiro de 2012

Último sobrenome (seguido de Jr., Filho ou Neto), Primeiro nome e demais sobrenomes.

Título da Dissertação ou Tese. Curitiba, 2001. 88p.

Dissertação (Mestrado) ou Tese (Doutorado) - Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. Palavra-chave 2. Palavra-chave 3. Palavra-chave 4. Palavra-chave. I. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática II-t.

Esta folha deve ser substituída pela ata de defesa devidamente assinada, que será fornecida pela secretaria do programa após a defesa.

Dedico este trabalho à família e amigos que me apoiaram nas mais diversas ocasiões.

Resumo

Com o crescimento das bases de dados e a necessidade de extrair informação destas, muitos algoritmos foram propostos. Em aplicações onde há a coleta de informações em tempo real ou com grande frequência, pode ocorrer a geração de um fluxo de dados. Como exemplos, pode-se citar: redes de sensores, tráfego de rede de computadores ou os dados gerados em uma aplicação interativa. Neste trabalho propomos um método para agrupamento em fluxo de dados utilizando o algoritmo SOM. Entretanto, o algoritmo SOM não extrai os grupos da base fornecida automaticamente, somente organiza-os em um mapa. Para realizar a extração automática dos grupos, desenvolvemos o algoritmo GSOM. O algoritmo GSOM extrai os grupos através do processo de refinamento iterativo da rede, utilizando a medida de coeficiente de Silhueta para a validação de grupos. Os resultados demonstram que o algoritmo GSOM é eficaz para extrair grupos de uma rede treinada SOM, conforme podemos observar através do número de grupos e qualidade obtidos através das 9 bases de dados testadas.

Palavras-chave: agrupamento, som kohonen, redes neurais.

Abstract

Researchers have proposed many algorithms to mine information on databases. In applications where information must be mined in real time or in applications where data is generated in a high frequency, a data stream may occur. Examples include sensor networks, computer network traffic, or generated data from an interactive application. In this work we propose a method for clustering in data streams using the SOM algorithm. The SOM algorithm does not extract the groups from the given training base automatically. In order t o make the automatic group extraction, we developed the GSOM algorithm. The GSOM algorithm extracts the grouping from the map by using a process called iterative grid refinement, using the silhouette validation measure. The results show that GSOM is effective to extract groups of a SOM's trained map, as we can observe by the cluster number and quality obtained through 9 bases we tested.

Keywords: clustering, som, neural network.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Com o crescimento das bases de dados e a necessidade de extrair informação e conhecimento destas, muitos algoritmos foram propostos [Ester et al., 1996, Guha et al., 1999, Dutta et al., 2005, Guha et al., 1998, Arai, 2007, Karypis and Kumar, 1999, Wu et al., 2008]. Entretanto, tais algoritmos trabalham com bases de dados completas e sempre disponíveis. Em alguns casos, a geração de dados é muito rápida. Estes dados, muitas vezes, não podem ser armazenados devido a limitação de recursos, *i.e.*, capacidade de armazenamento e comunicação. Esta grande quantidade de dados é tratada como um fluxo de dados.

Em aplicações onde há a coleta de informações em tempo real ou com grande frequência, gera-se um fluxo de dados. Como exemplos, pode-se citar redes de sensores, tráfego de rede de computadores, leitura de dados em uma frequência de rádio, dentre outras.

Técnicas de mineração de dados para lidar com estes fluxos foram propostos nos últimos anos [Gaber et al., 2005, Guha et al., 2003, Guha et al., 2000, Beringer and Hullermeier, 2006, Babcock et al., 2002, Golab and Ozsu, 2003]. O agrupamento em fluxo de dados está relacionado à extração de estruturas de conhecimento representados em modelos e padrões através de um fluxo contínuo de dados [Gaber et al., 2005].

Em alguns casos, onde há a necessidade de realizar agrupamento sobre uma grande quantidade de dados, temos o problema de tratamento de fluxo de dados. Em geral, o agrupamento necessita ser realizado em tempo real e levando-se em consideração a evolução do fluxo. Isso quer dizer que, em um determinado momento o agrupamento apresenta determinada característica, e em um momento seguinte o agrupamento pode apresentar características semelhantes às anteriores.

Uma aplicação interativa no âmbito da TV digital pode gerar um fluxo de dados. Para entender, tomemos como exemplo uma aplicação interativa sendo executada por vários usuários (telespectadores) simultaneamente. Durante a execução da aplicação e a consequente interação, os usuários vão gerando dados que, enviados a um servidor, devem ser processados. O grande volume de dados, e sua constante geração, formam um fluxo. Além de representar as ações realizadas pelos usuários durante a interação com a aplicação, tal conjunto de dados pode ser utilizado também para entender melhor o comportamento dos usuários. Pode-se, por exemplo, em função dos dados gerados, agrupar os usuários em categorias ou grupos. Estes grupos de usuários representam usuários que tenham a semelhanças relativas ao seu comportamento.

Neste trabalho apresentamos um algoritmo desenvolvido para, em conjunto com o algoritmo de agrupamento SOM, escolher de forma automática o número de grupos de um agrupamento. Existem métodos recentes para extração automática de grupos

[Ghaseminezhad and Karami, 2011] [Cabanes and Bennani, 2010], porém com abordagem diferente da utilizada neste trabalho. Os experimentos indicam que o algoritmo é capaz de escolher o melhor agrupamento de acordo com uma dada medida de similaridade, neste caso, utilizando a medida de silhueta.

1.1 Visão geral do trabalho

Este trabalho apresenta um método de agrupamento em fluxo de dados através de uma rede neural auto-organizável (SOM), tratando o fluxo de dados através de uma fila, utilizando técnicas de amostragem estatística. O fluxo de dados é serializado nesta fila, de onde será lido e agrupado.

Para se obter este fluxo de dados e realizar os experimentos, implementamos um simulador capaz de gerá-lo, com algumas especificações variáveis como a base a ser utilizada para fluxo, taxa de produção de dados por segundo, configurações da rede SOM, e opções de tratamento da fila para manter a estabilidade do sistema.

A leitura de pacotes de dados é realizada através de janelas de leitura. Esta, lê um bloco de informação, e este bloco é utilizado pelo algoritmo de agrupamento. Após o agrupamento pelo algoritmo SOM é preciso extrair os grupos (*labels*) do mapa resultante. Para isso foi desenvolvido um algoritmo de extração automática dos grupos.

Após realizado a extração automática dos grupos é obtido o modelo, o qual pode ser utilizado para associar novos registros àos grupos já obtidos ou simplesmente utilizar os grupos resultantes para análise dos dados. Em uma aplicação interativa, pode-se analisar o comportamento dos usuários, ou seja, extrair características relativas ao comportamento deles, para então agrupá-los.

1.2 Objetivos

O objetivo principal deste trabalho é projetar e validar um método de agrupamento em fluxo de dados utilizando o algoritmo de agrupamento SOM. O agrupamento em fluxo de dados necessita de uma forma de extração automática dos grupos a partir da rede treinada. Pretende-se realizar o agrupamento de dados mantendo-se a qualidade dos grupos através da validação não-supervisionada dos mesmos.

Definimos como objetivos específicos:

1. realizar o controle do fluxo de dados através de uma fila e amostragem estatística;
2. manter o nível da fila dentro dos limites estabelecidos;
3. realizar a extração automática de grupos da rede treinada SOM;
4. implementar um simulador a partir do método proposto.

1.3 Escopo da Pesquisa

Os experimentos realizados neste trabalho envolvem duas etapas: o algoritmo de extração automática de grupos, e o uso deste algoritmo durante o agrupamento em fluxo de dados.

Para testar o algoritmo de extração de grupos, utilizamos bases de dados artificiais e reais, analisando a qualidade e o número de grupos, comparando com os algoritmos DBSCAN e K-Means. O método de agrupamento em fluxo de dados foi testado utilizando-se um simulador desenvolvido. Diferentes parâmetros foram utilizados para se verificar os resultados.

1.4 Hipótese de Trabalho

A hipótese de trabalho visa verificar se é possível realizar o agrupamento em fluxo de dados utilizando o algoritmo SOM obtendo um agrupamento de forma não assistida.

1.5 Contribuições Científicas e Tecnológicas

Como contribuições científicas podemos citar o algoritmo de *extração automática de grupos* em redes SOM treinadas. Assim como a utilização do mesmo no *método para agrupamento em fluxo de dados*.

Como contribuição tecnológica apresentamos o simulador implementado a partir do *método para agrupamento em fluxo de dados*. Este permite realizar o agrupamento em fluxo de dados sob demanda utilizando a rede SOM, enquanto permite visualização em tempo real dos resultados obtidos.

1.6 Organização do Documento

Nos capítulos seguintes estão organizados da seguinte maneira: o capitulo 2 apresenta todo o embasamento teórico necessário para a compreensão do método proposto e algoritmos utilizados. No capítulo 3 apresentamos o método proposto para o agrupamento em fluxo de dados. Na sequência, o capítulo 4 relembra ao leitor o funcionamento do algoritmo SOM e apresenta o algoritmo GSOM, utilizado para a extração dos grupos no mapa treinado SOM. No capítulo 5 é apresentado o simulador e o software utilizado para rodar os testes, os quais são apresentados no capítulo 6. O documento é finalizado com o capítulo 7 onde é discutido os resultados obtidos e trabalhos futuros. Em anexo estão resultados obtidos com os algoritmos DBSCAN e KMEANS.

Capítulo 2

Fundamentos Teóricos

Neste capítulo será apresentado a fundamentação teórica necessária para o entendimento do método, descrito no capítulo 3. A fundamentação inclui definição de objetos e grupos, tipos de grupos, medidas de distância utilizadas nos algoritmos de agrupamento, agrupamento utilizando redes neurais artificiais e técnicas de validação do agrupamento. Este capítulo também apresenta técnicas utilizadas para agrupamento em fluxo de dados e trabalhos correlatos.

2.1 A Tarefa de Agrupamento

O estudo dos algoritmos de agrupamento é essencial para o entendimento do método de agrupamento em fluxo de dados, uma vez que será a principal tarefa utilizada neste projeto.

Agrupamento de dados é a organização de um conjunto de objetos em grupos baseado em similaridade. Em outras palavras, um objeto de um grupo é mais similar aos objetos do mesmo grupo do que os objetos de outros grupos. Estes objetos, geralmente, são representados em vetores de atributos (ou característica), ou um ponto em um espaço multi-dimensional. A tarefa de agrupamento de dados é organizar estes objetos em grupos de forma que os objetos pertencentes aos grupos sejam mais similares entre si, dado certas medidas de similaridade (coesão, erro quadrático, entropia, e pureza). Algumas definições [Jain et al., 1999] sobre agrupamento:

1. Um *objeto* (ou *registro*, *observação*, ou *datum*) $\mathbf{x}$ é uma única informação utilizada por um algoritmo de agrupamento. Tipicamente consiste em um vetor de $\mathbf{d}$ caracterísicas: $\mathbf{x} = (x_1, \ldots, x_d)$.

2. Um componente escalar individual x_i de um objeto $\mathbf{x}$ é chamado característica (ou atributo).

3. d é a dimensionalidade do objeto ou do espaço do objeto.

4. Um conjunto de objetos (ou registros) é representado por $\upsilon = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. O i-ésimo objeto em υ é denotado $\mathbf{x}_i = \left(x_{i,1}, \ldots, x_{i,d}\right)$. Em muitos casos o conjunto de objetos a ser agrupado é visto como uma matriz de objetos $n \times d$.

5. Uma classe pode ser entendida como o conjunto de objetos cuja distribuição no espaço de características é dado pela densidade específica da classe.

6. Técnicas de agrupamento exclusivo (*hard clustering*) associam um rótulo ℓ_i à um grupo em que cada objeto $\mathbf{x}_i$ pertença a um grupo. O conjunto de todos os rótulos de um conjunto de objetos υ é $\mathscr{L} = \{\ell_1, \ldots, \ell_n\}$, em que $\ell_i \in \{1, \ldots, k\}$, sendo k o número de grupos.

A principal distinção entre técnicas de agrupamento é se o agrupamento é aninhado ou não. Em outras palavras, se ela é particional ou hierárquica. Um algoritmo hierárquico é geralmente demonstrado graficamente através de um dendograma, que mostra a relação grupo-subgrupo e a ordem em que eles foram unidos (aglomerativo) ou divididos (divisivos). O agrupamento hierárquico pode ser visto como uma sequência de grupos particionais e um agrupamento particional pode ser obtido através de uma configuração desta sequência [Kumar et al., 2006].

Agrupamentos exclusivos associam cada objeto a um único grupo. Quando um objeto pertence a mais de um grupo, ele é caracterizado como não-exclusivo (ou sobreposto). De outra forma, um agrupamento sobreposto ou não-exclusivo, permite, que um mesmo objeto, pertença a dois ou mais grupos simultaneamente.

Em agrupamento completo, todos os objetos são associados a algum grupo, enquanto em um agrupamento parcial, isto não é necessário. Algumas vezes não necessitamos que todos os objetos pertençam à algum grupo, por exemplo, quando o conjunto de objetos tem ruído, *outliers* ou objetos que não interessam. Em outros casos, é preciso ser feito um agrupamento completo [Kumar et al., 2006].

Diz-se que os grupos formados por um algoritmo de agrupamento é heterogêneo se ele possui tamanho, formato, e/ou densidade diferente uns dos outros; o contrário são os grupos homogêneos.

2.2 Tipos de Grupos

Os diferentes tipos de grupos são definidos como [Kumar, 2000, Kumar et al., 2006]:

Grupo bem separado: é um conjunto de objetos tal que qualquer objeto em um grupo está mais próximo (ou é mais similar) a cada outro objeto no grupo do que a qualquer objeto que não pertence ao grupo (figura 2.1(a)).

Grupo baseado em centro: (ou protótipo) é um conjunto de objetos tal que qualquer objeto em um grupo está mais próximo (ou é mais similar) ao centro do grupo do que ao centro de qualquer outro grupo. O centro de um grupo pode ser um centróide, como o ponto médio dos objetos do grupo ou um medóide, o ponto mais representativo do grupo (figura 2.1(b)).

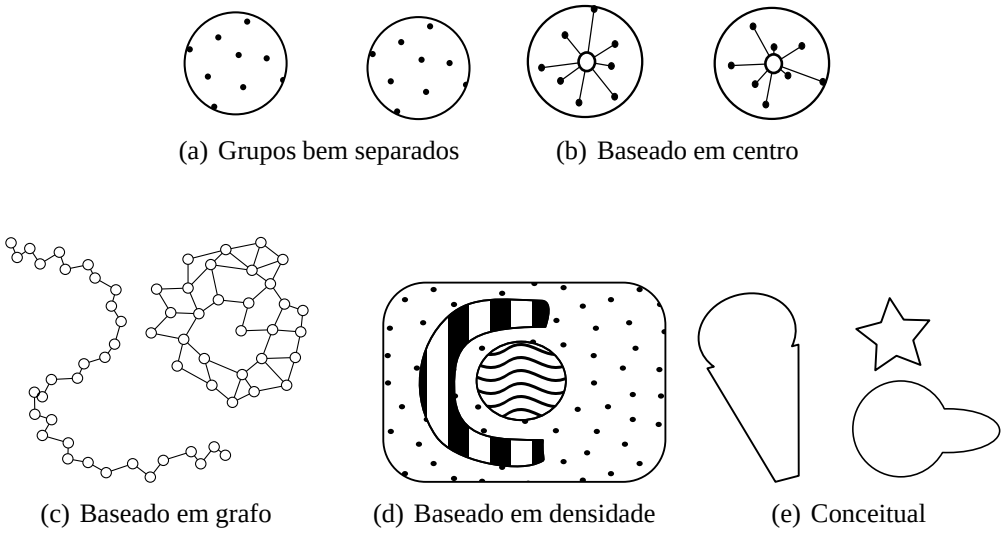
Grupo baseado em grafo: é representado em forma de grafo; os nós são os objetos e a similaridade entre eles são representados por suas arestas (figura 2.1(c)).

Grupo baseado em densidade: é uma região de objetos com uma densidade diferente de outras regiões onde há objetos com densidades diferentes (figura 2.1(d)).

Grupo conceitual: é um grupo de objetos que partilham as mesmas propriedades. Grupos podem assumir formas diferentes das descritas acima (*e.g.* uma forma geométrica em

um conjunto de objetos num espaço bidimensional). Nestes casos se utiliza técnicas de reconhecimento de padrões para realizar o agrupamento (figura 2.1(e)).

Como descrito acima, pode-se entender, que agrupamento é unir objetos de um determinado espaço em grupos que partilham as mesmas características, e, portanto, são semelhantes em algum aspecto.



(a) Grupos bem separados (b) Baseado em centro

(c) Baseado em grafo (d) Baseado em densidade (e) Conceitual

Figura 2.1: Tipos de grupos. Fonte: adaptado de [Kumar et al., 2006]

2.3 Medidas de Distância

Considerando as definições de grupo da seção 2.2, percebe-se que o principal aspecto para a realização de agrupamento é a forma de medir a distância entre objetos. Isto é dado em forma de uma função de distância, que é a medida de similaridade entre os objetos.

A medida de similaridade deve ser cuidadosamente escolhida devido à grande variedade de tipos e escalas das características. Algumas medidas de similaridade são métricas. Nem todas as medidas são consideradas métricas. As medidas de similaridade que satisfazem todas as seguintes propriedades são chamadas métricas, e as que não satisfazem os itens 4 e 5 não são métricas [Faceli et al., 2005] :

1. Para dissimilaridade: $S_{ij} = 0$ para todo i (Os pontos não são diferentes de si próprios)
   Para similaridade: $S_{ii} > maxS_{ij}$ (Os pontos são mais similares a si próprios)
2. $S_{ij} = S_{ji}$ (Simetria)
3. $S_{ij} \geq 0$ para todo i e j (Positividade)
4. $S_{ij} = 0$ somente se $i = j$
5. $S_{ik} \leq S_{ij} + Sjk$ para todo i, j e k (Desigualdade triangular)

A seguir, serão apresentado algumas medidas de similaridade.

2.3.1 Medidas de Distância Entre Objetos

A métrica de similaridade mais comum usada para objetos cujas características são contínuas é a *Distância de Minkowski* (2.1), mais especificamente suas especializações, as distâncias *Euclidiana*, *Manhattan* e *Supremum*. Estas medidas de distância são derivadas da equação 2.1 com $1 \leq p < \infty$. As métricas de Minkowski são sensíveis a variações de escala dos atributos, isto pode ser solucionado através da normalização dos atributos[1] [Jain et al., 1999, Faceli et al., 2005].

$$S_{ij} = (\sum_{k=1}^{d} \mid x_{ik} - x_{jk} \mid^{p})^{1/p} \tag{2.1}$$

Em que x é um vetor onde $x_i = \{x_{ij} \mid 1 \leq j \leq p\}$, sendo x_{ij} o valor da j-ésima característica do i-ésimo elemento, e p representa o número de características. O valor k representa um objeto do conjunto de objetos d. O mesmo se aplica as equações 2.2, 2.3, 2.4, 2.5, e 2.6.

Distância de Manhattan: $p = 1$, dado pela equação 2.2.

$$S_{ij} = \sum_{k=1}^{d} \mid x_{ik} - x_{jk} \mid \tag{2.2}$$

Distância Euclidiana: $p = 2$, dado pela equação 2.3. Apropriada para conjuntos de objetos que possuem grupos compactos ou isolados.

$$S_{ij} = (\sum_{k=1}^{d} (x_{ik} - x_{jk})^2)^{1/2} \tag{2.3}$$

Distância supremum: $p = \infty$, dado pela equação 2.4. É a diferença máxima entre quaisquer componentes dos vetores.

$$S_{ij} = \max_{1 \leq k \leq d} \mid x_{ik} - x_{jk} \mid \tag{2.4}$$

Métrica de Camberra: é dada pela equação 2.5. É muito sensível à pequenas mudanças próximas à $x_{ik} = 0 = x_{jk}$. Já possui normalização embutida.

$$S_{ij} = \sum_{k=1}^{d} \frac{\mid x_{ik} - x_{jk} \mid}{\mid x_{ik} \mid + \mid x_{jk} \mid} \tag{2.5}$$

Coeficiente de correlação de Pearson: é dado pela equação 2.6. Os valores desta medida estão no intervalo $[-1, 1]$. É insensível a diferenças na magnitude dos atributos, porém, é sensível a *outliers*.

$$S_{ij} = \frac{\sum_{k=1}^{d} (x_{ik} - \bar{x_i})(x_{jk} - \bar{x_j})}{(\sum_{k=1}^{d} (x_{ik} - \bar{x_i})^2 \sum_{l=1}^{d} (x_{jl} - \bar{x_j})^2)^{\frac{1}{2}}} \tag{2.6}$$

[1] Uma estratégia é aplicar o mesmo intervalo escalar para todos atributos.

Sendo: $\bar{x_i} = \sum_{k=1}^{d} x_{ik}/d$.

Distância de Mahalanobis:

Para atributos binários, temos o *Coeficiente de casamento simples* (eq. 2.7) e *Coeficiente de Jaccard* (eq. 2.8) [Gordon, 1999]. Considerando:

a_{11}: número de atributos com valor 1 para ambos os objetos,

a_{00}: número de atributos com valor 0 para ambos os objetos,

a_{01}: número de atributos com valor 0 para o objeto i e valor 1 para o objeto j, e

a_{10}: número de atributos com valor 1 para o objeto i e valor 0 para o objeto j,

temos:

$$S_{ij} = \frac{a_{00} + a_{11}}{a_{00} + a_{11} + a_{01} + a_{10}} = \frac{a_{00} + a_{11}}{d} \tag{2.7}$$

$$S_{ij} = \frac{a_{11}}{a_{11} + a_{01} + a_{10}} = \frac{a_{11}}{d - a_{00}} \tag{2.8}$$

Para dados nominais e ordinais, as medidas de similaridade entre pares de objetos são obtidas pela soma das contribuições individuais de todas as variáveis. A contribuição de cada objeto baseado em índices de discordância é dado por S_{ijk} entre pares de estados dos atributos categóricos, utilizado na equação da *similaridade nominal/ordinal geral* (eq. 2.9). Esta é adequada para obter a similaridade entre objetos descritos por características de diferentes tipos. [Gordon, 1999, Faceli et al., 2005]

$$S_{ij} = \sum_{k=1}^{d} s_{ijk} \tag{2.9}$$

O *coeficiente geral da similaridade* é dado pela equação 2.10.

$$S_{ij} = \frac{\sum_{k=1}^{d} w_{ijk} s_{ijk}}{\sum_{k=1}^{d} w_{ijk}} \tag{2.10}$$

em que s_{ijk} é a contribuição do k-ésimo atributo para a similaridade e w_{ijk} é 0 ou 1, dependendo se a comparação para a variável k é válida ou não.

2.3.2 Medidas de Distância Entre Grupos

Considerando N vetores i-dimensionais em um grupo: $\vec{X_i}$, sendo $i = 1, 2, \ldots, N$. Utilizando o conceito de centróide, considerando $\vec{x_0}$, R, e D, centróide (2.11), raio (2.12), e diâmetro (2.13), respectivamente, temos:

$$\vec{x_0} = \frac{\sum_{i=1}^{n_k} \vec{x_i}}{n_k} \tag{2.11}$$

$$R = \left(\frac{\sum_{i=1}^{n_k} (\vec{x_i} - \vec{x_0})^2}{n_k}\right)^{1/2} \tag{2.12}$$

$$D = \left(\frac{\sum_{i=1}^{n_k}\sum_{j=1}^{n_k}(\vec{x_i}-\vec{x_j})^2}{n_k(n_k-1)}\right)^{1/2} \tag{2.13}$$

Considerando N_1 objetos i-dimensionais em um grupo: $\vec{X_i}$, sendo $i = 1,2,\ldots,N_1$, e N_2 objetos em outro cluster: $\vec{X_j}$, sendo $j = N_1+1, N_1+2, ..., N_1+N_2$, podem ser definidas as seguintes distâncias entre dois grupos [Zhang et al., 1996]:

$$D_0 = \left((\vec{x_{01}}-\vec{x_{02}})^2\right)^{1/2} \tag{2.14}$$

$$D_1 = \mid \vec{x_{01}}-\vec{x_{02}} \mid \tag{2.15}$$

$$D_2 = \left(\frac{\sum_{i=1}^{n_{k1}}\sum_{j=n_{k1}+1}^{n_{k1}+n_{k2}}(\vec{x_i}-\vec{x_j})^2}{n_{k1}n_{k2}}\right)^{1/2} \tag{2.16}$$

$$D_3 = \left(\frac{\sum_{i=1}^{n_{k1}+n_{k2}}\sum_{j=1}^{n_{k1}+n_{k2}}(\vec{x_i}-\vec{x_j})^2}{(n_{k1}+n_{k2})(n_{k1}+n_{k2}-1)}\right)^{1/2} \tag{2.17}$$

$$D_4 = \sum_{k=1}^{n_{k1}+n_{k2}}(\vec{x_k}-\frac{\sum_{l=1}^{n_{k1}+n_{k2}}\vec{x_l}}{n_{k1}+n_{k2}})^2 - \sum_{i=1}^{n_{k1}}(\vec{x_i}-\frac{\sum_{l=1}^{n_{k1}}\vec{x_l}}{n_{k1}})^2 - \sum_{j=n_{k1}+1}^{n_{k1}+n_{k2}}(\vec{x_j}-\frac{\sum_{l=n_{k1}+1}^{n_{k1}+n_{k2}}\vec{x_l}}{n_{k2}})^2 \tag{2.18}$$

As equações representam: eq. 2.14 a distância euclidiana entre centróides, eq. 2.15 a distância *manhattan* entre centróide, eq. 2.16 a média inter-grupos, eq. 2.17 a média intra-grupos, 2.18 a *variance increase* entre grupos. As medidas x_0, R, e D são propriedades de um grupo, e D_0, D_1, D_2, D_3, e D_4 são propriedades entre dois grupos [Zhang et al., 1996].

2.4 Agrupamento Utilizando Redes Neurais

As redes neurais artificiais (RNA), também chamadas de *Artificial Neural Networks* (ANN) ou *Neural Networks* (NN), podem ser utilizadas para realizar agrupamento de objetos. Nesta subseção apresentaremos ao leitor a definição de uma RNA; o elemento básico (neurônio), as arquiteturas típicas, e seu uso para realização de agrupamento de objetos.

2.4.1 Redes Neurais Artificiais

Uma Rede Neural Artificial é um sistema de processamento de informações que tem certas características em comum com as redes neurais biológicas. Redes neurais artificiais vem sendo desenvolvidas a partir de generalizações de modelos matemáticos dos sistemas de cognição humanos, baseado nas seguintes pressuposições [Fausett, 1994]:

1. O processamento de informações ocorre em vários elementos simples chamados neurônios.

2. Sinais são passados entre os neurônios através de conexões (arestas, ou sinapses).

3. Cada conexão tem um peso, que, em uma rede neural típica, multiplica o sinal transmitido.

4. Cada neurônio aplica uma função de ativação (geralmente não-linear) à sua entrada (sinal de entrada aplicado ao peso) para determinar a saída.

A rede neural consiste em um grande número de elementos simples, chamados *neurônios*, *unidades*, *células*, ou *nós*.

Neurônios

Cada neurônio é ligado a outros neurônios através de conexões, cada uma associada a um peso. Os pesos representam as informações necessárias para resolver um problema. Cada neurônio têm um estado interno chamado *ativação* ou *nível de atividade*, que é uma função do sinal de entrada. Um neurônio só pode enviar um sinal por vez, mas pode ter vários receptores simultâneos (figura 2.2).

Uma conexão de uma unidade j para unidade i serve para propagar a *ativação* a_j de j a i. Cada conexão tem um peso $W_{j,i}$ associado que determina a força da conexão. A entrada de um neurônio i é a soma ponderada de seus sinais de entrada:

$$in_i = \sum_{j=0}^{n} W_{j,i} a_j \tag{2.19}$$

O neurônio aplica a função de ativação à esta soma (equação 2.19) para gerar a saída:

$$a_i = g(in_i) = g\left(\sum_{j=0}^{n} W_{j,i} a_j\right) \tag{2.20}$$

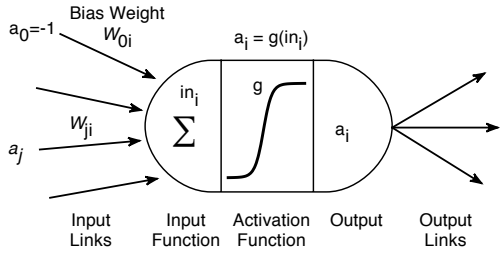
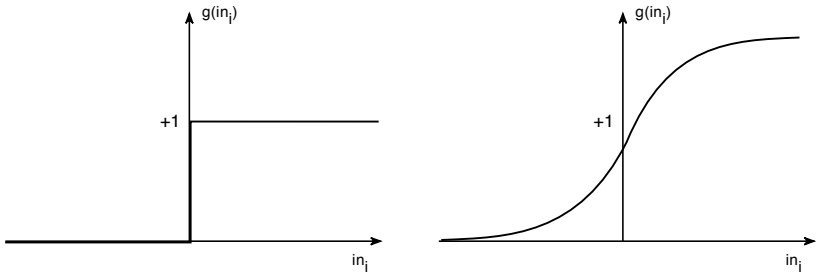


Figura 2.2: Modelo matemático de um neurônio. Fonte: [Russell et al., 1995]

Função de Ativação

A função de ativação g é projetada para determinar dois critérios. Primeiro, o neurônio deve estar "ativo" (próximo de +1) quando entradas "corretas" são apresentadas, e "inativo" (próximo de 0) quando entradas "erradas" são apresentadas. Segundo, a ativação deve ser não-linear, senão toda a rede se torna uma função linear [Russell et al., 1995].

Duas opções para g são demonstradas na figura 2.3, a função *threshold* (limiar) e a função sigmoidal. A função sigmoidal tem a vantagem de ser diferenciável, o que a torna importante para o ajuste de pesos [Russell et al., 1995].



(a) Função de ativação *threshold*. Retorna 1 quando a entrada é positiva e 0 caso contrário.

(b) Função de ativação sigmoidal. $1/(1+e^{-x})$

Figura 2.3: Funções de ativação do neurônio. Fonte: [Russell et al., 1995]

2.4.2 Arquiteturas Típicas

O arranjo dos neurônios em camadas (*layers*) e os padrões de conexão entre as camadas é chamado de arquitetura da rede. Muitas redes tem uma camada de entrada em que a ativação destas é o sinal de entrada de outros neurônios.

Geralmente as redes são classificadas em camada única, ou multi-camada. A camada de entrada não é contada como uma camada, porque não realiza computações. Desta forma é equivalente contar quantas camadas há na rede através dos pesos que as ligam, *e.g.* uma rede de 2 camadas e a camada de entrada, tem 2 "camadas" de arestas para ligar as camadas de neurônios.

Basicamente, temos três tipos de arquitetura:

- *Rede de camada única*: Esta rede tem somente uma camada de conexões de peso, sendo uma camada de entrada, que recebe os sinais do mundo, e uma camada de saída, por onde a saída da rede pode ser lida. Em uma típica rede de camada única, cada neurônio da camada de entrada está ligado com todos neurônios da camada de saída.

- *Rede multi-camadas*: Uma rede multi-camadas é uma rede com um ou mais camadas entre as camadas de entrada e saída (camada escondida). As redes multi-camadas podem resolver problemas mais complexos que as de camada única.

- *Camadas competitivas*: Estas redes são usadas, em geral, para agrupamento. Envolve um grande número de tipos de redes neurais artificiais, inclusive SOM, que será utilizada neste trabalho e descrita na seção 4.1

2.4.3 Aprendizagem não-supervisionada

Agrupamento é o processo pelo qual se divide um conjunto de objetos em grupos. O processo de agrupamento pode ser usado como classificação sendo que os rótulos dos grupos são definidos somente a partir dos próprios objetos. Em contrapartida, o processo de classificação é realizado utilizando-se rótulos fornecidos junto com a base de treino, este caso é denominado *aprendizagem supervisionada*.

A *aprendizagem não-supervisionada* de uma rede neural ocorre quando se apresentam vetores de características sem os seus respectivos rótulos. Neste caso haverá o agrupamento, ao invés do treinamento supervisionado (classificação).

Existem diversos algoritmos para realizar a aprendizagem não-supervisionada. Dentre alguns podemos citar: K-MEANS [Jain, 2010] (X-MEANS [Pelleg and Moore, 2000]), DBSCAN [Ester et al., 1996], COBWEB [Fisher, 1987]. Dentre os algoritmos que utilizam redes neurais podemos citar dois mais utilizados: O ART-2 [Carpenter et al., 1987] e SOM [Kohonen, 1990].

O algoritmo SOM, descrito no capítulo 4, é uma rede neural auto-organizável utilizada para agrupamento. A rede modifica seus pesos para que os objetos mais similares tenham a mesma saída, as quais representam os grupos. A rede neural vai produzir um vetor representativo para cada grupo formado - após o processo de refinamento da rede (apresentado no capítulo 4).

2.5 Validação de Grupos

A validação de grupos é a análise do resultado de um algoritmo de agrupamento - o quão bons são os grupos resultantes. Nela pode-se verificar se estruturas aleatórias estão contidas nos grupos finais, comparar taxa de acerto através de dados conhecidos (rotulados), comparar resultados para diferentes bases de dados e diferentes algoritmos.

No caso do algoritmo k-means pode-se utilizar o SSE (erro quadrático) como forma de avaliação dos grupos, mas para grupos baseados em densidade (que podem tomar quaisquer formas) o SSE pode não funcionar tão bem. Algoritmos de agrupamento irão encontrar grupos mesmo onde não há. Em espaços bi-dimensionais é mais fácil verificar a qualidade dos grupos, porém, quando o número de dimensões aumenta, estes problemas podem não ser facilmente detectados.

Portanto, existem medidas numéricas para avaliar os grupos, que podem ser classificadas em três critérios [Kumar et al., 2006]: *não supervisionado*, *supervisionado* e *relativo*.

2.5.1 Validação Não Supervisionada de Grupos

O critério não-supervisionado é usado para medir a qualidade dos grupos sem levar em consideração informações externas. Um exemplo é SSE. O índice interno pode ser usado para medir *coesão* e *separação*. O primeiro mede o relacionamento dos objetos pertencentes ao grupo, enquanto o segundo mede o quão distante é um grupo dos outros.

Pode-se considerar a validade de um agrupamento como:

$$V = \sum_{i=1}^{K} w_i validade(C_i) \tag{2.21}$$

sendo K o número de grupos, w_i o peso, e , *validade*, uma função de coesão, separação ou alguma combinação destas. Os pesos variam dependendo da função de validade. Se a função de validade é coesão, então resultados maiores são melhores. Se é separação, então resultados menores são melhores [Kumar et al., 2006]. A tabela 2.1 mostra alguns valores de pesos e medidas.

Coesão e Separação

De forma a conseguir as medidas de coesão e separação de agrupamentos, é necessário levar em consideração as particularidades entre grupos baseados em grafos, e grupos baseados em protótipo (centróide ou medóide).

1. Para grupos baseados em grafos, a coesão é calculada somando-se os pesos das arestas que conectam o grupo (figura 2.4(a)). De forma similar, a separação é calculada somando-se os pesos das arestas dos objetos de um grupo conectados a outro grupo (figura 2.4(b)). As medidas de coesão e separação são expressas pelas equações 2.22 e 2.23.

$$co(C_i) = \sum_{\substack{x \in C_i \\ y \in C_i}} p(x,y) \tag{2.22}$$

$$se(C_i, C_j) = \sum_{\substack{x \in C_i \\ y \in C_j}} p(x,y) \tag{2.23}$$

   sendo p, uma função de similaridade, disissimilaridade ou outra medida.

2. Para grupos baseados em protótipos, a coesão é calculada somando-se as medidas de proximidade dos objetos ao protótipo (centróide ou medóide) do grupo. De forma similar, a separação é calculada, somando-se as medidas de proximidade de dois protótipos (figuras 2.4(c) e 2.4(d)). Neste caso, a medida de coesão é expressa pela equação 2.24, e as medidas de separação são expressas pelas equações 2.25 e 2.26.

$$co(C_i) = \sum_{x \in C_i} p(x, c_i) \tag{2.24}$$

$$se(C_i, C_j) = p(c_i, c_j) \tag{2.25}$$

$$se(C_i) = p(c_i, c) \tag{2.26}$$

   sendo p, uma função de similaridade, dissimilaridade ou outra medida; c_i um protótipo do grupo C_i, e c um protótipo global. Existem duas medidas de separação: uma entre dois protótipos de grupos (2.25) e outra, entre um protótipo de grupo e o protótipo global (2.26). A separação entre protótipos de grupos está diretamente relacionada à separação entre protótipos de grupo e o protótipo global. A tabela 2.1 apresenta exemplos de medidas de validação, seu tipo e peso utilizados.

Coeficiente de Silhueta

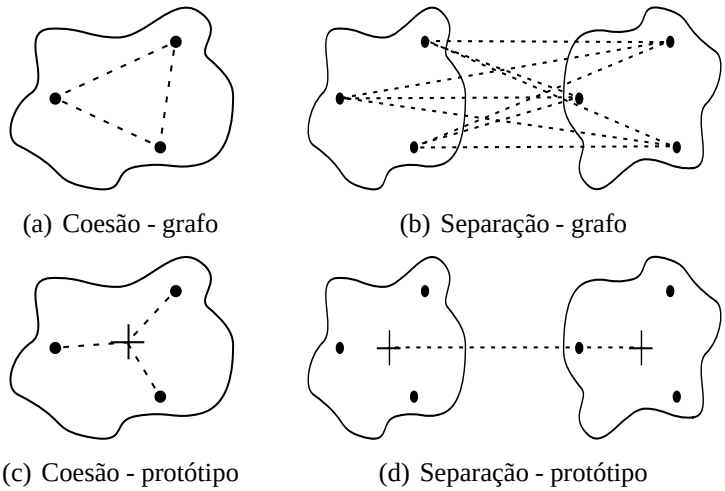
O método de silhueta combina ambos os métodos de coesão e separação. A seguir, descreve-se como calcular o método de silhueta para um dado objeto de um grupo.

1. Para o objeto em questão, calcular a similaridade média entre todos os objetos do grupo ao qual ele pertence. Este é o valor a_i.

2. Calcular a similaridade média a todos os objetos dos outros grupos e encontrar o menor valor. Este é o valor b_i.

Medida	Peso	Tipo
$\sum_{\substack{x\in C_i \\ y\in C_j}} p(x,y)$	$\frac{1}{m_i}$	grafo, coesão
$\sum_{x\in C_i} p(x,c_i)$	1	protótipo, coesão
$p(c_i,c)$	m_i	protótipo, separação
$\sum_{\substack{j=1 \\ j\neq i}}^{k} \sum_{\substack{x\in C_i \\ y\in C_j}} p(x,y)$	$\frac{1}{\sum_{\substack{x\in C_i \\ y\in C_j}} p(x,y)}$	grafo, separação e coesão

Tabela 2.1: Medidas de coesão e separação



(a) Coesão - grafo (b) Separação - grafo

(c) Coesão - protótipo (d) Separação - protótipo

Figura 2.4: Representação de coesão e separação em grupos. Fonte: adaptado de[Kumar et al., 2006]

3. Calcular o coeficiente de silhueta para o objeto através de $s_i = (b_i - a_i)/max(a_i, b_i)$.

O valor do coeficiente de silhueta varia entre -1 e 1. Um valor negativo não é desejado, pois, corresponde ao caso em que a_i, a similaridade média para os pontos no grupo, é maior que b_i, a menor média de similaridade entre pontos de outro grupo. O melhor é que o coeficiente de silhueta seja positivo $(a_i < b_i)$ e que a_i seja o mais próximo de 0 possível, desde que, o coeficiente assuma valor 1 quando $a_i = 0$ [Kumar et al., 2006].

Pode-se calcular a média dos coeficientes dos objetos de um grupo para ter a média deste grupo. Também pode-se calcular a média total do agrupamento calculando-se a média de todos os objetos do conjunto.

Matriz de Similaridade

A matriz de similaridade pode ser obtida calculando-se a similaridade entre duplas de objetos e colocando-as em forma de matriz. Os pontos podem ser identificados nas colunas e linhas, seus cruzamentos representam a similaridade entre estes objetos. Dado o conjunto de objetos $\upsilon = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, podemos representar a matriz de similaridade (de tamanho $i \times j$) como:

$$\begin{array}{cccccc} & x_1 & x_2 & x_3 & \dots & x_j \\ x_1 & x_{11} & x_{12} & x_{13} & \dots & x_{1j} \\ x_2 & x_{21} & x_{22} & x_{23} & \dots & x_{2j} \\ x_3 & x_{31} & x_{32} & x_{33} & \dots & x_{3j} \\ \vdots & & & & \ddots & \vdots \\ x_i & x_{i1} & x_{i2} & x_{i3} & \dots & x_{ij} \end{array} \tag{2.27}$$

É possível validar um agrupamento através de matrizes de proximidade de duas maneiras. Podemos analisar a correlação entre a matriz de similaridade e a matriz de similaridade ideal ou através da visualização gráfica.

Correlação: é possível analisar a qualidade de um agrupamento calculando-se a correlação entre a matriz de similaridade dos objetos e matriz de similaridade ideal. Na matriz de similaridade ideal os pontos pertencentes ao mesmo grupo tem similaridade 1, e pontos de outros grupos, 0. Uma correlação alta entre as matrizes indica que os pontos pertencentes ao mesmo grupo estão mais próximos, logo, uma correlação alta significa maior qualidade. Entretanto, para grupos baseados em grafo, em densidade e conceituais, esta técnica pode não funcionar devido a forma que os grupos podem tomar.

Representação visual: utiliza-se um gráfico bidimensional, onde os eixos x e y representam todos os objetos do agrupamento. A interseção de dois destes objetos representa a similaridade entre eles. A similaridade é representada entre 0 e 1, sendo 0 menos similar e 1 mais similar. Quanto menor a similaridade, mais fraco é o tom da cor utilizada e quanto maior, mais forte é o tom. Por fim as linhas e colunas são organizadas de forma que os objetos pertencentes aos mesmos grupos fiquem juntos (figura 2.5).

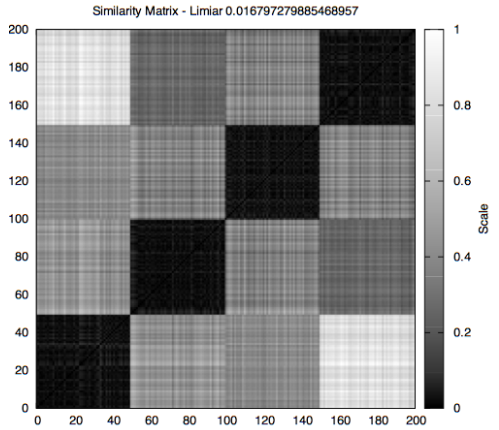


Figura 2.5: Exemplo de matriz de similaridade.

Distância Cofenética

Distância cofenética entre dois objetos: é a proximidade em que um algoritmo de agrupamento hierárquico aglomerativo posiciona os objetos no mesmo grupo pela primeira vez. Em tal matriz, os dados são distâncias cofenéticas entre os pares de objetos [Kumar et al., 2006].

Através da matriz de distância cofenética, pode-se calcular o Coeficiente de Correlação Cofenético (CPCC - *CoPhenetic Correlation Coefficient*), que é a correlação entre esta matriz e a matriz de dissimilaridade original.

Esta técnica é usada para validar qual algoritmo de agrupamento hierárquico é melhor para uma base de dados em especial.

2.5.2 Validação Supervisionada de Grupos

O critério supervisionado é usado para verificar se um grupo descoberto por um algoritmo de agrupamento está de acordo com outro grupo externo fornecido (entropia). É supervisionado ou externo pois utiliza informações não presentes na base de dados utilizada. Temos dois tipos de medidas nesta seção, as que utilizam classificação e as que utilizam medidas de similaridade.

Orientado a Classificação

Nesta forma de validação é medido o quanto as classes encontradas com determinado algoritmo correspondem às classes previamente fornecidas.

Entropia: mede o quanto um único grupo consiste de objetos de uma única classe. Primeiro, calcula-se a probabilidade de que um membro do grupo i pertença ao grupo j, conforme equação 2.28, sendo mi o número de objetos no grupo i e m_{ij} é o número de objetos da classe j no grupo i.

$$p_{ij} = m_{ij}/m_i \tag{2.28}$$

A entropia de cada grupo é calculada conforme equação 2.29, sendo L o número de classes. A entropia total é a soma das entropias de cada grupo, considerando o tamanho de cada grupo, conforme equação 2.30, em que K é o número de grupos e m é o número total de objetos.

$$e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij} \tag{2.29}$$

$$e = \sum_{i=1}^{K} \frac{m_i}{m} e_i \tag{2.30}$$

Pureza: é outra medida para determinar se um grupo contém objetos de uma única classe. A pureza de um grupo i é calculada através de $p_i = max_j p_{ij}$ e a pureza total do agrupamento é $p = \sum_{i=1}^{K} \frac{m_i}{m} p_i$.

Precisão: é a fração do grupo que corresponde a uma dada classe. A precisão do grupo i e classe j é $p(i,j) = p_{ij}$, conforme 2.28.

Recall: verifica se o grupo contém todos os objetos de uma classe específica. O recall do grupo i e classe j é $r(i,j) = m_{ij}/m_j$, sendo m_j, o número de objetos na classe j.

Medida-f: é uma combinação de precisão e recall. Calcula-se através da equação 2.31.

$$F(i,j) = \frac{2 \times p(i,j) \times r(i,j)}{p(i,j) + r(i,j)} \tag{2.31}$$

Orientado a Similaridade

Pode-se comparar duas matrizes, a matriz de similaridade do agrupamento ideal (já comentada na seção 2.5.1) e a matriz de similaridade de classe ideal. Como trata-se de um modo de validação supervisionada tem-se acesso às classes corretas, portanto, podem-se gerar a matriz de similaridade de classe ideal, esta matriz contém 1 onde dois objetos pertencem à mesma classe e 0 caso contrário.

Tendo essas duas matrizes, pode-se calcular a correlação entre elas e também as medidas de Jaccard e Rand. A correlação é calculada normalmente. Entretanto, para calcular Jaccard e Rand, usam-se os valores de $f_{00}, f_{10}, f_{10},$ e f_{11}, que são:

f_{00} número de pares de objetos que tem grupos diferentes e grupos diferentes;

f_{01} número de pares de objetos que tem grupos diferentes e mesmo grupo;

f_{10} número de pares de objetos que tem grupos iguais e diferentes grupos; e

f_{11} número de pares de objetos que tem grupos iguais e grupos iguais.

Com isso, calcula-se o coeficiente de Jaccard (equação 2.32) e o coeficiente de Rand (equação 2.33).

$$Jaccard = \frac{f_{11}}{f_{01} + f_{10} + f_{11}} \tag{2.32}$$

$$Rand = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}} \tag{2.33}$$

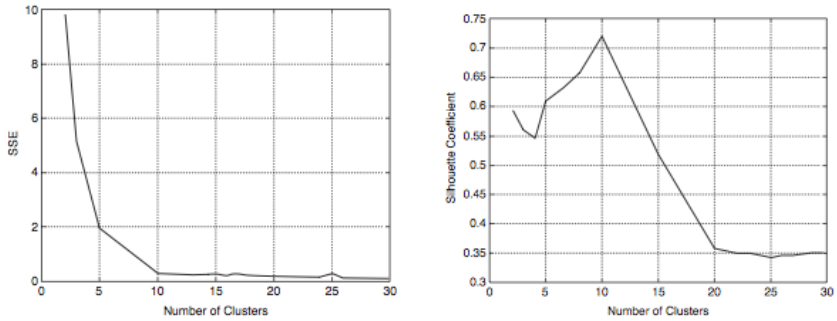
2.5.3 Critério Relativo da Validação de Grupos

É usado para comparação entre diferentes agrupamentos ou grupos. Pode ser tanto supervisionado, como não-supervisionado. Pode-se, por exemplo, comparar dois algoritmos de agrupamento utilizando SSE ou entropia.

2.5.4 Estimando o Número Correto de Grupos

Pode-se plotar um gráfico da medida de validação contra o número de grupos e procurar por um pico, depressão ou “joelho” no gráfico, que indica o número ideal de grupos [Kumar et al., 2006]. Este critério não funciona sempre, pois, alguns grupos podem estar muito juntos ou sobrepostos. É possível utilizar mais de uma medida, como SSE e o coeficiente de silhueta para tentar contornar este problema. Desta forma já pode-se ter uma idéia do número de grupos no agrupamento.

As figuras 2.6(a) e 2.6(b), encontrados em [Kumar et al., 2006], apresentam o número de grupos contra a medida de coesão e silhueta, respectivamente. Ambos gerados a partir de uma base de dados artificial com 5 pares de grupos (portanto, 10 grupos) e utilizando o algoritmo KMEANS para realizar o agrupamento. Percebe-se há uma depressão ou “joelho” no ponto 10, referente ao número correto de grupos no agrupamento.



(a) SSE vs número de grupos. Extraído de [Kumar et al., 2006]

(b) Silhueta média vs número de grupos. Extraído de [Kumar et al., 2006]

Figura 2.6: Estimando número de grupos

2.5.5 Tendência do Agrupamento

Trata-se de descobrir se determinado agrupamento possui grupos reais. A maneira mais óbvia é tentar agrupar; o problema é que a maioria dos algoritmos vão gerar grupos mesmo onde não há. Pode-se, ainda, testar a qualidade dos grupos, porém, o conjunto pode conter grupos de diferentes tipos. Para resolver este problema, pode-se testar vários algoritmos e aplicar algum algoritmo de validação, já comentado anteriormente. Outra forma, é utilizar uma abordagem estatística, como a equação de Hopkins (2.34) [Kumar et al., 2006]. Nesta técnica, deve-se gerar p registros uniformemente distribuídos pelo espaço e utilizar p registros da base real.

$$H = \frac{\sum_{i=1}^{p} w_i}{\sum_{i=1}^{p} u_i + \sum_{i=1}^{p} w_i} \tag{2.34}$$

sendo u_i as distâncias da vizinhança mais próxima dos dados gerados artificialmente, e w_i dos dados reais. Se u_i e w_i forem semelhantes (significa que o agrupamento está próximo dos dados gerados artificialmente), H deve ser próximo de 0.5. Valores de H próximos de 0 indicam que os dados estão devidamente agrupados, e próximos de 1 que os dados estão distribuídos uniformemente pelo espaço. Esta equação, mede a diferença entre os objetos do conjunto e objetos aleatórios, ou seja, mede a aleatoriedade dos objetos.

2.6 Agrupamento em Fluxo de Dados

Em muitas aplicações recentes, o conceito de fluxo de dados é mais apropriado do que conjunto (base) de dados fixo. Um modelo de base de dados fixo é apropriado para situações onde atualizações da base são pequenas ou pouco frequentes. Em fluxo de dados utiliza-se o conceito *agrupamento em fluxo de dados*, também chamado de aprendizagem incremental.

Com o crescimento das bases de dados e a necessidade de extrair informação destes, muitos algoritmos foram propostos [Ester et al., 1996, Guha et al., 1999, Dutta et al., 2005, Guha et al., 1998, Arai, 2007, Karypis and Kumar, 1999, Wu et al., 2008]. Entretanto, tais algoritmos trabalham com bases de dados completas e sempre disponíveis. Em alguns casos, a geração de dados é muito rápida. Estes dados, muitas vezes, não podem ser armazenados devido a limitação de recursos, *i.e.* capacidade de armazenamento e comu-

nicação. Esta grande quantidade de dados é tratada como um *fluxo de dados*. Técnicas de mineração de dados para lidar com estes fluxos foram propostos nos últimos anos [Gaber et al., 2005, Guha et al., 2003, Guha et al., 2000, Beringer and Hullermeier, 2006, Babcock et al., 2002, Golab and Ozsu, 2003].

O agrupamento contínuo está relacionado à extração de estruturas de conhecimento representados em modelos e padrões através de um fluxo contínuo de dados [Gaber et al., 2005]. O fluxo de dados ocorre em tempo real, contínuo, e ordenado (pelo tempo de chegada). Não é possível controlar a ordem em que os dados surgem, assim como não é possível armazenar todo o fluxo [Golab and Ozsu, 2003].

Este fluxo contínuo é uma sequência ordenada de pontos que pode ser lida em ordem e somente uma ou um pequeno número de vezes. Formalmente, um fluxo de dados é uma sequência de objetos $x_{i-1}, x_i, \ldots, x_{i+1}, \ldots$ ordenada pelo índice i. [Guha et al., 2000, Guha et al., 2003]. Uma leitura em um fluxo de dados é chamado *passo* ou *leitura linear*. Em fluxos de dados transientes, onde os dados não são armazenados em disco, somente um passo é possível. Devido a grande quantidade de informação, não é possível para um algoritmo de agrupamento contínuo guardar os antigos passos, logo, é necessário guardar apenas sumários[2] dos dados passados (deixando mais memória para processar os futuros dados). O critério para julgar um algoritmo incremental é dado pelo número de leituras (passos), assim como o tempo e memória usados [Guha et al., 2003].

2.7 Controle do fluxo de dados

O controle para os problemas envolvendo fluxo de dados geralmente utilizam duas abordagens: estatística e computacional. Estas podem ser, respectivamente, *baseadas em dados* e *baseadas em tarefas* [Gaber et al., 2005].

O controle baseado em dados, a idéia é analisar somente um subconjunto do total de objetos de forma a conseguir uma representação dos dados menor. Isso pode ser feito representando os dados em um sumário ou escolhendo um conjunto menor de dados à partir do conjunto original.

Agrupamento baseado em tarefas são técnicas que modificam outras já existentes ou criam novas para dar conta do processamento de fluxo de dados, são utilizadas técnicas para alcançar soluções eficientes em tempo e espaço.

Amostragem é a técnica utilizada na estatística em que se remove indivíduos de uma dada população de forma que esse subconjunto represente o conjunto total. A taxa de erro de uma dada computação desse subconjunto é dado em função da taxa de amostragem. Assim, uma resposta aproximada pode ser obtida. [Babcock et al., 2002].

Muitos servidores de arquivos, telefonia, etc, sofrem de picos no fluxo de dados. São os momentos em que a taxa de transferência de dados é uma ordem de magnitude maior que o fluxo típico. Em tais ocasiões os sistemas podem sobrecarregar e deixar de funcionar corretamente devido à falta de recursos.

O **derramamento de carga** é ato de descartar parte dos dados não processados para que o sistema continue retornando respostas atualizadas [Babcock et al., 2004]. O derramamento de carga é difícil de ser utilizado com algoritmos de mineração de dados pois ele descarta

[2] Pode ser entendido como o resultado encontrado até o momento, *i.e.*, o resultado encontrado em um passo.

partes do fluxo de dados que poderiam ser usados para gerar modelos ou poderia representar padrões de interesse em análises de séries temporais [Gaber et al., 2005].

O **esboço** é realizado utilizando parte dos dados de entrada do fluxo para representar o todo. Pode ser feito escolhendo-se aleatoriamente um subconjunto das características recebidas, reduzindo assim a dimensionalidade. Esta técnica é considerada um esboço do fluxo [Gaber et al., 2005, Muthukrishnan, 2005].

De forma mais genérica, manter um sumário do fluxo de dados através da redução do mesmo permite encontrar respostas próximas; assim, diminui-se o tempo computacional e memória para as *queries* [Babcock et al., 2004].

Agregação é um processo estatístico, *i.e.*, média e variância, que sumariza o fluxo de dados. Estas informações sumarizadas podem ser utilizadas em algoritmos de data mining. Em [Gaber et al., 2005], o autor considera que a performance não é muito boa onde o fluxo de dados é muito flutuante (instável). Isso ocorre pois ao realizar uma média de um conjunto de elementos onde há uma variância alta, a média pode não representar bem todos os elementos. Entretanto, existem outras formas de realizar a agregação, como descrito em [Aggarwal et al., 2003], onde o autor utiliza a soma dos quadrados dos valores e a soma dos valores recebidos durante um período de tempo.

2.7.1 Janela de Leitura

Assumindo que os registros mais recentes são mais importantes que os antigos, podemos utilizar o conceito de *janela de leitura* (ou *janela deslizante*) para tratá-las. Uma janela de leitura é uma janela temporal que representa uma subsequência do fluxo de dados completo.

Uma janela pode ser dividida em m subjanelas de tamanho v, conforme figura 2.7.

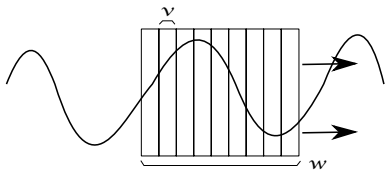


Figura 2.7: Janela Deslizante e v subjanelas

Desta forma o tamanho da janela é representado por $w = m \times v$. Utilizando as notações de [Beringer and Hullermeier, 2006], representamos o fluxo de dados através do vetor X,

$$X = (\underbrace{x_0, x_1, \ldots, x_{v-1}}_{B_1} \mid \underbrace{X_v, X_{v+1} \ldots, x_{2v-1})}_{B_2} \mid \ldots \mid \underbrace{x_{(m-1)v}, x_{m-1}v+1, \ldots, x_{w-1})}_{B_m} \tag{2.35}$$

sendo x_i uma observação, e $B_1, B_2, \ldots B_m$ atualizações do fluxo. A atualização do conceito do fluxo é dado em função de v, ou seja, a cada v novas observações. Uma atualização no fluxo é representada da seguinte maneira:

$$\begin{aligned} X &= B_1 \mid B_2 \mid B_3 \mid \ldots \mid B_{m-1} \mid B_m \\ X' &= B_2 \mid B_3 \mid \ldots \mid B_{m-1} \mid B_{m+1} \end{aligned} \tag{2.36}$$

sendo X' o próximo estado após X.

2.7.2 Filas

O fluxo de dados recebido pode ser inserido em uma fila, onde será realizado o controle (consumo e perda de dados). Os dados de chegada são armazenados em uma fila, sem prioridade, de tamanho relativo à disponibilidade de recursos, cuja disciplina é *First come, First served*.

A fila possui uma taxa de chegada *e* (registros por segundo) que depende da quantidade de usuários conectados, velocidade da geração dos dados, etc; e a taxa de saída *s* que corresponde ao número de registros utilizados em agrupamentos por segundo e varia conforme velocidade/configuração do algoritmo de agrupamento e tamanho da janela de leitura.

Se $(s-e) > 0$ a fila está estável, caso contrário a fila está instável e irá ficar cheia devido à pouca vazão. Para calcular o tempo restante até a fila encher, calcula-se: $\frac{f}{s-e}$, sendo f o tamanho da fila.

2.8 Pesquisas Envolvendo Agrupamento em Fluxo de Dados

Nesta seção serão apresentados trabalhos relacionados a agrupamento em fluxo de dados. Diversos algoritmos e métodos existem na literatura, abaixo, descrevemos alguns deles.

Algorithm Output Granularity é a primeira estratégia proposta para mineração em fluxo de dados que leva em consideração a flutuação dos dados, memória e velocidade de processamento. AOG tem basicamente 3 estágios: os primeiros dois estão relacionados à mineração e adaptação do algoritmo aos recursos disponíveis, assim como a taxa de fluxo de dados; a terceira está relacionada a sumarizar os dados quando há pouca memória disponível [Gaber et al., 2003].

O AOG cobre alguns dos problemas enfrentados quando trabalhamos com fluxo de dados. Os parâmetros do AOG são: *a*) limiar: parâmetro para controlar a criação de novas saídas levanto em consideração *memória*, *tempo* para preencher memória restante e taxa de fluxo de dados; *b*) limite inferior do limiar: quanto menor, maior a precisão e o tempo de processamento; *c*) limite superior do limiar: quanto maior, menor a precisão e o tempo de processamento; *d*) granularidade da saída: quantidade de resultados aceitáveis de acordo com uma medida de precisão pré-determinada (resultados estes que devem estar na memória antes de serem sumarizados); *e*) limiar de tempo: é o tempo necessário para gerar os resultados antes de qualquer sumarização (pode ser especificado ou calculado automaticamente); *f*) janela de tempo: é o tempo entre duas leituras lineares.

Outra abordagem é o algoritmo BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*) usado para agrupamento em grandes conjuntos de dados [Zhang et al., 1996]. Este algoritmo não trata especificamente de agrupamento em fluxo de dados, entretanto, ele tem algumas características que o permitem ser usado em fluxos de dados. Este algoritmo tenta produzir agrupamentos de melhor qualidade possível dinamicamente e incrementalmente com os recursos disponíveis (*i.e.* memória e tempo disponíveis). BIRCH consegue encontrar um bom agrupamento com uma única leitura dos dados, e melhora esse agrupamento com algumas leituras a mais. Por meio desta característica de não necessitar de múltiplas leituras dos mesmos dados, BIRCH também pode ser usado em fluxo de dados. Este algoritmo ainda consegue lidar com ruídos na base.

Em [Aggarwal et al., 2003], o autor leva em consideração a evolução dos dados presente no fluxo. O fluxo de dados é visto como um processo infinito em que os dados evoluem com o tempo. Como resultado, os grupos relacionados podem também mudar consideravelmente com o tempo. O autor utiliza duas técnicas chamadas *micro-cluster* e *macro-cluster*. O *micro-cluster* é um resumo do agrupamento realizado naquele instante (o autor chama de *snapshot* cada instante em que é armazenado um sumário dos dados). Em instantes específicos, o algoritmo realiza os *snapshots*, de forma que em um tempo mais recente a granularidade seja maior - desta forma, dando mais prioridade para os *micro-clusters* recentes. Em outro momento, o *macro-cluster* é utilizado, determinando a evolução do fluxo (em quais momentos apareceram, sumiram, ou grupos se uniram). Dado a frequência com que os *micro-clusters* são armazenados, este algoritmo pode representar o agrupamento de um longo fluxo de dados com o consumo de memória muito baixo.

Outros dois algoritmos para agrupamento em fluxo de dados são apresentados em [O' Callaghan et al., 2002]: *StreamKM* (*Stream k-median*) e *Stream LSearch*. O primeiro é basicamente o k-means modificado para ser utilizado em fluxo de dados, o segundo utiliza o algoritmo LSearch como parte do procedimento. Ambos trabalham tentando diminuir a soma do erro quadrático das distâncias dos pontos até os centróides. Este trabalho mostrou bons resultados quando comparado ao BIRCH.

Em particular, [Longo and Barrett, 2009] propôs um método para agrupar usuários de websites baseado em seus registros de comportamento. O autor utiliza redes SOM em seu método, entretanto, somente o método é apresentado. O autor também não resolve o problema de extração automática de grupos, para isso ele utiliza um especialista da área, e também não descreve como manipula o fluxo de dados.

No capítulo seguinte iremos descrever a abordagem tomada para a realização do agrupamento em fluxo de dados. Assim como em [Longo and Barrett, 2009] utilizamos o algoritmo SOM para a realização do agrupamento. O algoritmo SOM é versátil, pode realizar o agrupamento de vários tipos de grupos, sendo necessário especificar poucos parâmetros. O maior problema, a extração automática dos grupos, foi tratada através do algoritmo GSOM, descrito no capítulo 4.

2.9 Conclusões

Neste capítulo, apresentamos os principais conceitos acerca de agrupamento de dados, algoritmos e formas de validação de grupos. O algoritmo de agrupamento utilizado neste trabalho será descrito no capítulo 4.

Reunimos informações necessárias para o leitor compreender o fluxo de dados e técnicas para seu tratamento. O método proposto utiliza algumas das técnicas apresentadas neste capítulo, como, filas, sumário, amostragem.

O próximo capítulo irá apresentar o método proposto para realizar o agrupamento em fluxo de dados. O método utiliza os conceitos de fluxo e agrupamento estudados até o momento.

Capítulo 3

Um Método para Agrupamento em Fluxo de Dados

A coleta de informações em uma aplicação interativa, utilizada simultaneamente por várias pessoas, ao ser enviada a um servidor para processamento gera um fluxo de dados. O tratamento do fluxo de dados, assim como a aplicação de um algoritmo de agrupamento fazem parte do método proposto.

Neste capítulo será apresentado o método para o agrupamento em fluxo de dados, utilizando o algoritmo de agrupamento SOM com extração automática de grupos da rede treinada.

3.1 Visão Geral

Tendo em vista o contínuo fluxo de dados e as limitações de recursos, *i.e.*, capacidade de processamento e armazenamento, propomos o seguinte método para agrupamento em fluxo de dados contínuo, baseado em 3 etapas: *pré-processamento*, *processamento* e *pós-processamento*. As principais tarefas (figura 3.1) a serem realizadas por este método são: tratamento do fluxo de dados, leitura dos dados, realização do agrupamento, e apresentação dos resultados.

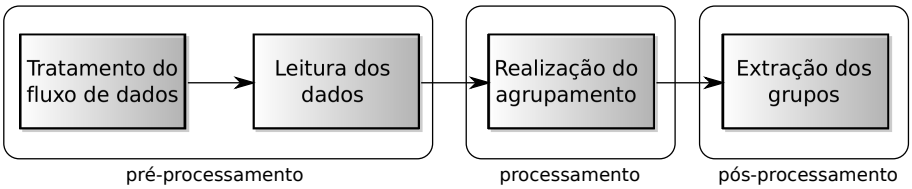


Figura 3.1: Visão geral do método

A etapa de pré-processamento envolve o *tratamento de fluxo de dados*, e a *leitura dos dados*. A primeira etapa utiliza uma fila (seção 2.7.2) *fifo* para organizar a chegada dos dados. O uso de uma fila auxilia no tratamento do fluxo de dados, podendo ser realizado amostragens ou agregação dos dados (seção 2.7) para evitar o derramamento de carga. Os dados são então lidos através de uma janela de leitura (seção 2.7.1) e enviados para o algoritmo de agrupamento.

Na etapa de processamento, que envolve a *realização do agrupamento*, os dados obtidos através do procedimento anterior são submetidos a um processo de agrupamento. O processo de agrupamento utiliza o algoritmo SOM, uma rede neural auto-organizável, para desco-

brir grupos nos dados recebidos. O algoritmo SOM foi escolhido pois permite o agrupamento de diferentes tipos de grupos: bem separados, baseado em centroide, de densidade e formato diferentes, conforme seção 2.2. O algoritmo não necessita de um especialista para extrair os grupos, para tanto, ele utiliza uma técnica de extração automática dos grupos que será apresentada no capítulo 4.

Na etapa de pós-processamento, que envolve a *apresentação dos resultados*, serão apresentados resultados obtidos através das técnicas de validação não-supervisionada de grupos (descrito na seção 2.5.1). O número de grupos encontrado e sua qualidade medida através do coeficiente de silhueta.

Um simulador foi construído para dar suporte a validação dos testes. O simulador será descrito no capítulo 5. Com ele é possível receber registros de aplicações iterativas através de sockets, configurar os parâmetros de controle da fila, do algoritmo SOM, e visualização dos resultados.

3.2 Detalhamento do Método

Nesta seção serão descritos em detalhes cada uma das etapas do método proposto.

3.2.1 Pré-Processamento

Na etapa de pré-processamento o tratamento do fluxo de dados é realizado através de uma fila *fifo*. Os clientes conectados, enviam seus registros (os vetores de características), cada um ao seu tempo e velocidade. Estas informações precisam aguardar em uma fila para serem agrupados, neste momento há a serialização dos registros, pois há somente uma fila.

A fila tem um tamanho t_f especificado a priori. Este valor pode ser modificado durante a execução, contudo, se o tamanho da fila diminuir a ponto de ficar menor do que o número de registros n_r contido, estes são perdidos. Portanto, se $t_f < n_r$ há o derramamento de carga. O derramamento de carga deve ser evitado, pois perde-se informação que deveria estar sendo agrupada.

Para evitar o derramamento de carga, o sistema deve estar estável. Diz-se que o sistema é estável quando a taxa de chegada é menor que a taxa de saída/serviço. Quando o sistema encontra-se instável é uma questão de tempo para que haja o *overflow*. Para saber o tempo restante, calcula-se o tempo de overflow $t_o = \frac{f-o}{s-e}$, sendo f o tamanho da fila, o o número de elementos na fila, e e s a taxa de chegada e a taxa de saída, respectivamente.

Tomemos como exemplo uma fila de tamanho $f = 100r$, com $o = 20r$ posições ocupadas, taxa de entrada $e = 5r/s$ e saída $s = 10r/s$. Como $e < s$, a fila está estável. O tempo restante até a fila encher é $t_o = \frac{f-o}{s-e}$, portanto, $100-20/10-5 = 16s$ - número positivo a fila não enche, pois fila encontra-se estável. Supondo que $e = 10r/s$ e $s = 5r/s$, a fila é instável, pois $e > s$, e o tempo restante é $100-20/5-10 = -16s$, ou seja, 16 segundos restantes.

Contamos com duas formas de controle: a *amostragem* e *agregação*. Estas são técnicas não destrutivas, ou seja, registros são perdidos, mas sua dimensionalidade é mantida. Obtemos, desta forma, uma representação próxima da real, ao invés de simplesmente não representá-las quando há o derramamento de carga.

Através da amostragem, selecionamos um subconjunto de indivíduos de uma população de forma que este subconjunto represente a população. Neste trabalho foi utilizado o

processo de amostragem aleatória simples, utilizando-se um erro amostral fixo de 5%. O cálculo para quantidade de elementos a serem aleatóriamente escolhidos é dado por: $\frac{N \times n_0}{N+n_0}$, sendo $n_0 = \frac{1}{E_0^2}$, E_0 o erro amostral tolerável (0,05 neste trabalho), e N o número de elementos da população (fila).

O processo de agregação, neste caso, calcula-se a média de dois vetores de registros utilizando-o no lugar dos dois registros. Assim diminui-se o tamanho da fila pela metade.

A normalização dos dados não é possível pois não conhecemos os valores que ainda não chegaram na fila. Desta forma não se pode manter os dados em um mesmo intervalo. Pode-se assumir um intervalo, ou simplesmente enviá-los já normalizados quando a base de dados já é conhecida. Outra alternativa seria anotar os valores maiores e menores conforme eles chegam. Portanto, a normalização dos dados pode ser feita a priori, ou sob demanda. Quando a normalização é feita sob demanda, encontra-se cada valor máximo e mínimo das características de cada um dos registros contidos na fila. Por já conhecermos a base realizamos a normalização dos dados antes do início do fluxo.

A janela deslizante tem um tamanho t_j fixo menor que o tamanho da fila. A janela deslizante sempre "lê" os dados da frente da fila, e sempre que a janela estiver cheia $(t_j \geq t_f)$. O algoritmo SOM irá utilizar os registros contidos na janela temporal para realizar os agrupamentos. Assim que os agrupamentos forem realizados, a janela temporal lê mais t_j registros. Cada leitura representa um bloco de informação. O bloco de informação contém $w = m \times v$ elementos, sendo m o número de registros e v o número de características de cada registro (conforme descrito em 2.7.1).

3.2.2 Processamento

Um algoritmo de agrupamento é executado sob os dados retirados da janela de leitura - bloco de informação. O resultado do algoritmo de agrupamento irá gerar os grupos referentes a este bloco. Ou seja, o resultado representa somente o agrupamento mais recente do fluxo de dados.

Neste método propomos a utilização do algoritmo SOM para realizar o agrupamento. SOM é utilizado para agrupar dados. O treinamento não supervisionado de uma rede neural ocorre quando se apresenta vetores de características sem os seus respectivos rótulos, realizado o agrupamento dos dados (conforme descrito em 2.4.3).

Na etapa de treinamento da rede, apresenta-se os registros obtidos através da última leitura realizada pela janela de leitura ao algoritmo SOM. Este algoritmo consegue representar registros de quaisquer dimensões em uma rede de duas dimensões. Após o treinamento ser completado esta rede estará organizada de forma que registros similares estejam associados a vetores mais próximos topologicamente. Neste mapa de neurônios organizados, os neurônios vizinhos respondem a entradas semelhantes, criando áreas que representam registros semelhantes.

Os grupos são descobertos quando observamos a proximidade dos neurônios na grade. Neurônios próximos representam grupos. Mas não há a separação exata, pois cada neurônio está ligado aos seus vizinhos. A tarefa de extração de grupos necessita de um especialista na área ou um algoritmo específico. Neste trabalho implementamos um algoritmo para extração automática dos grupos (descrito no capítulo 4).

A partir da extração dos grupos, obtemos o *modelo,* que representa o conhecimento adquirido deste procedimento de agrupamento. Neste ponto o modelo já pode ser interpretado, e utilizado para classificação de novos registros que chegam ao sistema.

Na figura 3.2.2 podemos visualizar a sequência de tarefas. Os vetores de características chegam a partir de sockets, por onde são serializados em uma fila. Quando o número de vetores é o suficiente, a janela de leitura remove-os da fila e envia para o algoritmo de agrupamento, liberando o espaço na fila para mais vetores chegarem. O algoritmo SOM realiza o agrupamento e como resultado obtemos a rede treinada.

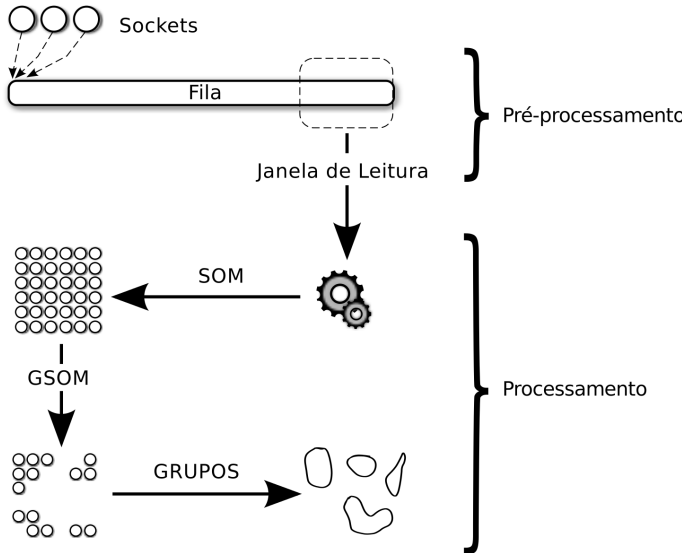


Figura 3.2: Conjunto de tarefas do método

3.2.3 Pós-processamento

O pós-processamento envolve a extração dos grupos na rede treinada SOM, assom como a visualização de um relatório de qualidade dos grupos.

Para extrair os grupos da rede treinada, utilizamos o algoritmo GSOM. O algoritmo irá encontrar os grupos utilizando a medida de silhueta (detalhado no próximo capítulo). O algoritmo GSOM separa as ligações dos neurônios treinados de uma rede SOM utilizando a medida de silhueta, criando assim áreas que representam os grupos.

Para demonstrar graficamente a qualidade do agrupamento obtido pelo agrupamento mais recente, utilizamos a visualização gráfica através de matriz de similaridade (descrito na seção 2.5.1). A matriz de similaridade permite demonstrar visualmente a relação de semelhança entre os elementos de um mesmo grupo. Elementos de um mesmo grupo aparecem mais próximos em tons de cinza mais escuro, conforme figura 2.5.

Outra visualização gráfica é a representação dos registros em um espaço euclidiano onde os registros de mesmo grupo possuem mesma cor. Obviamente, só é possível representar registros de até três dimensões. Neste trabalho plotamos gráficos de 2 dimensões para poder verificar graficamente o resultado do agrupamento.

Para cada grupo encontrado durante o processo de agrupamento, etapa de *processamento*, calcula-se uma medida de qualidade. Neste trabalho utilizamos a medida de coesão e silhueta (seção 2.5.1, subseção 2.5.1).

As medidas de coesão e silhueta para cada grupo individual e para o agrupamento serão disponibilizadas para análise do agrupamento. Esta medida é usada para plotar o gráfico da qualidade do agrupamento versus o *timeframe* atual. A medida de silhueta também é a medida utilizada para decidir a quantidade de grupos durante a fase de refinamento da rede. Neste caso, escolhe-se o agrupamento com melhor silhueta.

Portanto, podemos mensurar a qualidade do agrupamento graficamente através da matriz de similaridade e a visualização dos grupos, assim como numericamente através da medida de coesão.

3.3 Conclusões

O método proposto neste documento leva em consideração a qualidade do agrupamento dado uma quantidade estabelecida de memória (tamanho da fila). O resultado dos experimentos deverão nos fornecer a qualidade dos agrupamentos.

Este método utiliza somente o último agrupamento realizado. Agrupamentos anteriores são armazenados para analizar o histórico de grupos e qualidade. Em trabalhos futuros pretende-se utilizar os pesos dos neurônios descobertos no agrupamento anterior como ponto de partida para a nova aprendizagem, afim de aproveitar o conhecimento já adquirido.

No capítulo seguinte apresentaremos em detalhes o algoritmo de agrupamento SOM. Seu funcionamento, parâmetros, e dificuldades encontradas. A maior dificuldade é a decisão sobre o número de grupos correto existente no agrupamento. Resolvemos através do algoritmo GSOM, uma extensão do algoritmo original. Este serve para realizar a extração automática dos grupos na rede treinada SOM.

Capítulo 4

Extração Automática de Grupos no Algoritmo SOM

Neste capítulo será apresentado o algoritmo *Self-organizing Map*, utilizado para aprendizagem não supervisionada. Na sequência, será descrito o algoritmo GSOM, o qual foi criado para a extração automática dos grupos após o treinamento de uma rede SOM. A extração automática dos grupos é fundamental para utilização do algoritmo SOM em fluxo de dados, uma vez que não é possível a atuação de um especialista para realizar esta tarefa.

4.1 O algoritmo SOM

A rede SOM *Self-organizing Map* é utilizada para aprendizagem não supervisionada. A aprendizagem não supervisionada em uma rede neural ocorre quando se apresenta vetores de características sem os seus respectivos rótulos, realizado o agrupamento dos dados.

A rede SOM de *Kohonen*, é um tipo de Rede Neural auto-organizável para realização de agrupamento. O algoritmo realiza uma redução de dimensionalidade dos vetores fornecidos para treino, usualmente em um espaço uni ou bi-dimensional (aceitando também *n* dimensões) [Jain, 2010] [Fausett, 1994].

Durante o processo de auto-organização, escolhe-se como vencedor o neurônio cujo vetor de pesos for mais próximo ao vetor de entrada. O neurônio vencedor e seus vizinhos têm os pesos atualizados.

Neste trabalho utilizamos um espaço bi-dimensional para representar o agrupamento. A arquitetura da rede utilizada pode ser vista na figura 4.1. Os neurônios da grade estão conectados entre si. Estas conexões representam a distância ou força de ligação entre eles, e é calculada através da distância euclidiana. Os neurônios estão em posições fixas no espaço de forma que a distância não influencie na topologia. A camada de entrada não está fisicamente ligada aos neurônios da grade, sua ligação, em tracejado, simboliza a distância euclidiana até cada um dos neurônios. Esta é utilizada durante o treinamento, onde apresenta-se registros da base de treino à rede, ou após o treinamento para realizar uma classificação e descobrir qual o neurônio mais semelhante.

O processo de redução de dimensionalidade dos vetores organiza as instâncias da base de treino de forma que os mais semelhantes fiquem mais próximos na grade. No fim do treinamento da rede, obtém-se os vetores de pesos dos neurônios mais similares mais próximos

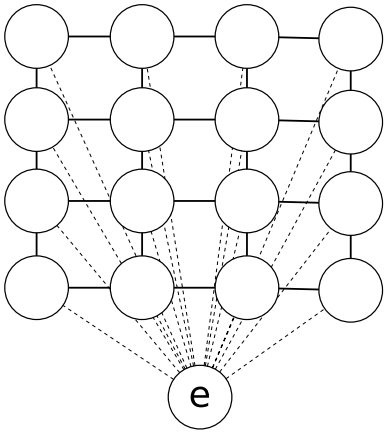


Figura 4.1: Arquitetura SOM com camada de saída de 4x4 neurônios conectados à camada de entrada *e*.

entre si, formando um agrupamento. Ao apresentar um novo registro à rede, este é associado ao grupo de neurônios cujo vetor de pesos é mais semelhante.

O processo de extração dos grupos da rede deve ser realizado manualmente, por exemplo, observando-se a u-matrix e escolhendo os neurônios que fazem parte de um mesmo agrupamento baseado em sua vizinhança. A u-matrix permite visualizar a relação (semelhança) de um neurônio com seus vizinhos (figura 4.4).

Processos automáticos para extração dos grupos existem, tal como: o uso de regras *fuzzy* para interpretação dos grupos [Drobics et al., 2000]; utilização do algoritmo Clusot modificado [Brugger et al., 2008]; o uso de algoritmos *PSO* (*Particle Swarm Optimization*) sobre a rede treinada SOM [Sharma and Omlin, 2006]; e mais recentemente [Cabanes and Bennani, 2010], que utiliza uma abordagem baseada em densidade, e [Ghaseminezhad and Karami, 2011] que utiliza uma técnica semelhante de redução de arestas entre os neurônios. Como já estava trabalhando em outra solução para a extração automatica de grupos, resolvemos dar continuidade para poder comparar a performance do novo algoritmo com os já existentes. Para realizar a extração automática dos grupos, desenvolvemos uma técnica através do *refinamento iterativo da rede*, o qual será descrito neste capítulo, na seção 4.3.

4.2 Descrição do Algoritmo SOM

A rede SOM, neste caso, é formada por uma grade de *NxM* neurônios (ou nós). Cada neurônio *i* possui um vetor de pesos W_i de *n* dimensões, $W_i = \{w_1, \ldots, w_n\}$. A camada de entrada possui um neurônio de dimensionalidade igual a dos elementos conjunto de treinamento V_t. O conjunto de treinamento V_t possui *x* registros, sendo $V_t = \{V_1, V_2, V_3, \ldots, V_x\}$, e cada registro possui dimensionalidade *n*, $V_i = \{v_1, \ldots, v_n\}$ ($V_i \in V_t \wedge |W_i| = |V_i|$).

Para cada registro de entrada apresentado, calcula-se a distância euclidiana entre o registro e os nós da rede. Escolhe-se como vencedor o neurônio mais próximo ao vetor de entrada. O neurônio vencedor e seus vizinhos - dentro de um raio *R* - têm os pesos atualizados de acordo com o fator de ajuste (*learning rate*) α. O algoritmo de treinamento da rede é apresentado no algoritmo 1.

O algoritmo 1 requer um conjunto V_t com os vetores de treinamento da rede, a taxa de aprendizagem α inicial e o raio R inicial. O algoritmo irá realizar a laço principal por E (*epochs*) vezes.

Algoritmo 1 SOM. Adaptado de [Fausett, 1994]

Require: (α, V_t, E, $Grid_{x,y}$) // α: learning rate, V_t: conjunto de treinamento, E: número de épocas, and $Grid_{x,y}$: tamanho do mapa

1: Inicializa aleatóriamente os pesos em W
2: Ajusta a taxa de aprendizagem
3: Ajusta a vizinhança
4: R = inicializaçãoDoRaio($Grid_x$, $Grid_y$)
5: $i \leftarrow 0$
6: **repeat**
7: **for each** V_i in V_t **do**
8: **for each** v_n in V_i **do**
9: $D(W_i) \leftarrow \sum_n (w_n - v_n)^2$
10: **end for**
11: $W_{BMU} \leftarrow encontrarBMU(V_i)$
12: **for each** W_i vizinha de W_{BMU} dado o raio R **do**
13: $W_i \leftarrow W_i(antigo) + \alpha[V_i - W_i(antigo)]$
14: **end for**
15: **end for**
16: atualizaAprendizagem(i, E, α)
17: atualizaRaio(i, E)
18: $i \leftarrow i + 1$
19: **until** E $\leq$ i

Os pesos W são inicializados em valores $0 < W < 1$. A base de treino V_t deve ter seus valores ajustados neste mesmo intervalo. A taxa de aprendizagem é inicialmente ajustada para 0.1 por padrão. O raio é inicializado com metade da distância entre os dois extremos da grade conforme algoritmo 2.

Algoritmo 2 Inicialização do raio

Require: (x, y) // x e y representam o tamanho da grade

1: $raioInicial \leftarrow \sqrt{x^2 + y^2}$
2: $raioInicial \leftarrow raioInicial/2$

Para cada registro da base de treinamento, encontra-se o neurônio com a menor distância euclidiana. Isso é realizado através das linhas 5 a 9 do algoritmo 1. O neurônio mais próximo, também chamado de BMU (*Best Matching Unit*), pode ser determinado de acordo com o algoritmo 3.

Algoritmo 3 Encontrar BMU

Require: (V_i) // um vetor da base de treino
1: Neurônio bmu ← nulo
2: distância ← ∞
3: **for each** Neurônio W_i **in** gradeDeNeurônrio **do**
4: tmp ← distânciaEuclidiana(W_i, V_i)
5: **if** tmp ≤ distância **then**
6: distância ← tmp
7: bmu ← W_i
8: **end if**
9: **end for**
10: retorna bmu

Após encontrar o BMU, deve-se encontrar todos os vizinhos. Os vizinhos são os nós com proximidade menor ou igual o raio atual. O raio é iniciado com um valor e decresce exponencialmente de acordo com a equação 4.1 (figura 4.2),

$$R_n(t) = R_i \exp\left(-\frac{t}{\lambda}\right) \tag{4.1}$$

sendo R_i o raio inicial, t a iteração atual, e λ o número de iterações total (épocas). Desta forma o raio decresce exponencialmente em função da iteração (ver figura 4.2). Pode-se, ainda, substituir o λ da equação 4.1 por $\frac{\lambda}{\log R_i}$ e obter um decremento alternativo (ver figura 4.2).

Todos os vizinhos dos BMUs devem ter seus pesos ajustados conforme equação 4.2.

$$W_i \leftarrow W_i(antigo) + \alpha[V_i - W_i(antigo)] \tag{4.2}$$

sendo W_i o vetor de pesos do neurônio atual, e α a taxa de aprendizagem. O novo vetor de pesos para este neurônio é igual ao vetor de pesos antigo mais uma fração (α) da diferença entre o vetor antigo e o vetor apresentado à rede.

Após realizar o ajuste dos pesos da vizinhança, é preciso reajustar os valores do raio e taxa de aprendizagem (correspondente as linhas 15 e 16 do algoritmo 1). De forma similar ao raio, a taxa de aprendizagem decresce de forma exponencial, conforme a equação 4.3:

$$L(t) = L_0 \exp\left(-\frac{t}{\lambda}\right) \tag{4.3}$$

sendo L_0 a taxa de aprendizagem inicial, t a iteração atual e λ o número de iterações total (épocas). Assim como o raio, a taxa de aprendizagem decresce de forma exponencial em função da iteração, conforme figura 4.3.

O raio é ajustado pelo algoritmo 4 que recebe um inteiro representando a iteração atual e retorna um número com o novo raio.

Algoritmo 4 Ajuste do raio

Require: (i, ep) // iteração atual e total de épocas
1: retorna $R \times e^{-1(i/ep)}$

A taxa de aprendizagem é ajustada pelo algoritmo 5 que recebe um inteiro representando a iteração atual e retorna um número com a nova taxa.

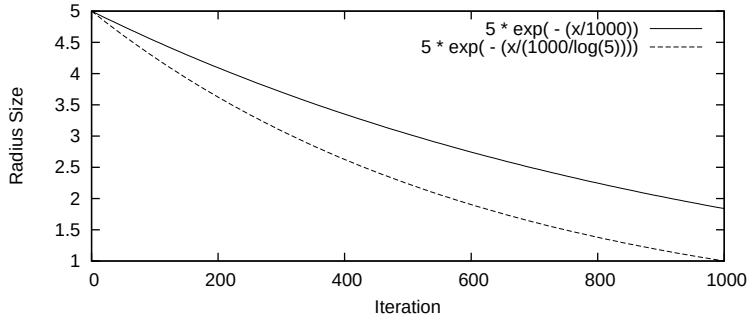


Figura 4.2: Exemplo de decremento do raio em função da iteração. Raio inicial: 5. Épocas: 1000

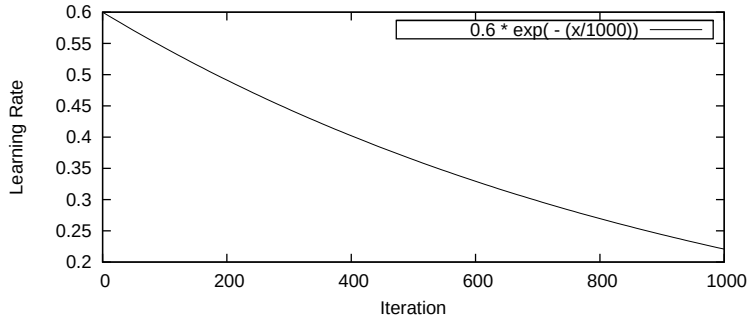


Figura 4.3: Exemplo de decremento da taxa de aprendizagem em função da iteração. Taxa inicial: 0.6. Épocas: 1000

Algoritmo 5 Ajuste de taxa de aprendizagem

Require: (i, ep, α) // iteração atual, total de épocas e taxa de aprendizagem inicial

1: retorna $\alpha \times e^{-1(i/ep)}$

A condição de parada do algoritmo 1 corresponde a quantidade de épocas estabelecido no código. O algoritmo deve rodar o loop principal pelo número de épocas correspondente. Após o treinamento, os vetores de pesos dos neurônios devem estar todos ajustados. Esta condição de parada pode gerar um *overfitting*; uma forma de evitá-lo é utilizar um conjunto de validação. Entretanto, não dispomos de conjunto de validação, visto que os dados utilizados são provenientes de um fluxo de dados.

4.3 Extração automática de grupos

Após a condição de parada ter sido atingida, a rede neural está devidamente treinada e já apresenta os vetores de pesos de seus neurônios devidamente ajustados para a base de treino utilizada. Através da matriz-u é possível verificar visualmente o estado dos neurônios (sua relação com a vizinhança), e na maioria dos casos, já é possível ver a separação dos grupos conforme figura 4.4.

Os neurônios escuros representam áreas de separação entre grupos. A cor de cada neurônio da matriz-u é ajustado de acordo com a soma das distâncias do próprio vetor de características até seus vizinhos, o p_u (peso-u), o qual pode ser calculado conforme equação 4.4

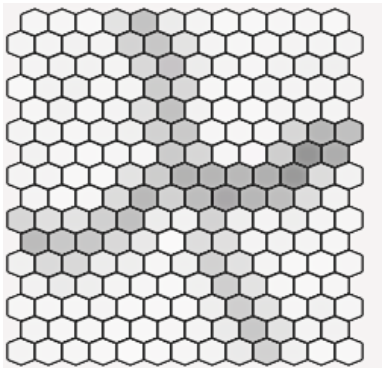


Figura 4.4: Exemplo de visualização da rede matrix-u.

$$p_u(n) = \sum_{m \in NN(n)} d(w(n) - w(m)) \tag{4.4}$$

, sendo $d(x,y)$ uma medida de distância, n um neurônio no mapa, $NN(n)$ a vizinhança de n no mapa, e $w(n)$ o vetor de pesos associados ao neurônio n [Ultsch, 2003, Ultsch and Mörchen, 2006].

Outra forma de visualização é a matriz de densidade. Após o treino, verifica-se a proximidade (distância euclidiana) de cada registro da base de treino até cada um dos neurônios da rede; o neurônio mais próximo ganha um ponto de acerto. Com isso cada neurônio apresenta uma quantidade de acerto sendo possível ter uma noção de densidade, conforme figura 4.5.

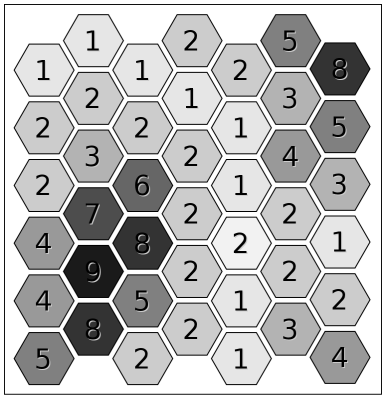


Figura 4.5: Exemplo de matriz de densidade

As regiões mais densas representam grupos de neurônios cujo vetor de características é similar para com seus vizinhos. Áreas menos densas representam regiões de separação entre grupos. Tendo em mente essa noção de densidade da rede, podemos aplicar o uso de *refinamento iterativo da rede*, conceito desenvolvido para extração automática dos grupos. Este conceito pode ser aplicado à uma rede treinada através do algoritmo GSOM.

4.3.1 O Algoritmo GSOM

A idéia principal do algoritmo GSOM é reduzir, gradualmente, o número de neurônios ativos na rede. Entende-se como neurônio ativo aquele que é usado para classificar uma

nova instância. Ao desativar os neurônios menos significativos, os grupos tendem a ficar mais definidos, pois os registros que antes eram associados à neurônios pertencentes a uma região de fronteira, agora são associados à um neurônio mais central (a figura 4.6 ilustra as iterações do algoritmo GSOM).

A cada passo do algoritmo os neurônios com menor número de registros associados são desativados. Isso é feito até que todos os neurônios da rede estejam desativados. Por exemplo, da iteração 1 (figura 4.6(a)) para a iteração 2 (figura 4.6(b)), são removidos os neurônios com acerto igual a 1. Os registros que antes eram associados a estes neurônios são naturalmente associados aos vizinhos mais próximos, devido ao resultado da organização do mapa pelo algoritmo SOM. Obviamente, se todos os neurônios forem desativados, não é possível definir grupos. Portanto, é utilizado o número de neurônios que maximiza uma medida de validação de grupo (seção 2.5.1).

Nota-se que sempre a primeira e última iteração possuem 1 grupo. No primeiro caso todos os neurônios estão interligados através de seus vizinhos, e no último há somente um neurônio (na maioria dos casos). Durante as iterações intermediárias o número de grupos varia conforme os neurônios são desativados.

A cada iteração a densidade de determinadas área aumenta, e forma-se uma separação nítida entre os neurônios. O número de grupos do agrupamento é o mesmo do número de grupos de neurônios formado. Por exemplo, na iteração 11 (figura 4.6(k)) temos a formação de 3 grupos. A cada iteração calcula-se uma medida de validação para o agrupamento resultante. O agrupamento com melhor resultado é escolhido como agrupamento resultante, e assim determina-se o número de grupos.

O algoritmo 6 requer como parâmetros iniciais a rede treinada W e a base de treino V, e é parte do algoritmo principal descrito em 7.

Algoritmo 6 GSOM - Iteração

```
Require: (W, V )   // rede treinada e a base de treino
 1: acertoMínimo ← ∞
 2: for each Neurônio W_i in gradeDeNeurônio do
 3:     W_i.acertos ← 0
 4: end for
 5: for each Registro V_i in baseDeTreino do
 6:     Neurônio bmu ← encontraBMU(V_i)
 7:     if bmu ≠ nulo then
 8:         bmu.acerto ← bmu.acerto + 1
 9:     end if
10: end for
11: for each Neurônio W_i in gradeDeNeurônio do
12:     if W_i ativo then
13:         if W_i.númeroDeAcertos < acertoMínimo then
14:             acertoMínimo ← W_i.númeroDeAcertos
15:         end if
16:     end if
17: end for
18: for each Neurônio W_i in gradeDeNeurônio do
19:     if W_i.númeroDeAcertos ≤ acertoMinimo then
20:         desativa(W_i)
21:     end if
22: end for
```

O algoritmo 6 diz respeito a uma iteração do algoritmo. Primeiro, é inicializado o número de acertos de cada neurônio (zero acertos), conforme linhas 1 a 4. Em seguida nas linhas 5 a 10, para cada registro da base de treino, encontramos seu BMU (algoritmo 3) e incrementamos o contador de acertos correspondente. Nas linhas 11 a 17 encontra-se o menor número de acertos. As linhas 18 a 22, desativa-se os neurônios que tiveram o número de acertos igual ao menor número de acertos.

Algoritmo 7 GSOM

```
Require: (W )   // rede treinada
 1: númeroDeGrupos ← 1
 2: listaDeAgrupamentos ← 0
 3: repeat
 4:     iteração(W)
 5:     Agrupamento agrupamento ← extraiGrupos(W)
 6:     númeroDeGrupos ← agrupamento.numeroDeGrupos
 7:     if númeroDeGrupos > 0 then
 8:         listaDeAgrupamentos.adiciona(agrupamento)
 9:     end if
10: until númeroDeGrupos ≤ 0
11: return melhorAgrupamento(listaDeAgrupamentos)
```

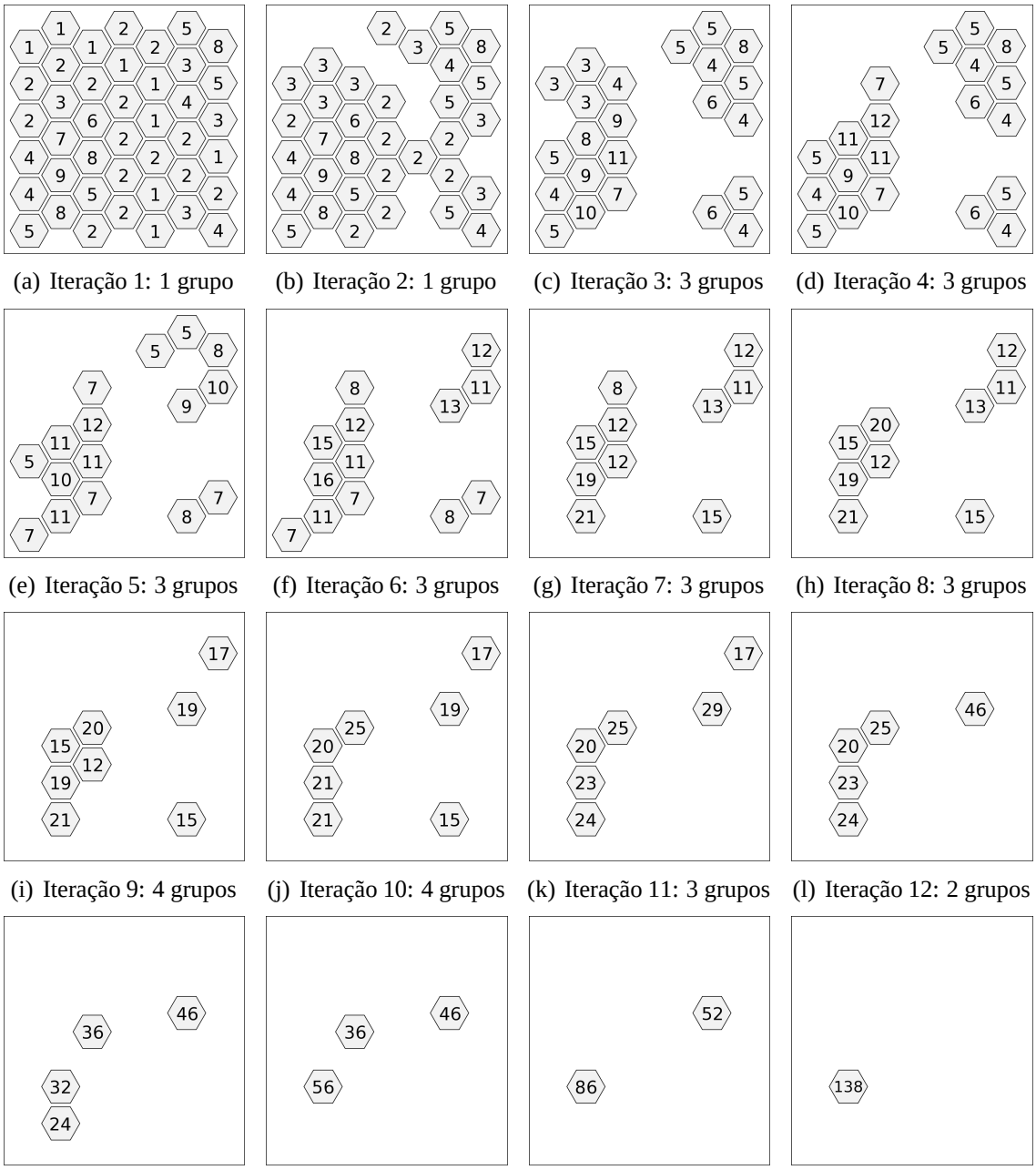
O algoritmo 7 executa as iterações do algoritmo 6 até que o número de grupos encontrados seja zero. O número de grupos é zero quando todos os neurônios são desativados. Uma

iteração é realizada na linha 4, e o agrupamento resultante é extraído na linha 5. A linha 6 calcula o número de grupos encontrado no agrupamento atual. A linha 8 adiciona o agrupamento a uma lista de agrupamentos. Esta lista serve para poder escolher o melhor agrupamento dentre todos os calculados durante as iterações. O melhor agrupamento é escolhido conforme algum cálculo de validação de grupo.

4.4 Conclusões

O algoritmo GSOM funciona para qualquer rede treinada utilizando o algoritmo SOM. O tempo de execução é proporcional ao tamanho da rede e o número de elementos da base de treino. Entretanto, o algoritmo pode ser paralelizado, tornando-o mais eficiente conforme o número de processadores disponíveis. O cálculo do BMU (best maching unit) e a extração de grupos de neurônios da rede, são métodos que podem ser paralelizados.

Juntos, os algoritmos SOM e GSOM, permitem realizar agrupamentos em diversas situações, com grupos de diferentes tipos, e, realizando a extração automática dos grupos. A extração automática depende de uma medida de validação de grupo, a qual deve ser maximizada. Os resultados dos testes utilizando a medida de coeficiente de silhueta estão descritas do capítulo 6.



(a) Iteração 1: 1 grupo (b) Iteração 2: 1 grupo (c) Iteração 3: 3 grupos (d) Iteração 4: 3 grupos

(e) Iteração 5: 3 grupos (f) Iteração 6: 3 grupos (g) Iteração 7: 3 grupos (h) Iteração 8: 3 grupos

(i) Iteração 9: 4 grupos (j) Iteração 10: 4 grupos (k) Iteração 11: 3 grupos (l) Iteração 12: 2 grupos

(m) Iteração 13: 3 grupos (n) Iteração 14: 3 grupos (o) Iteração 15: 2 grupos (p) Iteração 16: 1 grupo

Figura 4.6: Exemplo de iterações do algoritmo GSOM: representação visual da matriz de densidade durante as etapas de refinamento da rede.

Capítulo 5

O simulador para agrupamento em fluxo de dados

Neste capítulo será apresentado o funcionamento do simulador e suas telas de configurações.

5.1 O simulador

Os testes estão divididos em duas partes. O primeiro trata-se de um programa para testar o algoritmo GSOM, diferentes bases e parâmetros do algoritmo são testados para estimar o número correto de grupos. O segundo é o próprio simulador, este recebe o fluxo de dados e realiza o agrupamento utilizando o algoritmo apresentado.

Ambos os programas foram desenvolvidos em ambiente Unix, utilizando bibliotecas padrão C/C++ para que possam ser facilmente portáveis para outras plataformas. Única exceção é quanto a interface do simulador que estão implementados em Cocoa/Objective-C, entretanto, o modelo MVC utilizado permite a troca de *views* facilmente. Os testes do algoritmo GSOM foram realizados em um ambiente Linux, enquanto o simulador em ambiente MacOS 10.7.2.

O algoritmo GSOM pode ser utilizado através da linha de comando:

```
./gsomcpp <configFile.txt> <trainingBase.txt> <label colunm>
```

,sendo *gsomcpp* o executável, *configFile.txt* o arquivo de configuração da rede som, *trainingBase.txt* o arquivo com a base de treino, e *label colunm* o número da coluna que contém o rótulo/identificador da classe. O arquivo de configuração deve ser escrito conforme:

```
gridx=10
gridy=10
epochs=800
learning_rate=0.6
topology_type=squared
```

nesta exata ordem e substituindo-se os valores desejados para cada parâmetro. O parâmetro *topology_type* pode receber o valor *squared* ou *hexagonal*, a depender se a disposição dos neurônios na rede é *quadrado* ou *hexagonal*. Os resultados são mostrados na tela em forma de texto. Na versão gráfica do aplicativo é possível configurar os parâmetros dos algoritmos através das janelas de configurações, e o resultado é apresentado graficamente conforme figura 5.1.

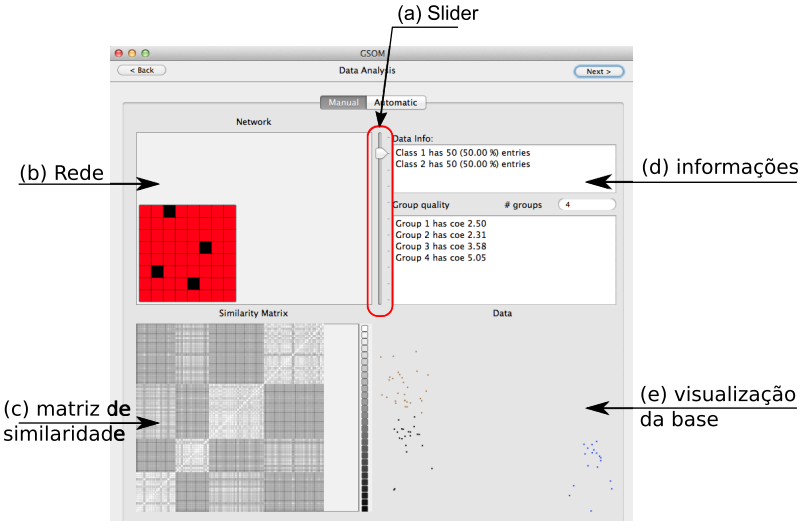


Figura 5.1: Janela de apresentação dos resultados do algoritmo GSOM

Nesta janela observamos o resultado da rede (figura 5.1.b) após a etapa de refinamento (nós vermelhos estão desativados), resultados da coesão do agrupamento via texto (figura 5.1.d), e um gráfico da matriz de similaridade (figura 5.1.c). É possível analisar todos os agrupamentos da etapa de refinamento movendo-se o *slider* (figura 5.1.a). A figura 5.1.e) apresenta a visualização gráfica da base de dados utilizada, sendo os grupos apresentados através de cores (somente para base de dados bi-dimensionais).

O simulador deve receber um fluxo de dados via *socket* de acordo com o seguinte protocolo:

```
<label>,<caracteristica 1>,<caracteristica 2>,<caracteristica n>^
```

sendo *label* o rótulo identificador da classe, seguido das características separadas por "," e finalizado com o delimitador de linha "∧" . Abaixo, um exemplo de transmissão de um vetor com duas características:

```
2,5.163508101851572,1.6351530311449274^
```

O servidor *socket* pode lidar com múltiplos clientes conectados simultaneamente. É importante que todos os vetores enviados pelos clientes tenham a mesma dimensionalidade. Os vetores recebidos por todos eles são inseridos na fila onde aguardam o agrupamento. Os vetores são descartados caso a fila esteja cheia.

A porta é o primeiro parâmetro a ser informado pelo usuário. A figura 5.2 mostra a tela inicial onde o usuário deve informar a porta. A figura 5.3 ilustra um exemplo de aplicação que faz a conexão com o simulador. Nesta aplicação, o usuário deve selecionar uma base de dados que contém vetores de características, e então conectar na porta e ip da o simulador que irá receber este fluxo. A velocidade é definida a partir de um *slider*, que pode ser alterado em tempo de execução.

Na tela de ajuste da fila (figura 5.4) é possível ajustar o tamanho da fila, tamanho da janela deslizante e verificar a quantidade atual de elementos. Pode-se também escolher se deseja realizar algum tipo de amostragem quando a fila atingir o tempo restante especificado para encher. Os tipos de amostragem realizados são: aleatória, estratificada, e média. A amostragem aleatória remove aleatoriamente o número de registros especificado. A amostragem estratificada

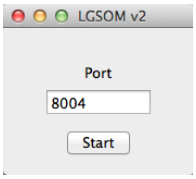


Figura 5.2: Escolha da porta de rede

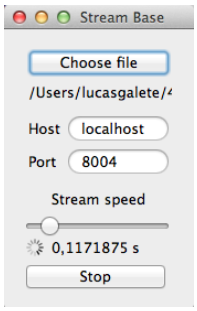


Figura 5.3: Tela de fluxo de uma base de dados

remove aleatoriamente e proporcionalmente o número de registros especificado de acordo com as classes já descobertas em agrupamentos passados. A amostragem média simplesmente une dois ou mais registros em um, utilizando a média destes como substituto.

Na tela de configuração do algoritmo SOM (figura 5.5)é possível alterar os parâmetros padrão do algoritmo, entretanto, as novas configurações só serão utilizadas no próximo agrupamento. Os parâmetros ajustáveis são: o tamanho da grade, o número de épocas, e a taxa de aprendizagem inicial. A grade pode ser quadrada ou hexagonal, a diferença é a disposição física dos neurônios da grade, o que influencia na vizinhança de cada um. Nesta tela pode-se observar o andamento do agrupamento atual na parte inferior da janela.

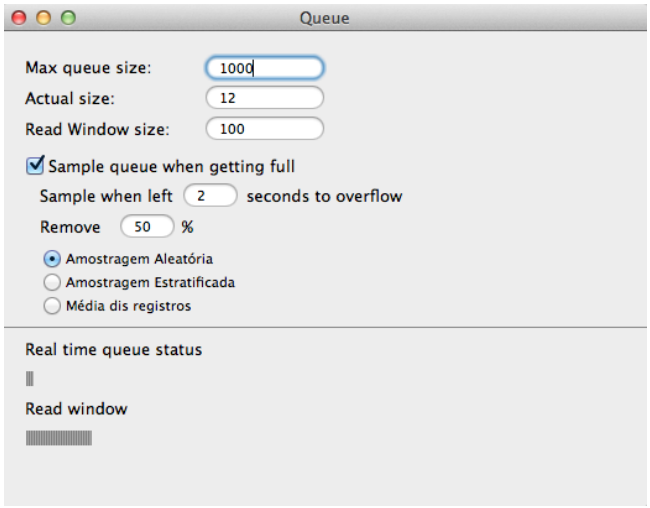


Figura 5.4: Tela de configuração da fila

As imagens 5.6(a) e 5.6(b) mostram o resultado do número de grupos e coesão, respectivamente, em tempo real, dos 10 últimos agrupamentos realizados. Conforme os registros recebidos pelo simulador mudam, a mudança pdoe ser observada também nos gráficos do resultado.

5.2 Conclusões

Neste capítulo apresentamos aspectos técnicos sobre o software desenvolvido para a realização dos experimentos. No capítulo seguinte, mostramos os resultados obtidos utilizando

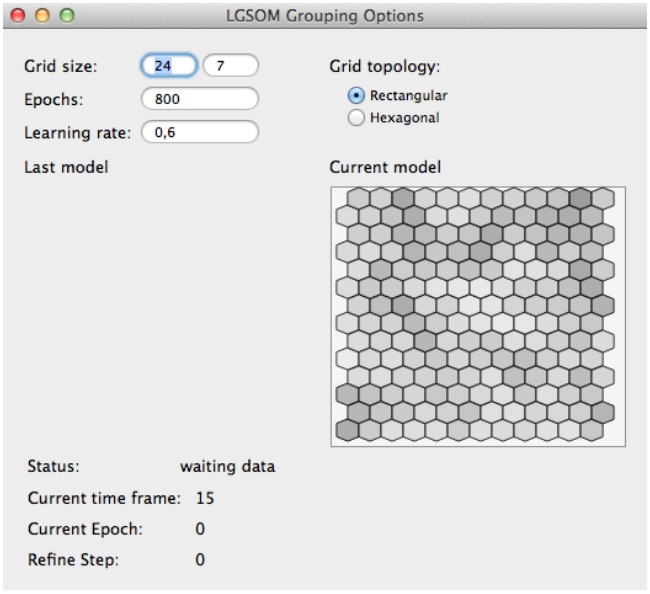
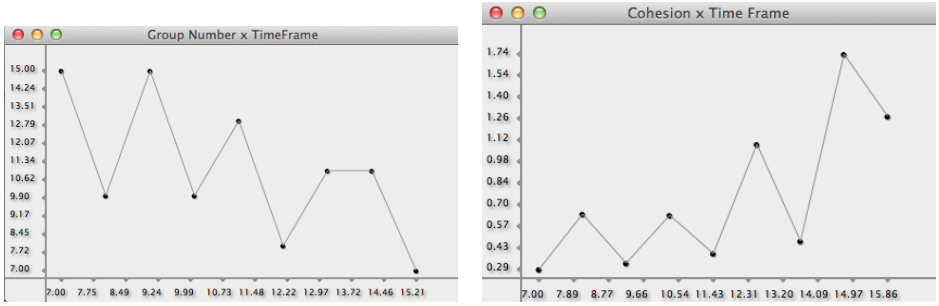


Figura 5.5: Tela de configuração do SOM



(a) Número de grupos na vertical vs *time frame* na horizontal

(b) Coesão do agrupamento na vertical vs *time frame* na horizontal

Figura 5.6: Gráficos gerados em tempo de execução.

o algoritmo GSOM via linha de comando. Ambos os softwares, via linha de comando ou modo gráfico, realizam a mesma tarefa.

O simulador, entretanto, só funciona em modo gráfico, pois necessita de interação com o operador para ajustar os parâmetros do algoritmo, fila, e visualização gráfica dos resultados.

Capítulo 6

Resultados e Discussões

Neste capítulo serão apresentados as bases utilizadas para testes do algoritmo GSOM, assim como os resultados obtidos e resultados esperados. Comparamos os resultados com outros dois algoritmos de agrupamento: KMeans e DBSCAN, assim como o resultado ideal (obtido a partir da coluna de rótulo das bases). A visualização gráfica do resultado, através das matrizes de similaridade, são apresentadas para todos os algoritmos, inclusive para o agrupamento ideal.

6.1 Bases de dados

Para realização dos experimentos utilizando o algoritmo GSOM, geramos cinco bases artificiais B_n contendo variáveis aleatórias bi-dimensionais com distribuição normal, sendo n o número identificador e número de grupos. A base de dados B_1 (figura 6.1(a)) contém 50 variáveis aleatórias bi-dimensionais, conforme:

$$x_j \sim N(\mu = [2,2], \sigma = [0.2, 0.2]), (j = 1, \ldots, 50) \tag{6.1}$$

; a base de dados B_2 (fig. 6.1(b)) contém 100 variáveis aleatórias bi-dimensionais, conforme:

$$x_j \sim N(\mu = [2,2], \sigma = [0.2, 0.2]), (j = 1, \ldots, 50)$$
$$x_j \sim N(\mu = [5,2], \sigma = [0.2, 0.2]), (j = 51, \ldots, 100)$$

; a base de dados B_3 (fig. 6.1(b)) contém 150 variáveis aleatórias bi-dimensionais, conforme:

$$x_j \sim N(\mu = [2,2], \sigma = [0.2, 0.2]), (j = 1, \ldots, 50)$$
$$x_j \sim N(\mu = [5,2], \sigma = [0.2, 0.2]), (j = 51, \ldots, 100)$$
$$x_j \sim N(\mu = [3.5,4], \sigma = [0.2, 0.2]), (j = 101, \ldots, 150)$$

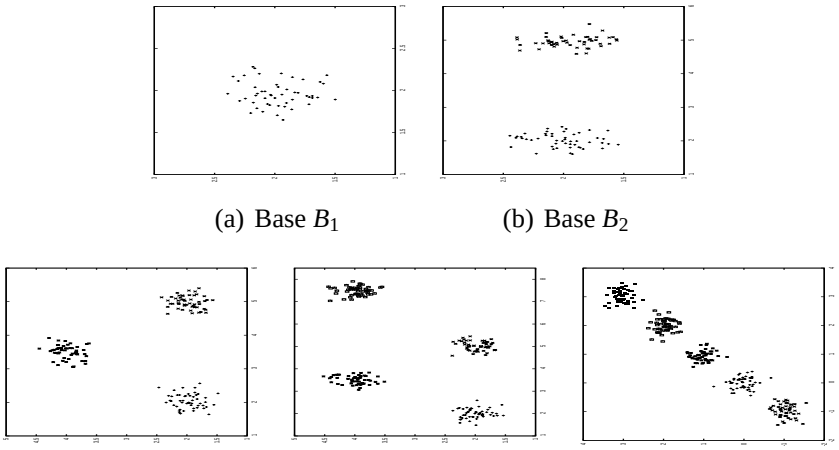
; a base de dados B_4 (fig. 6.1(b)) contém 200 variáveis aleatórias bi-dimensionais, conforme:

$$x_j \sim N(\mu = [2,2], \sigma = [0.2,0.2]), (j = 1,\ldots,50)$$
$$x_j \sim N(\mu = [5,2], \sigma = [0.2,0.2]), (j = 51,\ldots,100)$$
$$x_j \sim N(\mu = [3.5,4], \sigma = [0.2,0.2]), (j = 101,\ldots,150)$$
$$x_j \sim N(\mu = [7.5,4], \sigma = [0.2,0.2]), (j = 151,\ldots,200)$$

; a base de dados B_5 (fig. 6.1(b)) contém 250 variáveis aleatórias bi-dimensionais, conforme:

$$x_j \sim N(\mu = [-1,-1], \sigma = [0.2,0.2]), (j = 1,\ldots,50)$$
$$x_j \sim N(\mu = [0,0], \sigma = [0.2,0.2]), (j = 51,\ldots,100)$$
$$x_j \sim N(\mu = [1,1], \sigma = [0.2,0.2]), (j = 101,\ldots,150)$$
$$x_j \sim N(\mu = [2,2], \sigma = [0.2,0.2]), (j = 151,\ldots,200)$$
$$x_j \sim N(\mu = [3,3], \sigma = [0.2,0.2]), (j = 201,\ldots,250)$$

; sendo μ o ponto médio, e σ^2 a variância.



(a) Base B_1 (b) Base B_2

(c) Base B_3 (d) Base B_4 (e) Base B_5

Figura 6.1: Bases artificiais

Outras bases reais foram utilizadas para os testes: *Iris*[1], *Wine*[2], *Letter Recognition*[3], e *Dermatology*[4]. Estas bases podem ser encontradas no repositório de aprendizagem de máquina UCI[5]. A tabela 6.1 apresenta as características de cada base.

A distribuição das instâncias pelas classes de cada uma das bases pode ser observada na tabela 6.2.

[1]R.A. Fisher, Michael Marshall - 1988

[2]Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy., and Stefan Aeberhard - 1991

[3]David J. Slate. Odesta Corporation; 1890 Maple Ave; Suite 115; Evanston, IL 60201 - 1991

[4]Nilsel Ilter, M.D., Ph.D., H. Altay Guvenir, PhD. - 1998

[5]endereço web: http://archive.ics.uci.edu/ml/

Tabela 6.1: Bases de dados UCI

	Wine	Iris	Letter Rec.	Dermatology
Instâncias	178	150	20000	366
Atributos	13	4	16	34
Classes	3	3	26	6

Todas as bases de dados foram normalizadas para teste utilizando o algoritmo GSOM. Para a normalização, todos os valores de características dos vetores foram adaptados para o mesmo intervalo, neste caso, $[0,1]$. Os parâmetros do algoritmo SOM utilizados foram:

```
gridx=16
gridy=16
epochs=800
learning_rate=0.6
topology_type=squared
```

e

```
gridx=20
gridy=20
epochs=800
learning_rate=0.6
topology_type=squared
```

Esta última configuração foi utilizada somente em um teste com a base de dados *Letter Recognition*, pois o número de classes é maior, então criamos uma rede maior para comportar todas as classes com folga. Uma grade 20x20 gera 400 neurônios, cada um com 8 vizinhos (devido a topologia quadrada).

6.2 Resultados

Nós testamos o algoritmo GSOM com as bases descritas anteriormente para verificar se o número de grupos encontrado corresponde ao número ideal (calculado a partir dos rótulos). Os resultados estão descritos na tabela 6.3.

Na tabela 6.3, podemos verificar o número correto de grupos por base da dados, e o número de grupos encontrado nos testes. O algoritmo GSOM utilizou como parâmetros as configurações descritas na seção anterior. O algoritmo DBSCAN rodamos com duas configurações diferentes: $DBSCAN^2$ utilizou os parâmetros padrão encontrados no software Weka, sendo o número mínimo de pontos 6 e $\varepsilon = 0.9$; $DBSCAN^2$ o valor de ε é calculado automaticamente de acordo com [Ester et al., 1996]. O algoritmo KMEANS não consta na tabela pois ele não pode determinar o número de grupos - este é um parâmetro para o algoritmo.

A base B_1 apresentou um número de grupo muito acima do esperado, de 1 grupo, encontramos 48. Isso acontece pois os 48 grupos apresentam o coeficiente de silhueta superior do que 1 grupo. Portanto o algoritmo decide que este é o melhor resultado. Caso utilizássemos uma medida de validação diferente, que julgasse 1 grupo melhor que os 48 encontrados, o resultado seria diferente. Contudo, de forma geral, o coeficiente de silhueta apresenta ótimos resultados, como pode ser observado na tabela 6.3 e também graficamente através das matrizes de similaridade.

Tabela 6.2: Número de instâncias por classe

Classe	Wine	Iris	Letter Rec.	Dermatology
A/1	59	50	789	112
B/2	71	50	766	61
C/3	48	50	736	72
D/4	-	-	805	49
E/5	-	-	768	52
F/6	-	-	775	20
G	-	-	773	-
H	-	-	734	-
I	-	-	755	-
J	-	-	747	-
K	-	-	739	-
L	-	-	761	-
M	-	-	792	-
N	-	-	783	-
O	-	-	753	-
P	-	-	803	-
Q	-	-	783	-
R	-	-	758	-
S	-	-	748	-
T	-	-	796	-
U	-	-	813	-
V	-	-	764	-
W	-	-	752	-
X	-	-	787	-
Y	-	-	786	-
Z	-	-	734	-

O valor ideal foi calculado a partir dos rótulos disponíveis nas bases de dados utilizadas. Ou seja, apresenta o valor "correto"do agrupamento em questão. O mesmo se aplica para as matrizes de similaridade ideais.

A tabela 6.4 apresenta o valor do coeficiente de silhueta para os agrupamentos encontrados no algoritmo GSOM e demais. Conforme descrito na seção 2.5.1, o valor do coeficiente varia de -1 a 1, e o melhor valor é positivo, próximo a 0. O algoritmo GSOM busca exatamente este valor ótimo para resultado do cálculo do silhueta. Logo, o melhor agrupamento, de acordo com esse critério, é encontrado. Se observamos os valores encontrados e os valores ideais, podemos perceber que na maioria dos casos o valor é igual ou melhor.

Nos casos onde o valor de silhueta é diferente do ideal (*letter* e *dermatology*), não significa que o agrupamento estava errado. O algoritmo simplesmente encontrou outro agrupamento, dentro das diversas possibilidades que a base de dados possibilitava. Isso pode ser observado ao verificar os gráficos de matriz de similaridade resultantes e os ideais. As imagens 6.2, 6.3, 6.4, 6.5, 6.6, 6.7 e 6.8 comprovam que os agrupamentos encontrados são semelhantes, senão iguais, aos agrupamentos esperados, assim como seus coeficientes de silhueta. As imagens 6.10 e 6.9 apresentam matrizes de similaridade diferente das matrizes ideais, pois o

Tabela 6.3: Número de grupos encontrado

Base de dados	Núm. de grupos	SOM/GSOM	*DBSCAN*[1]	*DBSCAN*[2]
B_1	1	48	1	1
B_2	2	2	1	1
B_3	3	3	1	1
B_4	4	4	1	1
B_5	5	5	1	1
Wine	3	2	1	1
Iris	3	2	1	1
Letter	26	2	-	-
Dermatology	6	2	1	5

Tabela 6.4: Qualidade do agrupamento

Base de dados	Ideal[6]	SOM/GSOM	KMEANS	*DBSCAN*[1]	*DBSCAN*[2]
B_1	0	0.998715	0	0	0
B_2	0.733545	0.7335447538717975	0.7335447538717975	0	0
B_3	0.863608	0.8636083282136519	0.8636083282136519	0	0
B_4	0.893574	0.8935741999640276	0.8935741999640276	0	0
B_5	0.921725	0.9217247426308344	0.9217247426308344	0	0
Wine	0.387501	0.3885021143696126	0.3885021143696126	0	0
Iris	0.71955	0.7186887613243479	0.7186887613243479	0	0
Letter	0.336581	0.16396	-	-	-
Dermatology	0.664107	0.354617	0.6249519730979649	0	-

agrupamento encontrado foi outro, assim como são diferentes seus valores de coeficiente de silhueta.

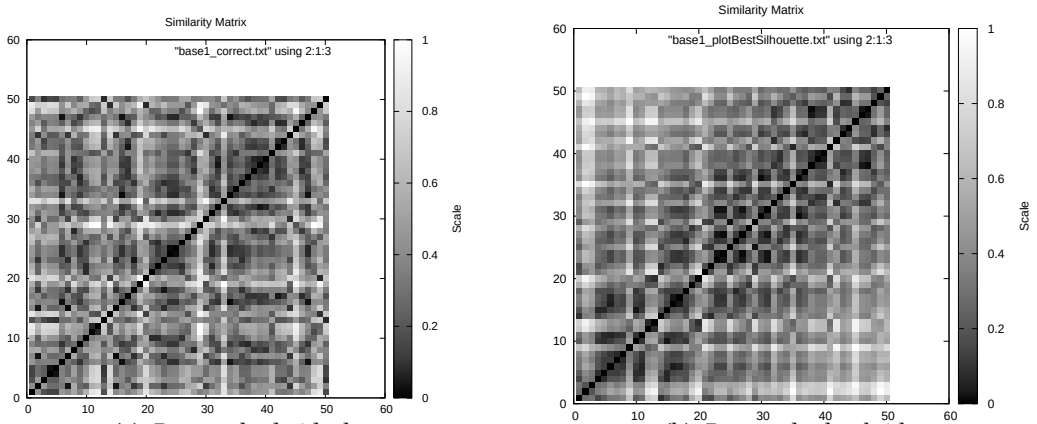
Em particular, os gráficos 6.9(a) e 6.9(b) são muito grandes devido a grande quantidade de registros em sua base de dados. Cada gráfico tem aproximadamente 500 MB e portanto tiveram sua resolução diminuída para caber no documento. A figura 6.11 apresenta uma aproximação dos dois primeiros grupos (próximos ao ponto 0,0) do gráfico 6.9(a).

6.3 Conclusões

Neste capítulo apresentamos os resultados obtidos através do algoritmo GSOM. Este algoritmo foi desenvolvido para realizar a extração automática dos grupos em uma rede treinada SOM.

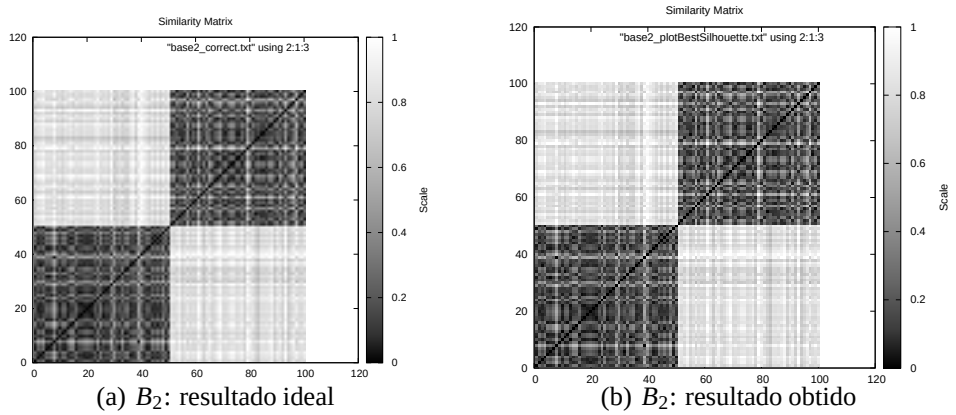
Através dos experimentos mostramos que é possível extrair os grupos automaticamente mantendo-se a qualidade do agrupamento de acordo com uma medida de validação de grupo, neste caso, o coeficiente de silhueta.

Comparando-se os resultados com o algoritmo KMEANS percebemos que os resultados são semelhantes, entretanto, o algoritmo KMEANS não define o número de grupos do agrupamento, este é um parâmetro que deve ser informado ao algoritmo. Em contrapartida, testamos o algoritmo DBSCAN com o modo automático para encontrar o parâmetro ε, este não se mostrou eficiente.



(a) B_1: resultado ideal (b) B_1: resultado obtido

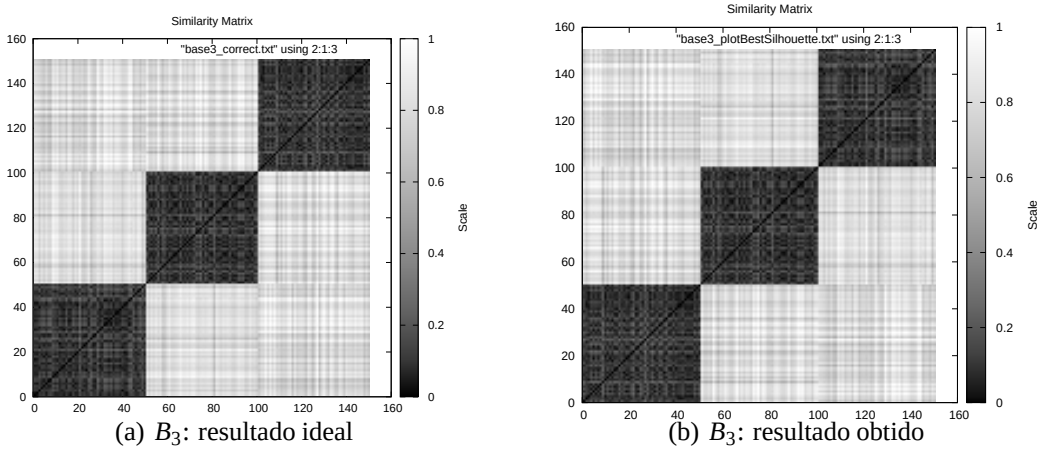
Figura 6.2: Matriz de similaridade dos resultados esperados e encontrados para a base B_1



(a) B_2: resultado ideal (b) B_2: resultado obtido

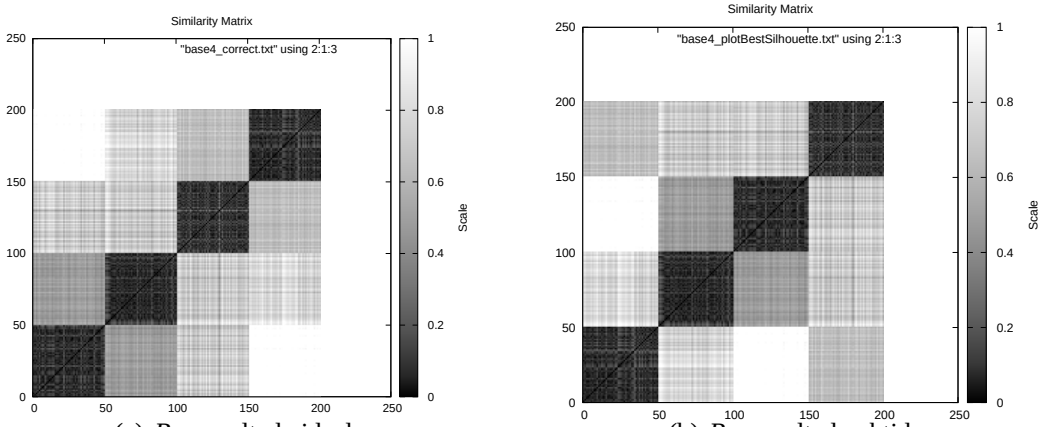
Figura 6.3: Matriz de similaridade dos resultados esperados e encontrados para a base B_2

Há diversos outros algoritmos de agrupamento que poderiam ser testados. O algoritmo SOM por sua natureza, se adapta às bases de dados, formando um mapa de semelhança. O algoritmo SOM é capaz de agrupar grupos de diversos tipos (seção 2.2) - isso o torna muito flexível para uso em diversas áreas. O algoritmo SOM aliado ao GSOM é uma ferramenta muito útil para o agrupamento de dados em diversas situações de forma autônoma.



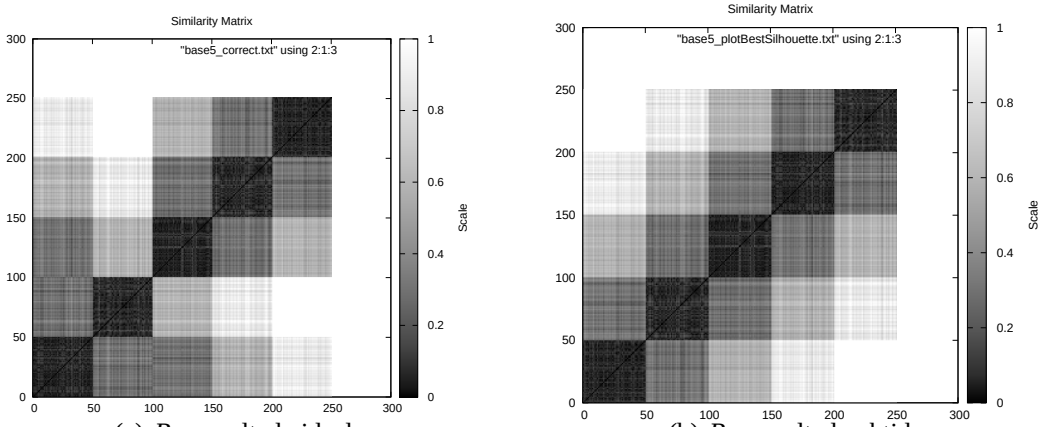
(a) B_3: resultado ideal (b) B_3: resultado obtido

Figura 6.4: Matriz de similaridade dos resultados esperados e encontrados para a base B_3



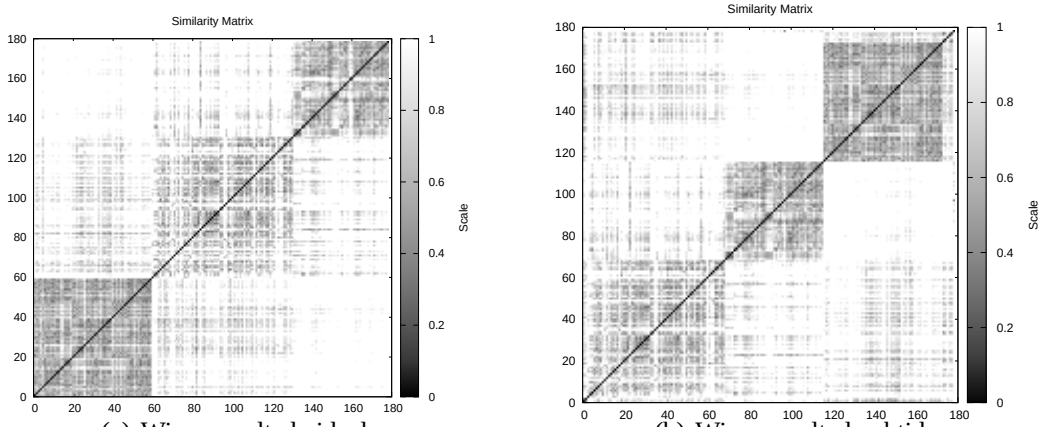
(a) B_4: resultado ideal (b) B_4: resultado obtido

Figura 6.5: Matriz de similaridade dos resultados esperados e encontrados para a base B_4



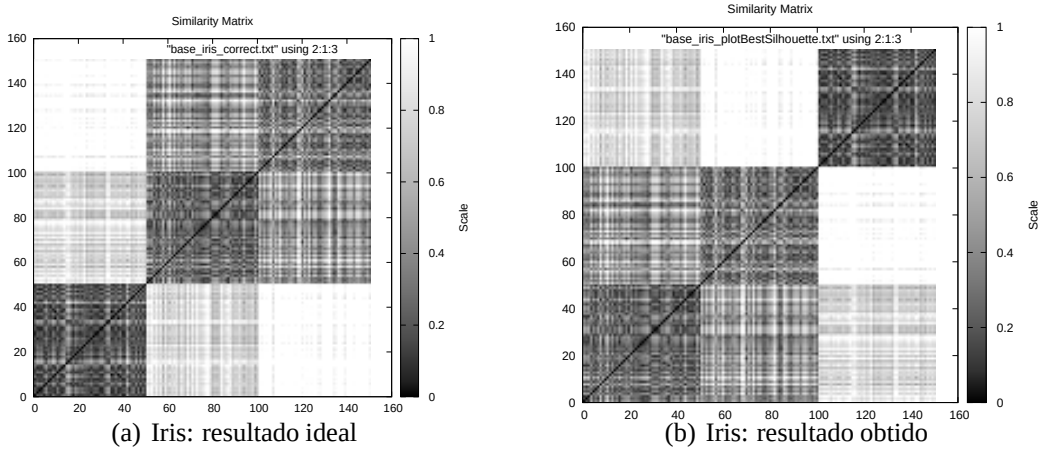
(a) B_5: resultado ideal (b) B_5: resultado obtido

Figura 6.6: Matriz de similaridade dos resultados esperados e encontrados para a base B_5



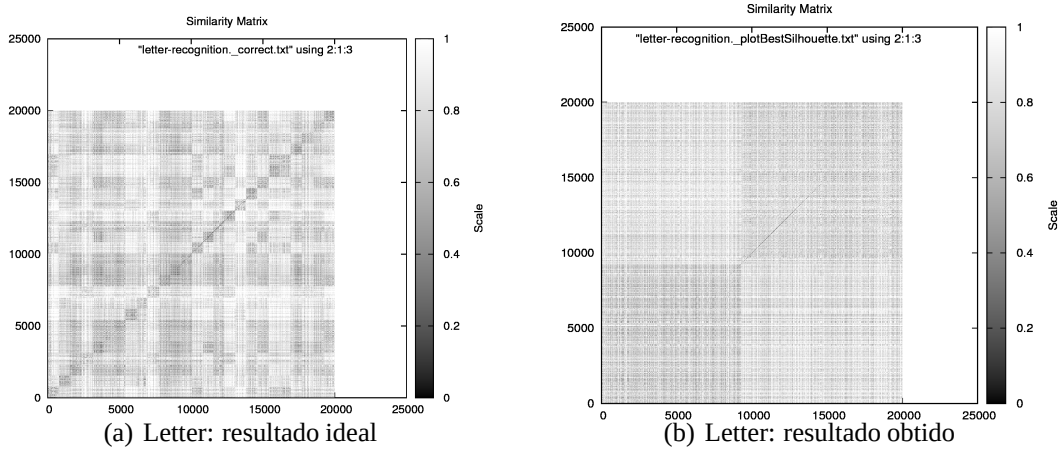
(a) Wine: resultado ideal (b) Wine: resultado obtido

Figura 6.7: Matriz de similaridade dos resultados esperados e encontrados para a base *Wine*



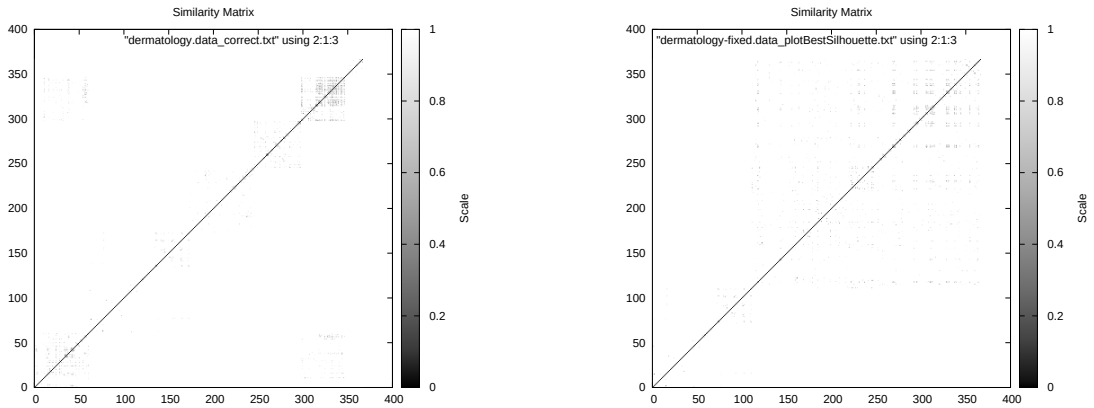
(a) Iris: resultado ideal (b) Iris: resultado obtido

Figura 6.8: Matriz de similaridade dos resultados esperados e encontrados para a base *Iris*



(a) Letter: resultado ideal (b) Letter: resultado obtido

Figura 6.9: Matriz de similaridade dos resultados esperados e encontrados para a base *Letter Recognition*



(a) Dermatology: resultado ideal (b) Dermatology: resultado obtido

Figura 6.10: Matriz de similaridade dos resultados esperados e encontrados para a base *Dermatology*

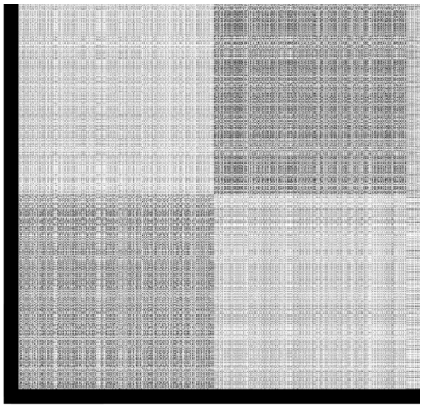


Figura 6.11: Detalhe do gráfico 6.9(a) mostrando pequena porção com 2 grupos.

Capítulo 7

Considerações Finais

Neste capítulo apresentamos as conclusões acerca do trabalho realizado e apontamos sugestões e melhorias para trabalhos futuros.

Conclusão e Trabalhos Futuros

O principal objetivo deste trabalho é realizar o agrupamento em um fluxo de dados utilizando o algoritmo SOM e mantendo-se a melhor qualidade possível dada uma medida de validação. Para tanto propomos um método para agrupamento em fluxo de dados utilizando o algoritmo SOM. O maior desafio, independente do algoritmo de agrupamento a ser utilizado, era a descoberta do número de grupos de um agrupamento. Os principais algoritmos de agrupamento necessitam de diversos parâmetros que devem ser especificados à mão, e dependem da natureza da base de dados. Isso é um fator limitante pois faz-se necessário a presença de um operador para ajustar os parâmetros. O algoritmo KMEANS em sua forma original, por exemplo, é um algoritmo onde o número de grupos deve ser especificado a priori.

Através do método proposto no capítulo 3, podemos concluir os objetivos específicos 1 e 2. Estes objetivos determinam que devemos realizar o controle do fluxo de dados e a leitura dos dados da fila. O objetivo específico 3 determina que devemos avaliar o processo de extração automática dos grupos o qual foi atingido à partir do desenvolvimento do algoritmo GSOM e constatado sua eficiência no capítulo 6. Por último, o objetivo específico 4, pode ser concluído através do simulador implementado conforme descrito no capítulo 5.

Como hipótese de trabalho, no primeiro capítulo, citamos a necessidade de aplicar o algoritmo de agrupamento SOM a um fluxo de dados e obter grupos/classes de forma não assistida. A prova da hipótese é demonstrada através do simulador implementado em conjunto com o algoritmo GSOM. Este permite que o algoritmo SOM funcione de forma não assistida, obtendo os grupos automaticamente.

O algoritmo SOM não necessita de muitos parâmetros iniciais. Os 3 parâmetros são simples e não dependem da natureza da base de dados. Em contrapartida, o SOM apresenta um grande problema após a realização do agrupamento, que é a extração dos grupos. Para resolver isso desenvolvemos o algoritmo GSOM para a extração automática dos grupos.

O algoritmo GSOM apresentou bons resultados, conforme demonstrado no capítulo 6. Embora nem sempre o número de grupos encontrados foi o correto, podemos perceber através das matrizes de similaridade que todos os agrupamentos conseguiram realizar o isolamento dos registros em grupos.

Como trabalho futuro, podemos trocar a medida de distância utilizada no nos algoritmos afim de melhorar a distribuição dos elementos no mapa após o treinamento. Assim como trocar a medida de qualidade no processo de refinamento da rede. Em nossos testes utilizamos as medidas de coesão, separação, e silhueta, obtendo através deste o melhor resultado.

Ajustes ainda devem ser feitos na interface gráfica do simulador. Testes com aplicações interativas reais estão em desenvolvimento. Uma aplicação interativa pode se conectar ao simulador através de *sockets* utilizando o protocolo de comunicação descrito no capítulo 5.

Tanto o algoritmo SOM quando GSOM podem ser paralelizados. Partes do código, como por exemplo, encontrar a BMU, funcionaria mais rápido conforme mais *threads* utilizadas. O tempo de execução do algoritmo seria reduzido proporcionalmente ao número de *threads* utilizadas.

Referências Bibliográficas

[Aggarwal et al., 2003] Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2003). A framework for clustering evolving data streams. *Very Large Data Bases*.

[Arai, 2007] Arai, K. (2007). ISODATA clustering with parameter (threshold for merge and split) estimation based on GA: Genetic Algorithm. *Science*, 36(1):17–23.

[Babcock et al., 2002] Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, page 16, New York, New York, USA. ACM.

[Babcock et al., 2004] Babcock, B., Datar, M., and Motwani, R. (2004). Load shedding techniques for data stream systems. In *Proceedings of the 20th International Conference on Data Engineering*, pages 1–3. Citeseer.

[Beringer and Hullermeier, 2006] Beringer, J. and Hullermeier, E. (2006). Online clustering of parallel data streams. *Data & Knowledge Engineering*, 58(2):180–204.

[Brugger et al., 2008] Brugger, D., Bogdan, M., and Rosenstiel, W. (2008). Automatic cluster detection in Kohonen's SOM. *Neural Networks, IEEE Transactions on*, 19(3):442–459.

[Cabanes and Bennani, 2010] Cabanes, G. and Bennani, Y. (2010). Learning the number of clusters in Self Organizing Map. *InTech*, pages 15–28.

[Carpenter et al., 1987] Carpenter, G., Grossberg, S., and Others (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied optics*, 26(23):4919–4930.

[Drobics et al., 2000] Drobics, M., Winiwater, W., and Bodenhofer, U. (2000). Interpretation of self-organizing maps with fuzzy rules. *Proceedings 12th IEEE Internationals Conference on Tools with Artificial Intelligence. ICTAI 2000*, pages 304–311.

[Dutta et al., 2005] Dutta, M., Mahanta, A. K., and Pujari, A. K. (2005). QROCK: a quick version of the ROCK algorithm for clustering of categorical data. *Pattern Recognition Letters*, 26(15):2364–2373.

[Ester et al., 1996] Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, volume 96, page 226–231.

[Faceli et al., 2005] Faceli, K., C.P.L.F. De Carvalho, A., and Carlos Pereira De Souto, M. (2005). Algoritmos de Agrupamento de Dados.

[Fausett, 1994] Fausett, L. (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice Hall Englewood Cliffs, NJ.

[Fisher, 1987] Fisher, D. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172.

[Gaber et al., 2003] Gaber, M., Krishnaswamy, S., and Zaslavsky, A. (2003). Adaptive mining techniques for data streams using algorithm output granularity. In *The Australasian Data Mining Workshop*. Citeseer.

[Gaber et al., 2005] Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: a review. *ACM SIGMOD Record*, 34(2).

[Ghaseminezhad and Karami, 2011] Ghaseminezhad, M. and Karami, a. (2011). A novel self-organizing map (SOM) neural network for discrete groups of data clustering. *Applied Soft Computing*, 11(4):3771–3778.

[Golab and Ozsu, 2003] Golab, L. and Ozsu, M. (2003). Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14.

[Gordon, 1999] Gordon, A. (1999). *Classification*. Chapman & Hall/CRC, 2 edition.

[Guha et al., 2003] Guha, S., Meyerson, A., Mishra, N., Motwani, R., and O'Callaghan, L. (2003). Clustering Data Streams : Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):515–528.

[Guha et al., 2000] Guha, S., Motwani, R., and O'Callaghan, L. (2000). Clustering Data Streams. In IEEE Press, editor, *41st Annual Symposium on Foundations of Computer Science (FOCS)*.

[Guha et al., 1998] Guha, S., Rastogi, R., and Shim, K. (1998). CURE: an efficient clustering algorithm for large databases. *International Conference on Management of Data*, 27(2).

[Guha et al., 1999] Guha, S., Rastogi, R., and Shim, K. (1999). ROCK: A Robust Clustering Algorithm for Categorical Attributes. *ICDE*, page 512.

[Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data Clustering: A Review.

[Karypis and Kumar, 1999] Karypis, G. and Kumar, V. (1999). Chameleon: hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75.

[Kohonen, 1990] Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480.

[Kumar, 2000] Kumar, V. (2000). An Introduction to Cluster Analysis for Data Mining.

[Kumar et al., 2006] Kumar, V., Tan, P., and Steinbach, M. (2006). Cluster Analysis: Basic Concepts and Algorithms. In *Introduction to Data Mining*, chapter 8, pages 487–586.

[Longo and Barrett, 2009] Longo, L. and Barrett, S. (2009). A context-aware approach based on self-organizing maps to study web-users' tendencies from their behaviour. In *Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009)*, pages 12–17. ACM.

[Muthukrishnan, 2005] Muthukrishnan, S. (2005). *Data streams: Algorithms and applications*. Now Publishers Inc.

[O' Callaghan et al., 2002] O' Callaghan, L., Mishra, N., Meyerson, A., Guha, S., and Motwani, R. (2002). Streaming-Data Algorithms for High-Quality Clustering. In *Proceedings of the International Conference on Data Engineering*, page 685–696. IEEE Computer Society Press; 1998.

[Pelleg and Moore, 2000] Pelleg, D. and Moore, A. (2000). X-means: Extending K-means with E cient Estimation of the Number of Clusters. In *Proceedings of the 17th International Conference on Machine Learning*, pages 727–734, San Francisco. Morgan Kaufmann.

[Russell et al., 1995] Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. (1995). *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, NJ, 2 edition.

[Sharma and Omlin, 2006] Sharma, A. and Omlin, C. (2006). Determining cluster boundaries using particle swarm optimization. In *Proceedings of world academy of science engineering and technology*, number 1, pages 250–254.

[Ultsch, 2003] Ultsch, A. (2003). U*-Matrix: a Tool to visualize Clusters in high dimensional Data. Technical report, Dept. of Mathematics and Computer Science, University of Marburg, Germany.

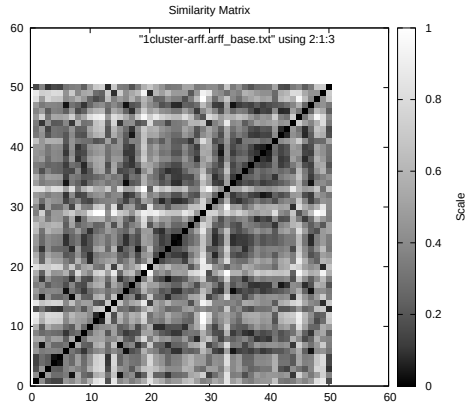
[Ultsch and Mörchen, 2006] Ultsch, A. and Mörchen, F. (2006). U-maps: topograpic visualization techniques for projections of high dimensional data. In *Proceedings of 29th Annual Conference of the German Classification Society (GfKl 2006)*.

[Wu et al., 2008] Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., and Others (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37.

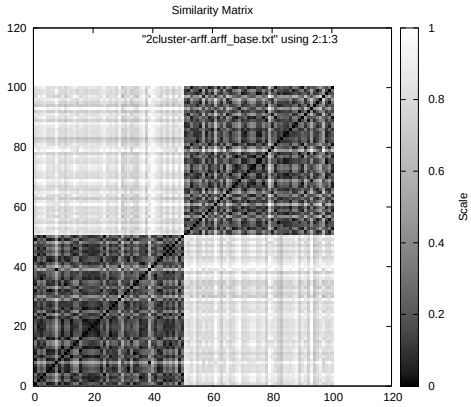
[Zhang et al., 1996] Zhang, T., Ramakrishnan, R., and Livny, M. (1996). BIRCH: an efficient data clustering method for very large databases. *ACM SIGMOD International Conference on Management of Data*, 25(2):103–114.

Apêndice A

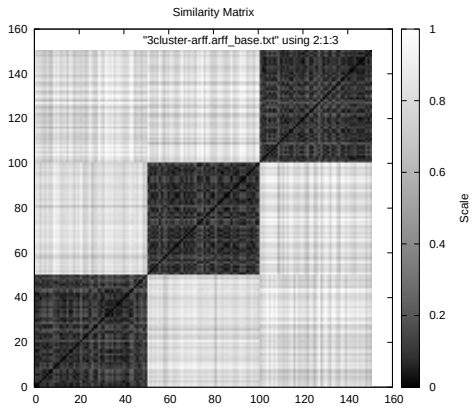
Resultados dos algoritmos KMEANS e DBSCAN



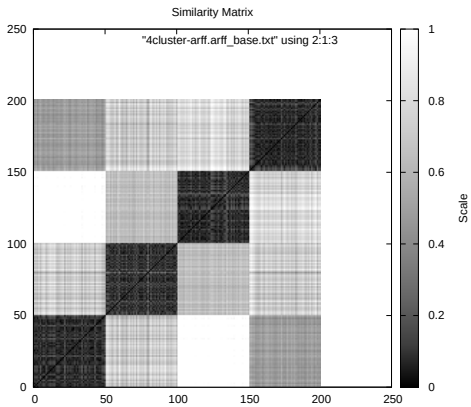
(a) KMEANS: base de dados B_1



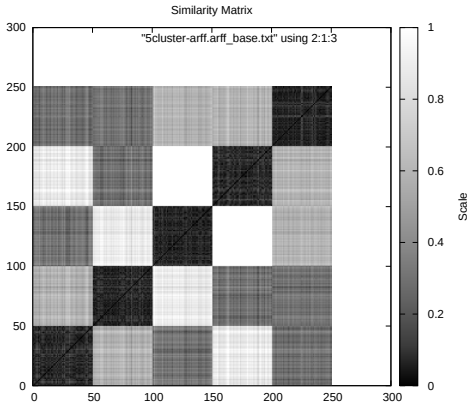
(b) KMEANS: base de dados B_2



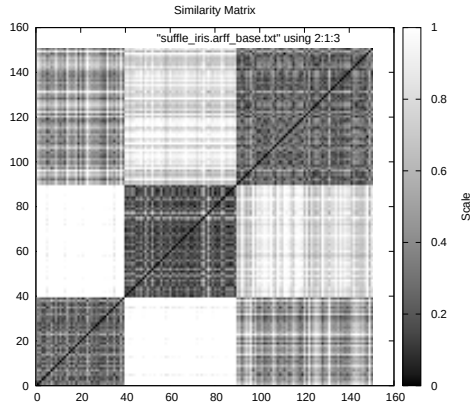
(c) KMEANS: base de dados B_3



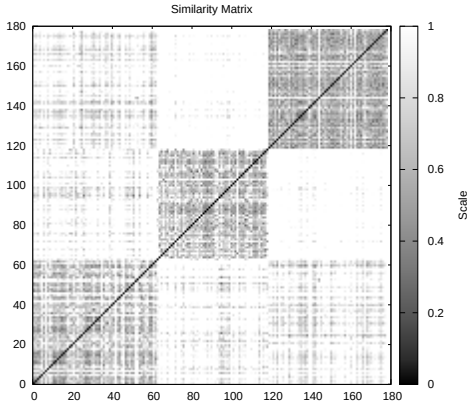
(d) KMEANS: base de dados B_4



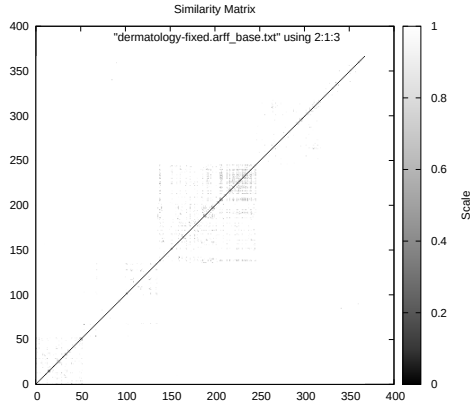
(e) KMEANS: base de dados B_5



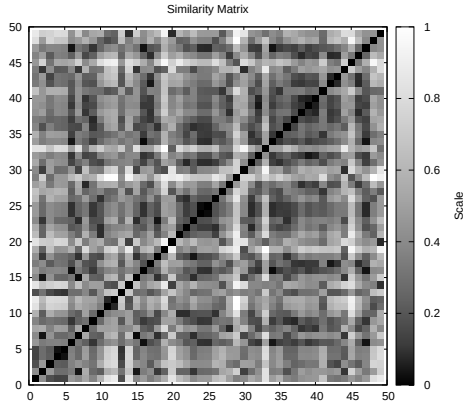
(f) KMEANS: base de dados Iris



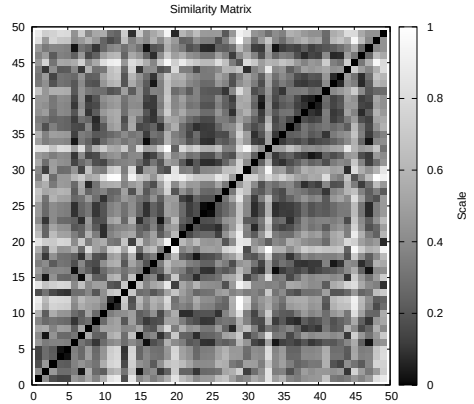
(g) KMEANS: base de dados Wine



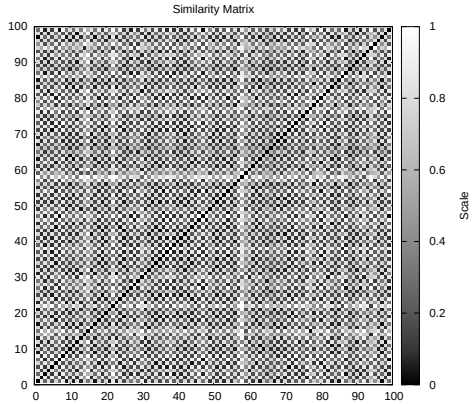
(h) KMEANS: base de dados Dermatology



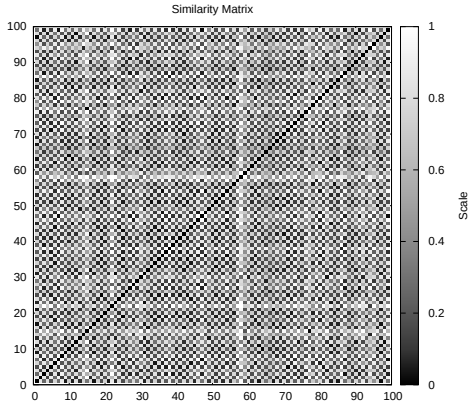
(i) DBSCAN $\varepsilon = 0.9$, $num_{pts} = 6$: base de dados B_1



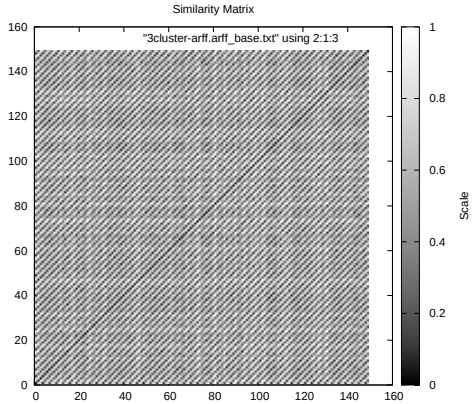
(j) DBSCAN $\varepsilon = auto$, $num_{pts} = 6$: base de dados B_1



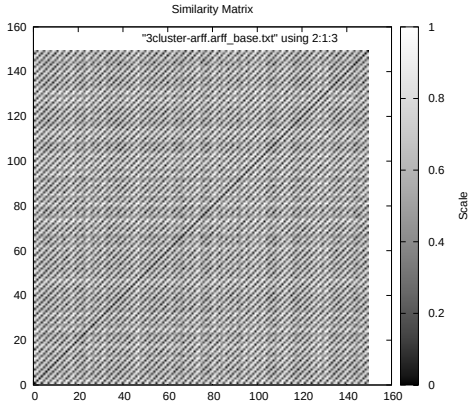
(k) DBSCAN $\varepsilon = 0.9$, $num_{pts} = 6$: base de dados B_2



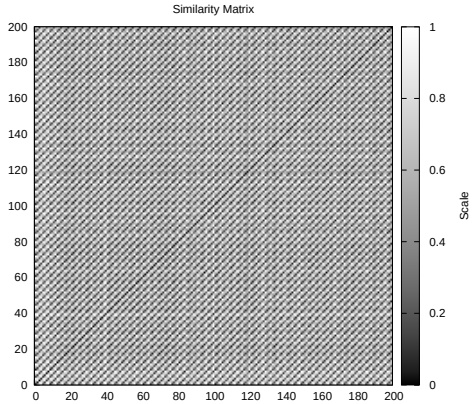
(l) DBSCAN $\varepsilon = auto$, $num_{pts} = 6$: base de dados B_2



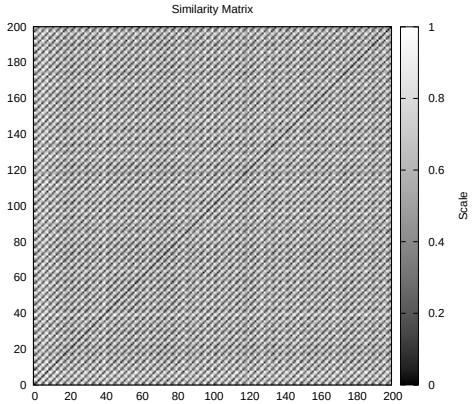
(m) DBSCAN $\varepsilon = 0.9$, $num_{pts} = 6$: base de dados B_3



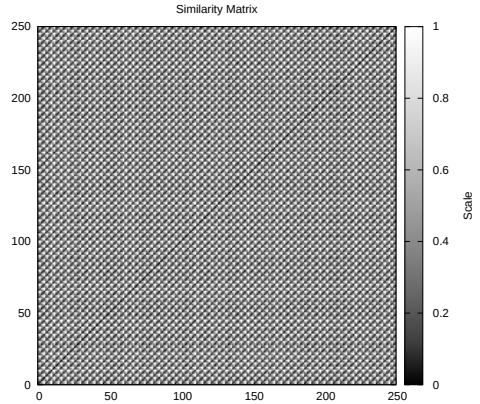
(n) DBSCAN $\varepsilon = auto$, $num_{pts} = 6$: base de dados B_3



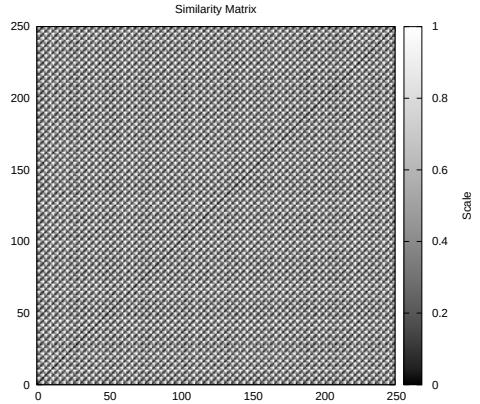
(o) DBSCAN $\varepsilon = 0.9$, $num_{pts} = 6$: base de dados B_4



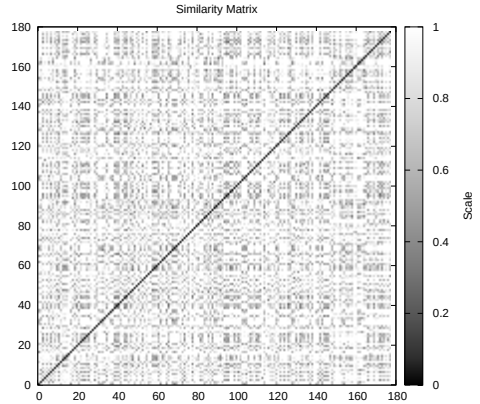
(p) DBSCAN $\varepsilon = auto$, $num_{pts} = 6$: base de dados B_4



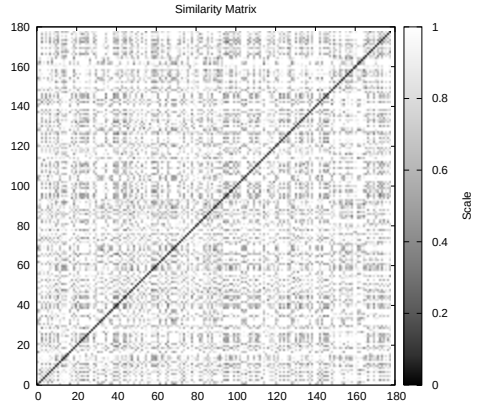
(q) DBSCAN $\varepsilon = 0.9$, $num_{pts} = 6$: base de dados B_5



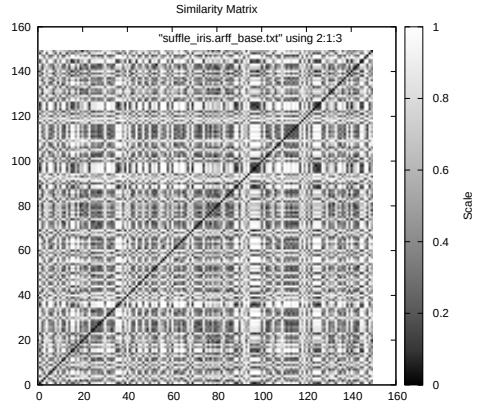
(r) DBSCAN $\varepsilon = auto$, $num_{pts} = 6$: base de dados B_5



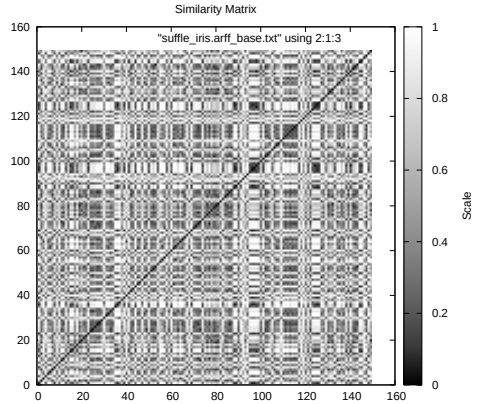
(s) DBSCAN $\varepsilon = 0.9$, $num_{pts} = 6$: base de dados Wine



(t) DBSCAN $\varepsilon = auto$, $num_{pts} = 6$: base de dados Wine



(u) DBSCAN $\varepsilon = 0.9$, $num_{pts} = 6$: base de dados Iris



(v) DBSCAN $\varepsilon = auto$, $num_{pts} = 6$: base de dados Iris