UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

COORDENAÇÃO DE TECNOLOGIA EM ANÁLISE E DESENVILVIMENTO DE SISTEMAS

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANDRÉ LUÍS FERREIRA

MAPAS AUTO-ORGANIZÁVEIS NA DESCOBERTA E VALIDAÇÃO DE PADRÕES EM BASES DE DADOS

TRABALHO DE DIPLOMAÇÃO

PONTA GROSSA

2012

ANDRÉ LUÍS FERREIRA

MAPAS AUTO-ORGANIZÁVEIS NA DESCOBERTA E VALIDAÇÃO DE PADRÕES EM BASES DE DADOS

Trabalho de Conclusão de Curso, apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, da COADS, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Marcos Vinicius Fidelis

PONTA GROSSA

2012

TERMO DE APROVAÇÃO

MAPAS AUTO-ORGANIZÁVEIS NA DESCOBERTA E VALIDAÇÃO DE PADRÕES EM BASES DE DADOS

por

ANDRÉ LUÍS FERREIRA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 05 de Junho de 2012 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Marcos Vinicius Fidélis
Prof.(a) Orientador(a)

_____
Helyane Bronoski Borges
Membro titular

_____
Simone de Almeida
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

Dedico este trabalho em primeiro lugar a Deus, por tudo que tem me proporcionado na vida.

Aos meus pais Daniel Jorge Ferreira e Josely M. Maravieski Ferreira, que abriram mão dos seus sonhos e seus desejos para que eu pudesse realizar o meu.

À minha amiga e namorada Aline Buenos Aires, ao meu irmão amigo Vitor G. Ferreira e a meus Professores, que de alguma forma colaboraram para a elaboração desta pesquisa.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço aos meus pais Daniel Jorge Ferreira e Josely M. Maravieski Ferreira, que são os meus maiores apoiadores na vida.

Agradeço também à minha amiga e namorada Aline Buenos Aires, pelo carinho, compreensão e companheirismo.

Agradeço ao meu orientador Professor Marcos Vinicius Fidelis, pela sabedoria e paciência com que me guiou nesta trajetória.

Agradeço também a Professora Ms. Helyane Bronoski Borges, pelos ensinamentos compartilhados através do grupo de pesquisas de mineração de dados, que me induziram a iniciar esta pesquisa.

Agradeço ainda a Professora Simone de Almeida pela colaboração e confiança depositada e que resultaram na conclusão deste trabalho.

Aos meus colegas de sala, pela amizade e parceria durante todo o período de faculdade.

A Secretaria do Curso, pela cooperação.

Gostaria de deixar registrado também, o meu reconhecimento à toda minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

Não basta ensinar ao homem uma especialidade, porque se tornará assim uma máquina utilizável e não uma personalidade. É necessário que adquira um sentimento, senso prático daquilo que vale a pena ser empreendido, daquilo que é belo, do que é moralmente correto.
(EINSTEIN, Albert, 1879-1955)

RESUMO

FERREIRA, André Luís. **Mapas Auto Organizáveis na Descoberta e Validação de Padrões em Bases de Dados**. 2012. 74f. Trabalho de Conclusão do Curso de Análise e Desenvolvimento de Sistemas - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2012.

O Mapa Auto Organizável de Kohonen (SOM) é um excelente e poderoso algoritmo para mineração de dados baseado em Redes Neurais Artificiais (RNA). Através do SOM é possível fazer a análise e o processamento de bases de dados de grande dimensão, e então obter os agrupamentos dos dados presentes nestas bases de dados, de forma que os itens de cada grupo compartilham propriedades semelhantes em comum. A partir dos agrupamentos formados pelo SOM, é possível então realizar uma análise focada dos agrupamentos resultantes. Em alguns casos, entretanto, o SOM pode trazer agrupamentos de dados não classificáveis ou entendíveis para o contexto, uma vez que ele agrupa os dados baseado em todas as diversas características deles. Para que os resultados dos agrupamentos sejam válidos e tragam informações úteis, é importante que os dados sejam pré-processados para se adequarem ao algoritmo, e posteriormente, que a classificação realizada pelo SOM seja validada a fim de se obter a precisão com que o SOM classificou os dados. Com a padronização dos dados e a posterior validação dos agrupamentos é então possível verificar a confiabilidade do algoritmo. A partir dos dados resultantes de todos estes processos é possível então extrair estes dados em uma forma humanamente entendível, trazendo informações que anteriormente não seriam compreensíveis ou previsíveis nos dados brutos, favorecendo desta forma a interpretação dos resultados.

Palavras-chave: Mapas Auto Organizáveis. SOM. Redes Neurais Artificiais. Mineração de Dados. Agrupamento de Dados. Validação de Agrupamentos.

ABSTRACT

FERREIRA, André Luís. **Self-Organizing Maps on Discovery and Validation of Patterns in Data Bases**. 2012. 74f. Completion Course Work – Systems Analysis and Development - Federal Technology University of Paraná. Ponta Grossa, 2012.

The Self-Organizing Maps (SOM) of Kohonen is an excellent and powerful algorithm for data mining based in Artificial Neural Networks (ANN). Through the SOM is possible to do processing and analysis of databases of large size, and then discover/retrieve the items in the data groups present in the database, having the items in each group that share similar properties in common. From the clusters formed by SOM, is possible then perform an focused analysis of the resulting clusters. In some cases however, the SOM find groups of data not classified or understandable to the context, since it groups the data based on all of its n characteristics. For the results of clustering being/get valid and useful information, is important that the data be pre-processed to suit the algorithm, and subsequently the classification performed by the SOM be/is/get validated in order to obtain the precision with which the SOM classified the data. With the standardization of the data and subsequent validation of the clusters is then possible to verify the reliability of the algorithm. From the data resulting from all these processes it is now possible to extract this data in a humanly understandable way, providing information that previously would not be understandable or predictable in the raw data, favoring the interpretation of results.

Keywords: Self-Organizing Maps. SOM. Artificial Neural Networks. Data Mining. Data Clustering. Clustering Validation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

Ex.	Número da experiência realizada
E.M.	Erro médio do resultado apresentado

LISTA DE SIGLAS

SOM	Mapas Auto Organizáveis (Self-Organizing Maps)
JKNNL	Java Kohonen Neural Network Library
KDD	Descoberta de conhecimento em bases de dados (Knowledge Discovery in Databases)
RNA	Redes Neurais Artificiais
BP	Retro Propagação (Back Propagation)

LISTA DE FUNÇÕES

SUMÁRIO

1 INTRODUÇÃO

A base para qualquer estudo fundamentado ou “orientado à aplicação” se dá através da análise de dados. Tradicionalmente técnicas de transformações lineares e redução de dados são aplicadas para este fim. A vantagem da utilização de técnicas de redução dos dados, é que em geral, estruturas de dados de alta dimensionalidade podem ser projetadas em sistemas de baixa coordenada[1]. Os dados em questão podem ser facilmente visualizados e analisados por gráficos de pontuação dos objetos, provendo informações sobre as correlações existentes entre as variáveis de entrada.

O Cérebro humano por sua vez é considerado o mais eficiente processador baseado em carbono existente descreve Tatibana (2000), e de acordo com Blinkov e Glezer (1968) ele é composto por aproximadamente 100 bilhões de neurônios que conectam entre si e juntos formam uma grande Rede Neural, o que proporciona uma fabulosa capacidade de processamento e armazenamento de informações.

Com o advento de melhores tecnologias, além dos próprios avanços da computação, se tornou então possível a criação de modelos virtuais os quais podem reproduzir de forma similar tais características fisiológicas do cérebro. Hoje é possível simular o processo de execução de um neurônio natural, porem ainda é impossível alcançar o poder de processamento de uma rede neural natural devido à diversas restrições de processamento, especialmente pela limitação dos atuais processadores na realização de processamento paralelo, que é a maior característica do cérebro humano.

À aproximadamente três décadas atrás, Kohonen intruduziu uma elegante técnica de mapeamento para avaliação de conjuntos de dados com estruturas de alta dimensionalidade: os Mapas Auto-Organizáveis (*Self-Organizing Maps* - SOM). O SOM segundo Melssen et al. (2006), incorpora de forma não supervisionada a topologia presente nos dados. Intuitivamente o funcionamento de uma rede de Kohonen pode ser comparado com à bem conhecida Projeção de Mercator (utilizado

[1] Sistemas que visam reduzir a complexidade dos dados a serem visualizados pelo usuário.

em mapas mundi), a qual mapeia o globo Terrestre tridimensional, em uma representação plana bidimensional.

Este trabalho inicia uma pesquisa na área de reconhecimento de padrões em bases de dados, passando por etapas de ‘Descoberta do Conhecimento em Bancos de Dados’ como o pré-processamento, aonde os dados serão preparados para o treinamento, a mineração de dados, onde será aplicada a utilização de um algoritmo de redes neurais baseado em Mapas Auto Organizáveis, e finalmente haverá o pós-processamento, que apresentará os dados de forma simplificada, permitindo uma análise dos resultados, comparando estes resultados com os resultados extraídos por meio de outras aplicações de mineração de dados.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Estudar e aplicação do algoritmo de redes neural denominado Mapas Auto Organizáveis através da biblioteca denominada *Java Kohonen Neural Network Library* (JKNNL), bem como o desenvolvimento de métodos para extração e validação dos índices obtidos dos agrupamentos resultantes, que serão utilizados para posterior análise e comparação com os resultados extraídos através de outros algoritmos de agrupamento de dados existentes na aplicação WEKA (*Software* de Mineração de Dados da Universidade de Waikato) (HOLMES et al., 1994).

1.1.2 Objetivos Específicos

1. Simular e comparar o desempenho do algoritmo neural competitivo SOM com outros algoritmos e publicações que tratam o tema de análise de agrupamentos;
2. Customizar a biblioteca JKNNL a fim de se armazenar informações relativas ao processo de treinamento do algoritmo SOM e adquirir conhecimento dos dados processados;

3. Comparar as técnicas de validação de resultados através dos índices: Silhuetas Simplificadas, Silhuetas e Davies-Bouldin, para as tarefas de análise e validação de agrupamentos;
4. Avaliar os agrupamentos formados, de acordo com cada base de dados utilizada no processo de treinamento e aprendizado da rede SOM;
5. Investigar o efeito de cada método de validação de resultados na determinação do número ótimo de neurônios ou protótipos;
6. Avaliar os resultados dos índices implementados para cada bateria de testes;

1.2 PROBLEMA

Atualmente um grande volume de dados é gerado por sistemas computacionais, dos quais grande parte são dados numéricos, que podem ser oriundos de transações bancárias como no estudo de SHIN e SOHN (2003), informações geográficas como no exemplo de TAKATSUKA (2001) ou até mesmo, por exemplo, dados gerados em tempo real providos por algum sensor como nos experimentos de NAGI et al (2008).

Apesar de dados deste tipo aparentemente não terem relacionamento entre si, eles podem conter informações as quais podem ser agrupadas de acordo com algum critério específico e desta forma definir tendências ou relacionamentos entre estes dados aparentemente não correlatos.

Nesses casos a mineração de dados desponta como um método para se aplicar técnicas especificas sobre um conjunto de dados, com o objetivo de revelar padrões, similaridades ou diferenças, e produzir regras e resumos a partir destes dados (FAYYAD et al. 1996).

Dos conjuntos de dados formados nestes processos pode ainda haver dúvidas da precisão e validade dos resultados, uma vez que métodos de agrupamento de dados baseados em redes neurais trabalham com conceitos de auto-organização, e não com técnicas lineares que demonstram de forma mais simples a origem de seus resultados.

1.3 JUSTIFICATIVA

É relativamente fácil dar respostas a questões bem formuladas sobre a natureza estatística de um conjunto de dados em que se tenha amplo conhecimento, como por exemplo, "quão grande deve se o *cockpit* de um avião para acomodar 95% de deus potenciais pilotos?" ou “qual o modelo de celular com tela *touchscreen* vendeu mais no natal de 2011?”. Nestes exemplos assume-se que os dados sejam proporcionalmente distribuídos e é simples estimar a resposta. Quanto mais dados há disponível, mais precisa uma resposta pode ser dada. Entretanto, se não há muito conhecimento sobre os dados, e o problema a ser analisado não for claramente especificado, um aumento na quantidade de dados talvez possa ter o efeito oposto. Isso vale para dados multivariados[2], em particular. Se o objetivo é simplesmente tentar encontrar sentido no conjunto de dados para gerar hipóteses lógicas ou encontrar alguns novos padrões interessantes, paradoxalmente parece que quanto mais dados estiverem disponíveis, mais difícil será de entender o conjunto de dados. As estruturas estarão escondidas no meio da grande quantidade de dados multivariados.

Quando se explora um conjunto de dados para novos conhecimentos, somente métodos que “descobrem e ilustram eficazmente as estruturas nos dados” podem ajudar. Tais métodos aplicados a grandes conjuntos de dados serão apresentados neste trabalho.

Devido à necessidade de se adquirir conhecimento a partir dos dados gerados por sistemas computacionais, surge necessidade da criação de técnicas mais adaptáveis e inteligentes, que busquem informações não lineares em bases de dados, e que possam validar informações que são somente observadas, mas não comprovadas.

A Mineração de Dados surge então como uma técnica cujo objetivo está relacionado com o processo de KDD (*Knowledge Discovery in Database*). O

---

[2] Contém tanto valores reais (números), quanto texto (sequencia de caracteres).

processo de KDD é formado pela interseção de diferentes áreas como a inteligência computacional, onde várias técnicas podem ser utilizadas como, por exemplo, as Redes Neurais Artificiais (RNA).

As RNA são modelos computacionais não lineares, inspirados na estrutura e operação do cérebro humano, que procuram reproduzir características humanas, tais como: aprendizado, associação, generalização e abstração e visam gerar conhecimentos úteis de informações desconhecidas como explica Kohonen (1990).

Existem vários algoritmos que implementam essa abordagem. Esse trabalho visa à aplicação de Redes Neurais com Mapas Auto Organizáveis (*Self Organizing Maps* - SOM) ou também conhecidos como Mapas de Kohonen, devido o nome de seu criador, Teuvo Kohonen.

Este algoritmo destaca-se pelo fato de ser um algoritmo de aprendizagem não supervisionada, ou seja, não existe a necessidade de agentes externos indicando a resposta que a rede deve dar. Assim, a sua aplicação em bases de dados permite a descoberta de grupos de dados que apresentam características similares.

Este trabalho inclui o estudo e aplicação de métodos de validação, dos quais serão gerados índices estatísticos que julgarão de maneira qualitativa os agrupamentos formados.

Os métodos escolhidos podem usar critérios internos, externos ou relativos para investigar a validade de um agrupamento (FACELI et. al., 2005).

Como resultado deste trabalho será possível entender na prática o funcionamento da mineração de dados e de sistemas inteligentes, uma vez que serão aplicadas várias etapas do processo de KDD. As implementações feitas no algoritmo da biblioteca utilizada, tornarão possíveis a obtenção de resultados que tragam informações ricas para a análise dos resultados, transformando a biblioteca em um sistema de mineração de dados voltado para a análise de agrupamentos de dados.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está desenvolvido em cinco capítulos.

O capítulo dois aborda a fundamentação pesquisada para apoiar e desenvolver o trabalho, contendo informações detalhadas dos principais assuntos que fazem parte dos temas abordados.

No capitulo três são apresentadas as informações pertinentes a metodologia utilizada para que os objetivos propostos pudessem ser alcançados, tratando de informações referentes a alterações e configurações feitas nos algoritmos utilizados.

O capitulo quatro apresenta uma análise dos resultados obtidos com os experimentos que foram definidos e configurados no capitulo anterior.

Por fim, o capítulo cinco apresenta a conclusão e a contribuição do trabalho desenvolvido, além de sugestões para o desenvolvimento de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem por objetivo apresentar os conceitos fundamentais utilizados no desenvolvimento deste trabalho. A Seção 2.1 descreve sucintamente o processo de descoberta em base de dados e a mineração de dados, focando a tarefa de agrupamento, a qual é ponto fundamental para o desenvolvimento do trabalho. A Seção 2.2 Irá relacionar e explicar índices para avaliação de agrupamentos. A Seção 2.3 descreve-se sobre Redes Neurais Artificiais, focando particularmente os Mapas Auto Organizáveis, que é o tema principal do trabalho.

2.1 DESCOBERTA DE CONHECIMENTO EM BASES DE DADOS (KDD)

A análise de grandes quantidades de dados pelo homem é inviável sem o auxílio de ferramentas computacionais apropriadas. Portanto, torna-se imprescindível o desenvolvimento de ferramentas que auxiliem, de forma automática e inteligente a tarefa de analisar, interpretar e relacionar esses dados para que possa se selecionar e desenvolver estratégias de ação em cada contexto de aplicação (GOLDSCHMIDT e PASSOS, 2005).

É neste contexto então que surge uma área que vem despertando grande interesse nas comunidades científicas, bem como nas áreas industrial e comercial. Esta área denomina-se ‘Descoberta do Conhecimento em Bancos de Dados’ (*Knowledge Discovery in Database* - KDD) (FAYYAD, 1996). A expressão Mineração de Dados, que é mais popular até mesmo que o KDD, na verdade é uma das etapas da Descoberta de Conhecimento em Bases de Dados.

A definição aceita por diversos pesquisadores de Descoberta de Conhecimento em Bases de Dados foi elaborada por Fayyad como sendo: “Extração de Conhecimento de Bases de Dados é o processo de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis embutidos nos dados” (Silva apud Fayyad et al., 2004). Examinando estes termos individualmente:

> **Dados:** conjunto de fatos F, como instâncias de um banco de dados. Por exemplo, uma coleção de n cadastros de pessoas físicas contendo idade, profissão, renda etc.

> **Padrão:** expressão E em uma linguagem L descrevendo fatos em um subconjunto F_E de F. E é dito um padrão se é mais simples do que a enumeração de todos os fatos em F_E. Por exemplo, o padrão: "Se renda < R$ então a pessoa não recebe financiamento" seria aplicável para uma escolha apropriada de r.
>
> **Processo:** geralmente em KDD, processo é uma sequência de vários passos que envolvem preparação de dados, pesquisa de padrões, avaliação de conhecimento, refinação envolvendo iteração e modificação.
>
> **Válido:** os padrões descobertos devem ser válidos em novos dados com algum grau de certeza. Uma medida de certeza é uma função C mapeando expressões em L para um espaço de medidas M_C. Por exemplo, se um limite de padrão de crédito é ampliado, então a medida de certeza diminuiria, uma vez que mais financiamentos seriam concedidos a um grupo até então restrito a esta operação.
>
> **Novo:** em geral, assume-se que "novidade" pode ser medida por uma função $N(E, F)$, que pode ser uma função booleana ou uma medida que expresse grau de "novidade" ou "surpresa". Exemplo de um fato que não é novidade: sejam $\mathrm{E} = \text{"usa tênis"}$ e $F = \text{"}alunos\ de\ colégio\text{"}$ então $\mathrm{N(E, F)} = 0$ ou $\mathrm{N(E, F)} = \mathrm{false}$. Por outro lado: sejam E = "bom pagador" e $F = \text{"}trabalhador\ da\ construção\ civil\text{"}$ então $N(E, F) = 0{,}85$ ou $N(E, F) = true$.
>
> **Potencialmente útil:** padrões devem potencialmente levar a alguma atitude prática, conforme medido por alguma função de utilidade. Por exemplo, regras obtidas no processo podem ser aplicadas para aumentar o retorno financeiro de uma instituição.
>
> **Compreensível:** um dos objetivos de KDD é tornar padrões compreensíveis para humanos, visando promover uma melhor compreensão dos próprios dados. Embora seja um tanto subjetivo medir compreensibilidade, um dos fatores frequentes é a medida de simplicidade. O fator de compreensão dos dados está relacionado à intuitividade da representação destes, bem como da granularidade alta o suficiente para que estes sejam compreendidos. Por exemplo: o log de um servidor Web não é uma representação compreensível; já fatos estatísticos extraídos deste log, tais como totais de acesso ou classificação dos acessos realizados, fornecem informação num formato mais intuitivo e de granularidade humanamente compreensível. (SILVA, 2004, p.2).

O processo KDD é definido por um conjunto de etapas, envolvendo desde o entendimento do domínio da aplicação até a interpretação e consolidação dos resultados. Durante este processo são realizadas as etapas como, por exemplo: a seleção dos dados a serem utilizados, pré-processamento dos dados para prepara-los para transformação e desta forma torná-los mais eficientes, a transformação dos dados para um padrão utilizável e compatível com a técnica de mineração que se deseja utilizar, a mineração de dados em si que é realizada através da aplicação de algum algoritmo, e finalmente a interpretação dos dados resultantes, gerando então a partir deles conhecimento. Um exemplo desse processo pode ser observado na Figura 1.

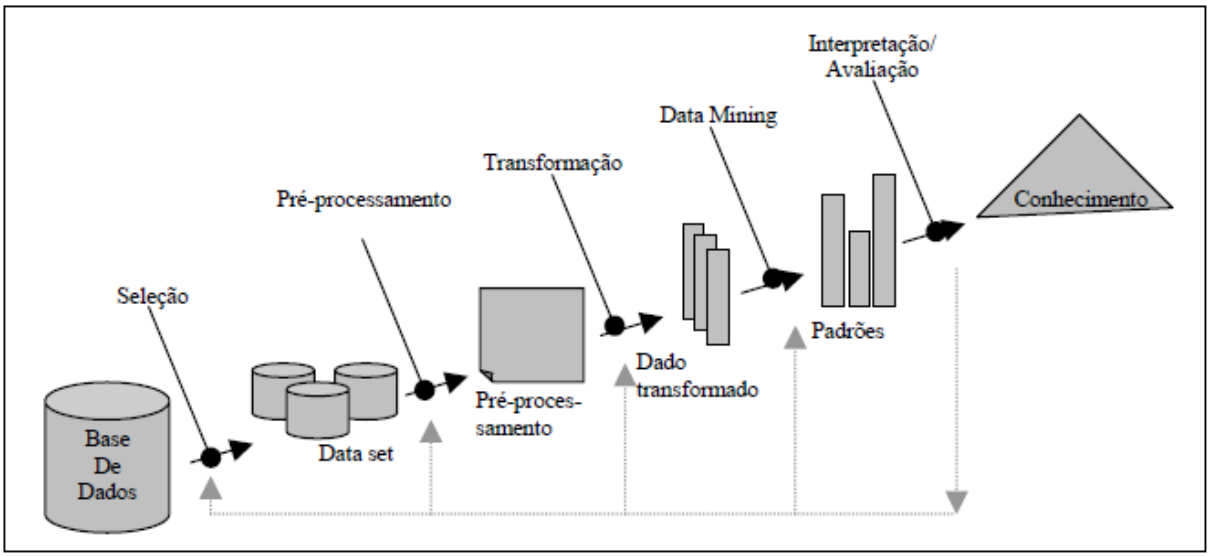

Figura 1 - Processos de KDD.
Fonte: Fayyad et al. (1996a) apud Gonçalves et al . (2001).

Normalmente, os dados disponíveis para análise não estão em um formato adequado para a Extração de Conhecimento. Além disso, em razão da limitação de memória ou de tempo de processamento, muitas vezes não é possível a aplicação direta dos algoritmos de extração de padrões de dados. Dessa maneira, torna-se necessária a aplicação de métodos para tratamento, limpeza e redução do volume de dados antes de se iniciar a etapa de Extração de Padrões. É importante salientar que a execução das transformações deve ser guiada pelos objetivos do processo de extração, de modo que o conjunto de dados gerados apresente características necessárias para que os objetivos sejam cumpridos (Rezende, 2002).

Nos dados a serem utilizados ainda pode haver a necessidade de se reduzir o número de atributos quando um mesmo atributo, para um dado elemento, apresenta mais de um valor. Para este fim usa-se a redução de valores por suavização, que consiste na substituição de um atributo contínuo (inteiro ou real) por um atributo que represente a média dos atributos válidos que se repetem.

2.1.1 Mineração de Dados

Mineração de Dados é a etapa que realiza a busca efetiva de conhecimentos úteis na base de dados. Nesta etapa, é escolhido o método e são definidos os algoritmos que realizarão a busca pelo conhecimento implícito e útil do conjunto de dados. Ela utilizará técnicas baseadas em analise estatística e Inteligência Artificial (IA), que no caso deste trabalho utilizaremos a técnica de Aprendizagem de

Máquina. A mineração de dados tem ainda como as principais tarefas utilizadas nessa etapa a associação, a classificação e o agrupamento de dados.

2.1.2 Clusterização: Agrupamento de Dados

A palavra Clusterização remete a agrupamento de registros, observações, ou relacionamentos dentro de classes de objetos similares, e por teste motivo, neste trabalho será definida como padrão a utilização do termo “Agrupamento de Dados” para representá-la.

Agrupamentos de dados podem ser encontrados sobre diferentes nomes em diferentes contextos, tais como aprendizado não supervisionado (em reconhecimento de padrões), taxonomia numérica (na biologia, ecologia), tipologia (em ciências sociais) e particionamento (na teoria dos grafos) (THEODORIDIS e KOUTROUMBAS, 1999).

Um *cluster*, ou agrupamento é uma coleção de registros que são similares entre si, e diferentes de registros em outros agrupamentos. A técnica de agrupamento de dados diferencia-se da classificação de modo que não há uma variável almejada para o agrupamento. A tarefa de agrupamento não tenta classificar, estimar ou prever o valor de uma variável almejada. Em vez disso, algoritmos de agrupamento procuram segmentar todo o conjunto de dados em subgrupos relativamente homogêneos ou agrupamentos aonde a similaridade dos registros dentro do grupo é maximizada e a similaridade com registros fora do grupo é minimizada.

Os registros serão agrupados de forma que os elementos de um determinado grupo compartilhem de propriedades comuns que os diferenciem de elementos em outros grupos, extraindo-se assim agrupamentos similares dos dados, maximizando similaridades intra-*cluster*[3] e minimizando as similaridades inter-*cluster*[4].

Diferente da classificação que tem rótulos predefinidos, a clusterização precisa automaticamente identificar os rótulos, o que traz à técnica de agrupamento

---

[3] Similaridade interna de todos os elementos de um único agrupamento.
[4] Similaridade de elementos de um agrupamento com os demais agrupamentos formados.

de dados a denominação de indução não supervisionada. Agrupamento de dados pode ser definido como uma tarefa básica da Mineração de Dados que auxilia o usuário a realizar agrupamentos naturais de registros em um conjunto de dados.

A análise dos *clusters* envolve, portanto, a organização de um conjunto de padrões (usualmente representados na forma de vetores de atributos ou pontos em um espaço multidimensional) em *clusters*, de acordo com alguma medida de similaridade. Intuitivamente padrões pertencentes a um dado grupo devem ser mais similares entre si (compartilham um conjunto de propriedades em comum) do que em relação a padrões pertencentes a outros grupos.

Após a definição desses grupos é possível fazer uma análise dos elementos que compõe cada um deles, identificando as características comuns aos seus elementos, desta forma, podendo criar um rótulo que represente cada grupo (GOLDSCHMIDT, PASSOS, 2005).

Uma vastidão de métodos de Data Mining são propostos pela literatura. Os algoritmos de clusterização podem ser classificados de acordo com:

- O tipo de dados de entrada do algoritmo.
- Os critérios de agrupamento definindo a similaridade entre os dados.
- A teoria e conceitos fundamentais das técnicas de análise de agrupamentos em que são baseadas (teoria *fuzzy*, estatísticas, etc.).

Portanto de acordo com o método adotado para definir os agrupamentos, o algoritmo pode ser amplamente classificado dentro dos seguintes tipos: (JAIN et al.,1999):

Agrupamento particional: tenta diretamente decompor o conjunto de dados em um conjunto de *clusters* disjuntos. Mais especificamente, eles tentam determinar o número de partições que potencializam uma dada função de critério. A função de critério pode enfatizar a estrutura local ou global dos dados e sua otimização é um processo iterativo. K-Means, por exemplo, realiza este tipo de agrupamento.

Agrupamento hierárquico: procede sucessivamente tanto fundindo agrupamentos maiores em outros menores, ou dividindo agrupamentos grandes. O resultado do algoritmo é uma árvore de *clusters*, denominado dendrograma, que

mostra como os *clusters* estão relacionados. Ao cortar o dendrograma em um nível desejado, um agrupamento dos itens de dados dentro de grupos disjuntos é obtido. O algoritmo DIANA (*DIvisive ANAlysis*), por exemplo, realiza agrupamento hierárquico.

Agrupamento baseado na densidade: A ideia principal deste tipo de agrupamento é agrupar objetos vizinhos de um conjunto de dados em clusters baseados em condições de densidade. Charmeleon é um exemplo de algoritmo que utiliza esta técnica.

Agrupamento baseado em grade: Este tipo de algoritmo são principalmente propostos para mineração espacial de dados. Sua principal característica é que eles estipulam o espaço em um número finito de células e então fazem todas as operações no espaço estipulado. A rede de Kohonen, foco deste trabalho, realiza este tipo de agrupamento.

Para cada uma das categorias citadas há uma grande variedade de subtipos e diferentes algoritmos para encontrar os *clusters*.

Clusterização é também frequentemente executada como passo preliminar no processo de mineração de dados, com os *clusters* resultantes sendo hostilizados como entradas em técnicas mais avançadas como, por exemplo, em Redes Neurais iguais aos Mapas Auto Organizáveis de Kohonen, que é baseado nos conceitos de redes neurais. A rede de Kohonen tem nós de entrada e saída, onde a camada de entrada (ou nós de entrada) tem um nó para cada atributo do registo, cada um ligado a todos os outros nós de saída (na camada de saída). Cada conexão é associada a um peso, que determina a posição do nó de saída correspondente. Assim, de acordo com um algoritmo que irá mudar os pesos corretamente, os nós de saída movem-se para formar os agrupamentos.

2.2 MEDIDAS DE AVALIAÇÃO DE AGRUPAMENTOS

O resultado de um agrupamento pode ser avaliado de duas formas: quantitativamente e qualitativamente. Uma avaliação quantitativa visa mostrar apenas a quantidade de elementos que foram corretamente agrupados. A avaliação qualitativa por sua vez, foca em descrever em que ponto um agrupamento revela os padrões escondidos nos dados. Isto obviamente depende da definição e da representação dos agrupamentos. O sucesso dessa abordagem depende também

da habilidade e experiência do avaliador e não pode ser avaliado sem o conhecimento profundo dos dados. A avaliação qualitativa é assim suscetível de não ser tão objetiva como uma descrição quantitativa. Ainda assim, muitas vezes é a forma mais interessante de avaliação, uma vez que, literalmente, mede o grau de precisão do resultado para o usuário.

Se, no entanto, não houver nenhum conhecimento a *priori* sobre o conjunto de dados disponível, então uma avaliação quantitativa pode ser a melhor opção.

Em geral, um método de agrupamento de dados é classificado como extrator de resultados de alta qualidade, se os agrupamentos possuem alta similaridade intra-cluster e baixa similaridade inter-cluster (SCHATZMANN, 2003).

Determinar o número adequado de agrupamentos é geralmente um processo baseado em duas principais etapas: Primeiramente, os dados são particionados em diferentes agrupamentos com diferentes valores. Em segundo lugar, medidas de qualidade, também chamadas de índices de validade relativa por Jain e Dubes (1988), são utilizadas para avaliar a adequação das partições obtidas.

Um tipo de índice, que está entre um dos melhores índices em um estudo comparativo apresntado por Vendramin et al (2009), é o Critério de Largura de Silhueta (ROUSSEEUW, 1987). Neste trabalho ainda é utilizado o índice de Largura de Silhueta Simplificado de Kaufman e Rousseeuw (1990), que busca obter melhor desempenho através da redução de cálculos ao comparar a distância inter-*cluster* através do centroide do outro agrupamento, em vez de fazer o mesmo cálculo da média para todos os registros do outro agrupamento. Outro ótimo índice a ser apresentado é o Davies-Bouldin (Davies e Bouldin, 1979), conhecido pela sua habilidade de medir a qualidade de um agrupamento.

2.2.1 Índice de Silhueta

As silhuetas são úteis quando as proximidades estão em uma escala proporcional (como no caso de distâncias Euclidianas) e quando se procura agrupamentos (*clusters*) compactos e claramente separados. De fato, a definição faz uso de médias de proximidade como no caso da média do relacionamento do grupo, que se sabe funcionar melhor em situações com *clusters* aproximadamente esféricos.

Para construir as silhuetas é preciso apenas de duas coisas: o agrupamento obtido (através de alguma técnica de agrupamento) e a coleção de todas as proximidades entre os registros (objetos). Para cada objeto i será extraído um valor $s(i)$, utilizado para calcular as dissimilaridades. De um objeto i qualquer do conjunto de dados, é definido A como o *cluster* ao qual ele foi atribuído. Para uma ilustração concreta, veja a figura 2. Quando o cluster A conter outros objetos além de i, iremos calcular:

$a(i)$ = dissimilaridade média de i para todos os outros objetos de A.

Na Figura 2, este é o comprimento médio de todas as linhas dentro de A. Considere agora qualquer cluster C que é diferente de A, e execute o seguinte cálculo:

$d(i, C)$ = dissimilaridade média de i para todos os objetos de C.

Na Figura 2, este é o comprimento médio de todas as linhas que vão de i para C. Depois de computar $d(i, C)$ para todos os clusters $C \neq A$, é selecionado o menor destes números o qual é definido como:

$$b(i) = \min_{C \neq A} d(i, C) \quad (1)$$

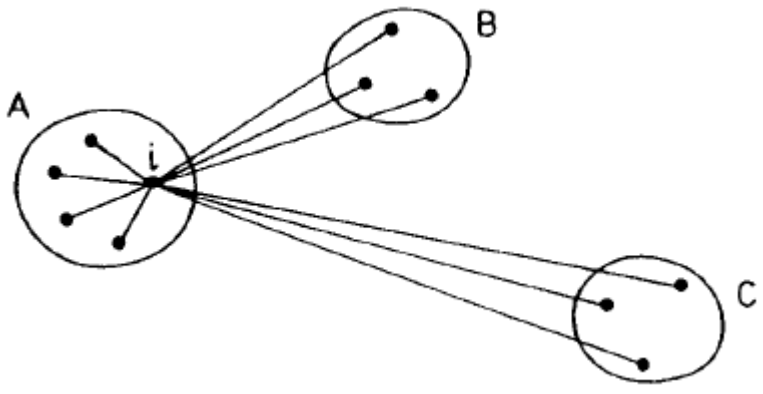

Figura 2 - Ilustração dos elementos envolvidos no cálculo de s(i), onde i pertence ao cluster A.
Fonte: Rousseeuw (1987).

O *cluster* B no qual este mínimo é obtido (isto é, $d(i,B) = b(i)$) será chamado de vizinho do objeto i. Este é tido como a segunda melhor escolha para o objeto i: se ele não pudesse ser alocado ao *cluster* A, que *cluster* B seria o competidor mais próximo? Na Figura 2, o *cluster* B parece de fato ser o mais próximo (na média) ao objeto i, quando o próprio A for descartado. É muito útil, portanto, saber o vizinho de cada objeto no conjunto de dados. Note que a construção de $b(i)$ depende da disponibilidade de outros agrupamentos além do A, então deve ser assumido que o número de k *clusters* é maior que 1 (um).

Os números $s(i)$ são obtidos pela combinação de $a(i)$ e $b(i)$ conforme é mostrado na equação 2:

$$s(i) = \begin{cases} 1 - \dfrac{a(i)}{b(i)} & se\ a(i) < b(i), \\ 0 & se\ a(i) = b(i), \\ \dfrac{b(i)}{a(i)} - 1 & se\ a(i) > b(i), \end{cases} \tag{2}$$

Sendo possível escrever isto na fórmula:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \tag{3}$$

Quando o cluster A contém apenas um único objeto não fica claro como $a(i)$ deveria ser definido, e então simplesmente é definido $s(i)$ igual a zero. Esta escolha é com certeza arbitrária, mas um valor de zero aparece ser o mais natural. Portanto, a partir das definições citadas se pode facilmente ver que $-1 \leq s(i) \leq 1$ para cada objeto i.

2.2.2 Índice de Silhueta Simplificado

O índice simplificado de Silhueta (*Silhouette*) é um índice interno, e como tal qualifica a relação entre um algoritmo de agrupamento e os dados agrupados por ele. Todo o processo de clusterização e validação usa nada mais que os próprios dados. Claramente, estes índices podem também ajudar a determinar o número de agrupamentos (PEDRYCZ, 2005 apud AMORIM, 2011, p.28).

Para explicar este índice, considere um objeto X_j pertencente ao agrupamento Ci. A dissimilaridade[5] entre X_j e o centróide[6] $\overline{X}_i$ do *cluster* Ci é indicada por $a(X_j)$, enquanto a dissimilaridade entre X_j e o centróide de outro *cluster* C_l é chamada $d(X_j, \overline{X}_i)$. Após computar $d(X_j, \overline{X}_i)$ para todos os *clusters* $C_l \neq C_i$, a menor distância guardada é chamada de $b(X_j)$, ou seja, $b(X_j) = \min(X_j, \overline{X}_i)$. Este valor representa a dissimilaridade entre X_j e seu *cluster* vizinho mais próximo. Uma vez que $a(X_j)$ e $b(X_j)$ tenham sido definidos, a Silhueta simplificada S(X_j) pode então ser definida por:

$$S(X_j) = \frac{b(X_j) - a(X_j)}{max\{a(X_j), b(X_j)\}} \quad (4)$$

É fácil verificar que 0<= $S(X_j)$ <= 1 se os objetos estão sempre atribuídos ao *cluster* com o protótipo mais próximo. Quanto mais alto o $S(X_j)$, melhor é a atribuição do objeto X_j para um dado *cluster*. Se $S(X_j)$ é igual ou está próximo de zero, então não está claro se o objeto deveria ter sido atribuído ao seu cluster atual ou a um vizinho. Finalmente, se um *cluster* C_i é um *singleton*, então $S(X_j)$ não está definido e a escolha mais natural é definir $S(X_j)$= 0 (KAUFMAN, ROUSSEEUW, 1990).

A Silhueta Simplificada de uma determinada coleção de objetos de dados é definida como a média aritmética dos valores individuais $S(X_j)$ calculado para cada objeto. Dessa forma, é possível calcular o índice tanto para *clusters* (C_i) individuais ou para toda uma partição (C) de dados, calculando a média de $S(X_j)$ sobre os objetos correspondentes, como a seguir nas fórmulas 5 e 6:

$$f(C_i) = \sum_{X_j \in C_i} \frac{S(X_j)}{|C_i|} \quad (5)$$

[5] Diferença entre dois objetos, obtida através da distância entre os vetores.
[6] Centro do agrupamento, obtido através da média de todos os objetos do grupo.

Onde |·| representa a cardinalidade de um conjunto.

$$f(C) = \sum_{j=1}^{N} \frac{S(X_j)}{N} \tag{6}$$

Quando usado como uma função objetiva para um procedimento de clusterização, que é precisamente o caso do presente trabalho, o valor máximo da Silhueta Simplificada irá indicar que o melhor resultado de agrupamento foi alcançado.

2.2.3 Índice Davies-Bouldin

Davies e Bouldin (1979) propuseram uma forma precisa de calcular a qualidade dos resultados de agrupamentos. Eles definiram C para ser o número de agrupamentos. $S_C(Q_k)$ é a distância intra-cluster do agrupamento Q_k, definido como a soma de todas as distâncias entre os membros do agrupamento X_i e o centróide C_k dividido pelo número de membros do agrupamento.

$$S_C = \frac{\sum_i \|X_i - C_k\|}{N_k} \tag{7}$$

A distância inter-*cluster* $d_{ce}(Q_k Q_l)$ entre dois agrupamentos Q_k e Q_lé definida como a distância entre dois centróides.

$$d_{ce} = d_{centroid} = \|C_k - C_l\| \tag{8}$$

O índice de Davies-Bouldin pode agora ser formulado como:

$$DB = \frac{1}{C} \sum_{k=1}^{C} \frac{max}{i \neq k} \left(\frac{S_C(Q_k) + S_C(Q_t)}{d_{ce}(Q_k, Q_t)} \right) \tag{9}$$

Deve-se mencionar que há várias maneiras fato de definir as distâncias intra-cluster e inter-cluster (SCHATZMANN, 2003).

O índice DB, proposto por Davies e Bouldin (1979) pode ser definido então como, uma função da razão da soma da dispersão dentro dos agrupamentos pela dispersão entre os agrupamentos (PAKHIRA et al., 2004).

2.3 REDES NEURAIS ARTIFICIAIS

O cérebro desempenha um papel muito especial entre todos os nossos órgãos: ao contrário de outros órgãos, ele não processa produtos metabólicos, mas sim uma "substância" que não se tornou objeto de investigação cientifica sistemática até o fim do século XVIII, ou seja, informação.

Até o final do século XVIII, o cérebro era considerado uma glândula cuja secreção era distribuída por todo o corpo ao longo das vias nervosas. A estrutura do cérebro como um complexo de tecidos entrelaçados de multiplas as células em rede trocando sinais entre si foi primeiramente reconhecido durante o século passado, principalmente através da pesquisa de Ramón y Cajal (1909).

Uma vez dissiminado este ponto de vista moderno, logo foi descoberto a especialização do cérebro em áreas, assim chamadas córtex, responsáveis por atividades específicas, tais como visão, audição, ou o movimento dos músculos.

Inúmeros experimentos e estudos têm estendido e refinado esta imagem em muitos aspectos, entre eles a invenção do computador que aprimorou a visão refinada do real "maquinário" com uma visão igualmente aprimorada da sua tarefa abstrata, que é o processamento de "dados". (RITTER et al, 1992).

2.3.1 Fundamentos Biológicos

Redes Neurais Artificiais (RNA) são modelos matemáticos inspirados nos princípios de funcionamento dos neurônios biológicos e na estrutura do cérebro. Esses modelos têm capacidade de adquirir, armazenar e utilizar conhecimentos

experimentais e buscam simular computacionalmente habilidades humanas tais como aprendizado, generalização, associação e abstração.

Com os avanços nas pesquisas na área da biologia, começou-se a entender o mecanismo natural de pensamento que torna possível que humanos aprendam com experiências anteriores.

Devido também aos avanços foi possível entender que os elementos de processamento fundamentais eram os neurônios, os quais estão presentes em grandes quantidades no cérebro, e são células especializadas interconectadas e permitem que o cérebro guarde informações na forma de padrões.

Cada neurônio consiste em um corpo celular, denominado soma, que contém o núcleo celular. Ramificando-se a partir do corpo celular, há uma serie de fibras chamadas dendritos e uma única fibra longa chamada axônio. Entre os neurônios existem conexões que são chamadas sinapses que recebem os sinais eletroquímicos enviados por outros neurônios, conforme pode ser observado na Figura 3.

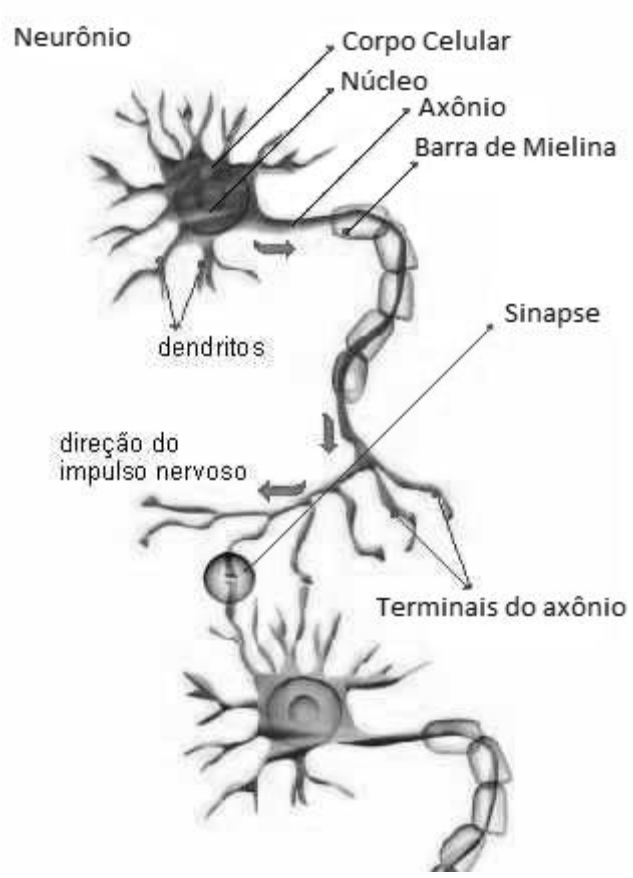

Figura 3 - Estrutura de um neurônio (esquemática).
Fonte: Adaptado de Portal São Francisco (2011).

Redes Neurais Artificiais foram inicialmente criadas para simular virtualmente o funcionamento do cérebro humano. Para este propósito os atuais computadores são extremamente rápidos na realização de cálculos, necessários para a aprendizagem, porém não é possível uma aproximação real de funcionamento e desempenho devido às limitações da tecnologia atual.

Os neurônios naturais usam os dendritos para receber informações de entrada de outros neurônios e combinam a informação de entrada, gerando uma resposta não linear resultado da reação quando algum limite é alcançado, a qual é enviada a outros neurônios usando o axônio. A figura 4 mostra um modelo de neurônio artificial utilizado na maioria das redes neurais.

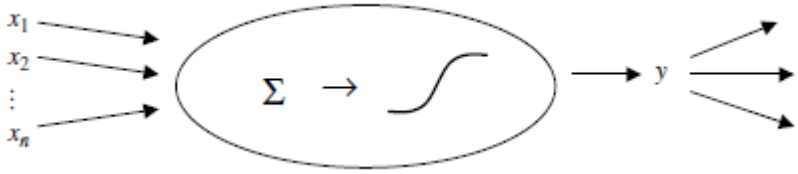

Figura 4 - Modelo de Neurônio Artificial
Fonte: Adaptado de Larose, Daniel T. (2005).

As entradas (X_i) são recebidas dos neurônios de entrada iniciais e combinados através de uma função de combinação (∑), que é então colocada em uma função de ativação (geralmente não linear) para produzir uma resposta de saída (y), que é depois canalizada adiante para outros neurônios.

O Quadro 1 resume a relação entre características e comportamentos de RNA com elementos da natureza.

Parâmetro	Cérebro	Computador
Material	Orgânico	**Metal e Plástico**
Velocidade	Milissegundos	**Nano segundos**
Tipo de Processamento	Paralelo	**Sequencial**
Armazenamento	Adaptativo	**Estático**
Controle de Processos	Distribuído	**Centralizado**
Numero de elementos processadores	10^{11} à 10^{14}	**10^5 à 10^6**
Ligações entre elementos processadores	**10.000**	**<10**

Quadro 1 - Comparativo entre o cérebro humano e o computador
Fonte: Tatibana e Kaetsu (2011).

Os principais objetivos de ferramentas de mineração de dados são a previsão e descrição. A previsão faz uso de variáveis de um banco de dados para prever valores desconhecidos ou futuros. A descrição é voltada para a busca de padrões, com objetivo de se descrever os dados e posteriormente apresentá-los de forma entendível para o usuário.

Existem vários algoritmos de Mineração de Dados para resolver diferentes problemas específicos. Estes algoritmos podem ser categorizados como: um algoritmo de associação, em que seu objetivo é achar todas as associações em que a presença de um conjunto de itens em uma transação implica em outros itens; algoritmos para classificação, que podem, por exemplo, desenvolver perfis de diferentes grupos em uma base de clientes; algoritmo para se trabalhar com padrões sequenciais, que visam identificar tipos de padrões sequenciais em restrições mínimas especificadas pelo usuário, ou ainda um algoritmo de agrupamento, que segmentam um banco de dados em subconjuntos ou grupos. Os algoritmos citados podem ainda ser associados em diferentes partes do processo de mineração a fim de se obter uma solução híbrida.

Classificação, Regressão, Previsão de Séries Temporais e Agrupamento, são exemplos de tarefas de Mineração de Dados que podem igualmente ser implementadas por métodos de redes neurais. Alguns modelos de Redes Neurais podem até mesmo ser aplicados em mais de um tipo de tarefa de Mineração.

Em geral, em aplicações de Mineração de Dados a camada de entrada do modelo neural, recebe os dados pré-processados de cada registro de uma base de dados. A rede processa esses dados, produzindo uma saída cuja natureza também varia em função da aplicação (GOLDSCHMIDT, PASSOS, 2005).

Em redes neurais com aprendizado supervisionado, a entrada corresponde aos atributos preditivos[7] enquanto a saída do modelo corresponde ao atributo objetivo[8] do problema. Assim sendo, o algoritmo de aprendizado pode estimar o erro, ou distância, entre a saída produzida pela rede e a saída desejada.

---

[7] Atributos de exemplo para a previsão, normalmente obtidos na fase de treinamento.
[8] Atributo que representa um valor desejável a ser alcançado durante o aprendizado.

Em função do erro calculado, o algoritmo ajusta os pesos das conexões da rede a fim de tornar a saída real tão próxima quanto seja possível da saída desejada. Modelos neurais deste tipo são muito uteis em geral para reconhecimento de padrões e em particular, para tarefas de mineração de dados que envolvem predição.

Por outro lado, redes neurais com aprendizado não supervisionado são adequadas para tarefas que envolvem descrição dos dados, como por exemplo, a tarefa de Clusterização. Nestes casos, não há saída desejada.

As redes neurais artificiais são compostas por nós ou unidades conectadas por vínculos orientados. Um vínculo de uma unidade j para uma unidade i serve para propagar a ativação aj desde j até i. Cada vínculo também tem um peso numérico $W_{j,i}$ associado a ele, o qual determina a intensidade e o sinal da conexão.

Cada unidade i primeiro calcula a soma ponderada de suas entradas:

$$in_i = \sum_{j=0}^{n} w_{j,i} a_i \tag{10}$$

Então aplica uma função de ativação g e a essa soma para derivar a saída:

$$in_i = g(in_i) = g\left(\sum_{j=0}^{n} w_{j,i} a_i\right) \tag{11}$$

A função de ativação g é projetada para atender duas aspirações: Primeiro, deseja-se que a unidade seja “ativa” (próxima de +1) quando as entradas “corretas” forem recebidas e “inativa” (próxima de 0) quando as entradas “erradas” forem recebidas. Em segundo lugar, a ativação precisa ser não linear, caso contrário, a rede neural inteira entrará em colapso, tornando-se uma função linear simples.

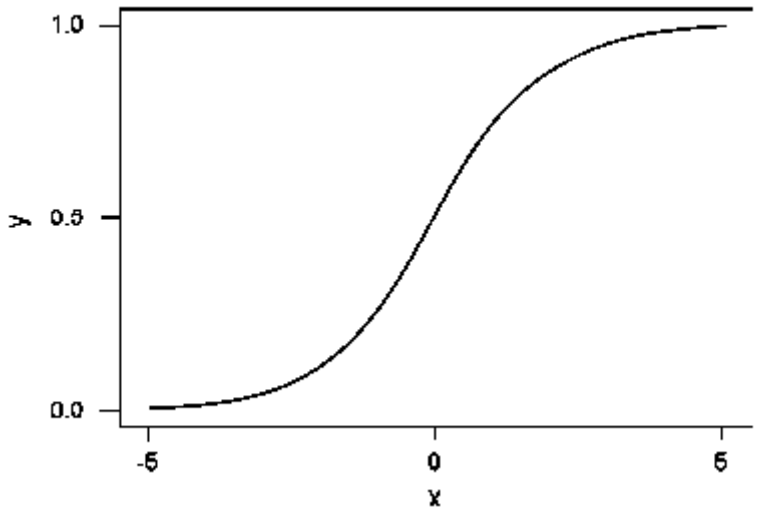

Figura 5 - Gráfico da função Sigmoide y = f (x) = 1/(1 + e−x).
Fonte: Ritter, H.; Martinez, T.; Schulten, K. (1992).

Como “candidata” há a função sigmóide, também conhecida como função logística ou de ativação, que tem como principal característica a de ser diferençável, e desta forma muito útil para algoritmos de aprendizagem de pesos.

Algumas das abordagens mais importantes modelos de redes neurais são o modelo *Perceptron*, o modelos *Hopfield*, e o algoritmo de retro propagação. Estes modelos surgiram no início das pesquisas, contribuindo para o desenvolvimento dos conceitos das redes neurais, e serviram de base para o posterior desenvolvimento dos Mapas Auto Organizáveis de Kohonen.

2.3.2 Perceptron

A rede *Perceptron* proposta por Rosenblatt (1958) é constituída de um número fixo de N elementos, cada um preenchido com um "padrão de entrada" por L canais. Cada um dos padrões de entrada é descrito por um vetor L componente de características $x = (x1, x2, \ldots, xL)T$, e pertence a uma das N "classes padrão".

A rede *Perceptron* por padrão aprende a correta classificação dos vetores de entrada usando exemplos de classificação conhecidos durante a "fase de treinamento", o que a classifica como uma rede de treinamento supervisionado.

Para a classificação de um padrão de entrada x, cada elemento r calcula uma saída binária y_r conforme a expressão:

$$y_r = \theta\left(\sum_{i=1} Lw_{ri}\, x_i\right) \tag{12}$$

Durante a fase de treinamento cada elemento ajusta seu coeficiente w_{ri} de forma que apenas irá reagir a padrões de entrada de sua classe C_r com um valor de saída y_r=1. Para isto ser possível, a existência de uma solução deve primeiro ser garantida, por exemplo, devem existir pesos (vetores) para que a função (12) resolva corretamente o problema de classificação.

2.3.3 Hopfield

É perceptível que a análise matemática dos sistemas com o retorno deste tipo acaba por ser muito mais difícil do que em casos lineares, e assim continuou por um longo tempo até simulações de computador para modelos de limites não lineares como nos neurônios de McCulloch, Pitts (1943).

Isso mudou em 1982, devido a importante ideia de Hopfield.

Se uma parte das linhas de saída for reenviada para a entrada da rede, a porção correspondente dos padrões de saída y pode contribuir para o padrão de entrada x. Um caso especialmente interessante surge para $y = x$, por exemplo, onde cada padrão de entrada é associado consigo mesmo como saída resultante (auto associação). Se for apresentado um padrão de entrada incompleto para um sistema recorrente em que tais pares de treinamento estão armazenados, então a princípio resultará correspondentemente em um padrão de saída incompleto.

No entanto, se a saída é realimentada, a porção intacta pode ser suficiente para a reconstrução de parte dos dados de entrada em falta. O sistema pode reagir ao padrão de entrada melhorado com uma saída melhor, que por sua vez reconstrói ainda mais o padrão de entrada, e assim por diante, até que finalmente o sistema acaba em um estado no qual o padrão de entrada está completamente restaurado.

Devido à realimentação, cada neurônio afeta a entrada de todos os outros neurônios da rede. O comportamento de um sistema deste tipo é geralmente muito difícil de analisar. No entanto, através da exploração de uma analogia à sistemas de interação de muitas partículas da física estatística, Hopfield foi capaz de caracterizar

o comportamento de uma classe interessante de tais sistemas em um modelo elegante.

2.3.4 Retro Propagação (Back-Propagation)

No modelo de Hopfield, cada neurônio é conectado a todos os outros neurônios. Assim, com relação a suas conexões, o modelo não tem "estrutura interna" e é "homogêneo". No entanto, as redes neurais geralmente são estruturadas.

Uma estrutura frequentemente encontrada resulta de se conectar várias camadas de neurônios em série. A primeira camada é geralmente reservada para os padrões de entrada. Cada neurônio desta camada envia conexões para cada neurônio da camada seguinte (figura 6). Isso continua até que a última camada seja alcançada, cuja atividade padrão constitui a saída.

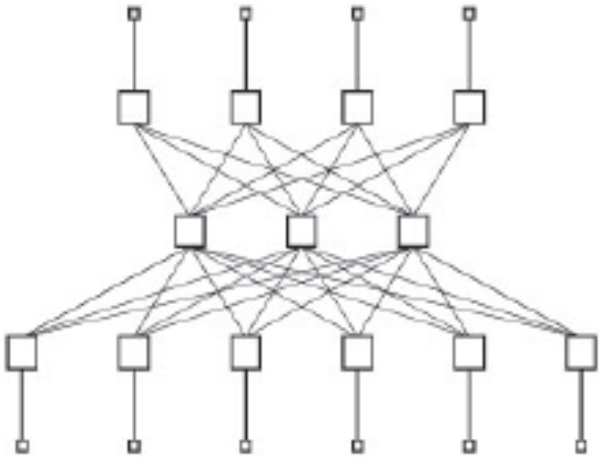

Figura 6 - Rede Neural de três camadas, cada camada envia conexões para a camada anterior. Fonte: Ritter, H.; Martinez, T.; Schulten, K. (1992).

Cada camada individual pode realizar uma transformação parcial do padrão de atividade da camada anterior. Para o *Perceptron* - essencialmente correspondentes a uma única camada - ocorre uma séria limitação das possíveis transformações entre a entrada e a saída. Assim, uma questão importante é a de como superar as limitações do *Perceptron*, ligando várias camadas em série e, portanto, concatenar suas transformações.

Em contraste ao *Perceptron* e às redes do tipo *Hopfield*, uma rede em camadas do tipo alimentação-progressiva (*feed-forward)* contém unidades de rede ocultas que não estão diretamente ligados às camadas de entrada ou saída.

Portanto, o estado de atividade desses neurônios não pode ser afetado diretamente pelo "mundo exterior", mas só pode ser influenciado indiretamente através do circuito interno da rede.

Devido às unidades ocultas serem apenas indiretamente afetados por sinais de entrada, surge o seguinte problema: se a tarefa dada foi mal realizada, não fica claro qual dos pesos é responsável pelo mau resultado e como eles devem ser alterados. Este problema é conhecido como o problema de atribuição de crédito e foi uma das razões que levou ao desaparecimento do *Perceptron* multicamadas e seus sucessores na década de 1960. O algoritmo de Retro Propagação (Back-Propagation - BP) é uma abordagem interessante para resolver este problema.

Em uma rede BP cada observação do conjunto de treinamento é processada através da rede, e um valor de saída é produzido a partir do nó de saída.

O valor de saída é então comparado com o valor real da variável-almejada para esta observação do conjunto de treinamento, e o erro (atual - saída) é calculado. Este erro de predição é análogo aos resíduos em modelos de regressão.

Para medir o quão bem as previsões de saída estão compatíveis com os atuais valores almejados, modelos de rede neural comumente usam a soma dos erros (*sum of squared error*) ao quadrado:

$$SSE = \sum_{registros} \sum_{nó\ saída} (atual - saída)^2 \tag{13}$$

Onde a previsão dos erros ao quadrado serão a soma de todos os nós de saída e sobre todos os registros no conjunto de treinamento.

O algoritmo BP obtém a previsão do erro (atual - saída) para um determinado registro e filtra o erro de volta através da rede, atribuindo a responsabilidade particionada do erro para as várias conexões. Os pesos sobre essas conexões são então ajustados para diminuir o erro, utilizando gradiente descendente.

2.4 MAPAS AUTO-ORGANIZÁVEIS

A rede neural de Mapas Auto Organizáveis (*Self-Organizing Maps* - SOM), foi proposta por Teuvo Kohonen (1990), como um modelo computacional de mapas topográficos corticais, tais como mapa somatosensório, mapa tonotópico, dentre outros. Mapas corticais são representações sensoriais e motoras existentes no córtex cerebral. Para cada sensação haveria um verdadeiro mapa da superfície sensível (corpo, retina, ouvido) na área cortical primária[9] correspondente (SILVA apud GATTASS, 2004).

Esta rede pertence à classe das Redes Neurais Não Supervisadas, conhecidas também como Competitivas ou Auto Organizáveis. Nesta classe de Redes Neurais as células vizinhas de uma Rede Neural competem pela ativação através de mútuas interações laterais, em vetores de pesos, e se desenvolvem de forma adaptativa em detectores específicos de diferentes padrões de entrada.

Em uma Rede Neural Auto Organizável o treinamento é não supervisionado, geralmente organizado em forma de uma competição entre os elementos processadores. Isto quer dizer que os dados são apresentados à rede SOM, mas a saída correta que corresponde aos dados não está especificada. Usando a rede SOM esses dados podem ser então agrupados de acordo com a similaridade de suas características.

Entre as principais aplicações das Redes Auto Organizáveis estão:

- Tarefa de Clusterização - Tarefa na qual os dados de entrada devem ser agrupados em conjuntos que agregam padrões semelhantes.
- Detecção de regularidades - Modelo em que o sistema deve extrair características relevantes dos padrões de entrada.

O método de aprendizado mais comum nas Redes Auto Organizáveis é denominado aprendizado por competição (*Competitive Learning*), que consiste em uma forma de aprendizado que divide o conjunto de padrões de entrada em grupos inerentes aos dados. Para tanto, este método considera, em sua abordagem mais

---

[9] Aquela que primeiro recebe a informação sensorial codificada vinda da superfície receptora.

simples, que os neurônios de saída da rede competem entre si, resultando em apenas um neurônio vencedor (com maior ativação).

A arquitetura típica de uma Rede de Kohonen é mostrada na Figura 7, e nela é possível observar a “camada de entrada”, que terá um nó de entrada para cada característica do atributo que estiver sendo apresentado à rede.

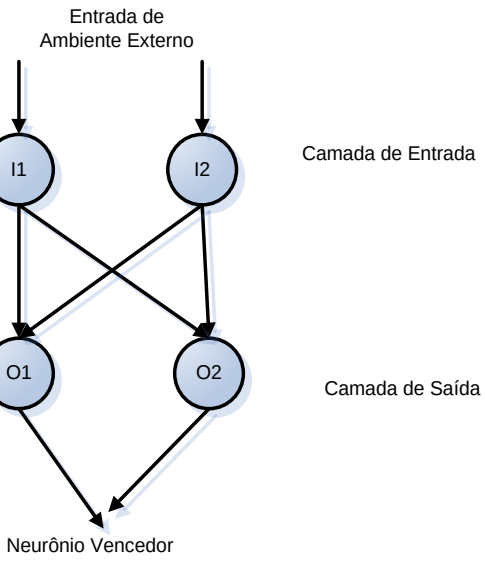

Figura 7 - Camadas de uma Rede Neural de Kohonen.
Fonte: Adaptado de Heaton (2005).

Basicamente, apenas uma célula ou grupo local de células por vez respondem ativamente à entrada corrente. A localização das respostas tende a se tornar ordenadas como se algum sistema significativo de coordenadas para diferentes padrões de entrada estivessem sendo criados sobre a rede. A localização espacial, ou coordenada de uma célula na rede, que basicamente é uma matriz de pesos, corresponde então ao domínio específico dos padrões de sinais de entrada.

Cada célula ou grupo de células locais age como um decodificador separado para a mesma entrada, ou seja, tanto a presença ou ausência de uma resposta ativa naquela região, e não necessariamente a transformação do sinal de entrada-saída nem a magnitude da resposta, que fornecem uma interpretação da informação de entrada (HEATON, 2005).

A velocidade de aprendizagem, especialmente quando são usados atalhos computacionais, pode ser aumentada para ordens de magnitude maior do que a de muitas outras redes neurais.

Em geral, Redes Neurais baseadas em Aprendizado Competitivo possuem as seguintes características:

- Topologia: Os neurônios são dispostos em uma única camada, estruturada em uma ou duas dimensões. Todos os neurônios desta camada recebem os valores dos padrões de entrada.
- Regra de Propagação: Produto escalar entre as entradas do elemento processador e os respectivos pesos associados às conexões subsequentes ao referido elemento.
- Função de ativação: Função Degrau, aplicada apenas ao neurônio vencedor, ou seja, somente o neurônio com maior potencial de ativação é submetido à função de ativação.
- Regra de Aprendizado: $\Delta W_{ik} = \alpha(X_i - W_{ik})$ Segundo essa regra, o aprendizado é não supervisionado, e ocorre de tal forma que a direção de atualização dos pesos minimiza a diferença entre o vetor de pesos vencedor e o vetor de entrada do elemento processador vencedor. O funcionamento dessa regra requer que os vetores X e W estejam normalizados. Essa regra de aprendizado tem como principal objetivo ajustar os pesos de tal forma que vetores de entrada similares ativem sempre o mesmo neurônio.

Como pode ser observada na Figura 8, a rede de Kohonen contém apenas duas camadas, uma delas é a camada de entrada, sendo esta responsável por apresentar os dados à rede. A segunda camada é a camada de saída, a qual contém o mapa e seus vetores de características, que são representados por pesos vetoriais os quais influenciarão na ativação dos neurônios da rede.

Assim como em outras redes neurais, estas entradas não processam informações por si só, mas simplesmente passam o atributo de entrada adiante, além disso, são redes completamente conectadas. Isto significa que cada nó da camada de entrada está conectado a todos os outros nós na camada seguinte, mas não estão conectados com outros nós na mesma camada. Cada ligação entre nós tem um peso associado a ele, que na inicialização tem um peso comum aleatoriamente atribuído com valor entre 0 e 1.

O ajuste destes pesos é a chave para o mecanismo de aprendizagem dos Mapas Auto Organizáveis. Os valores das variáveis precisam ser normalizados ou padronizados, para que certas variáveis não sobreponham outras no algoritmo de aprendizagem.

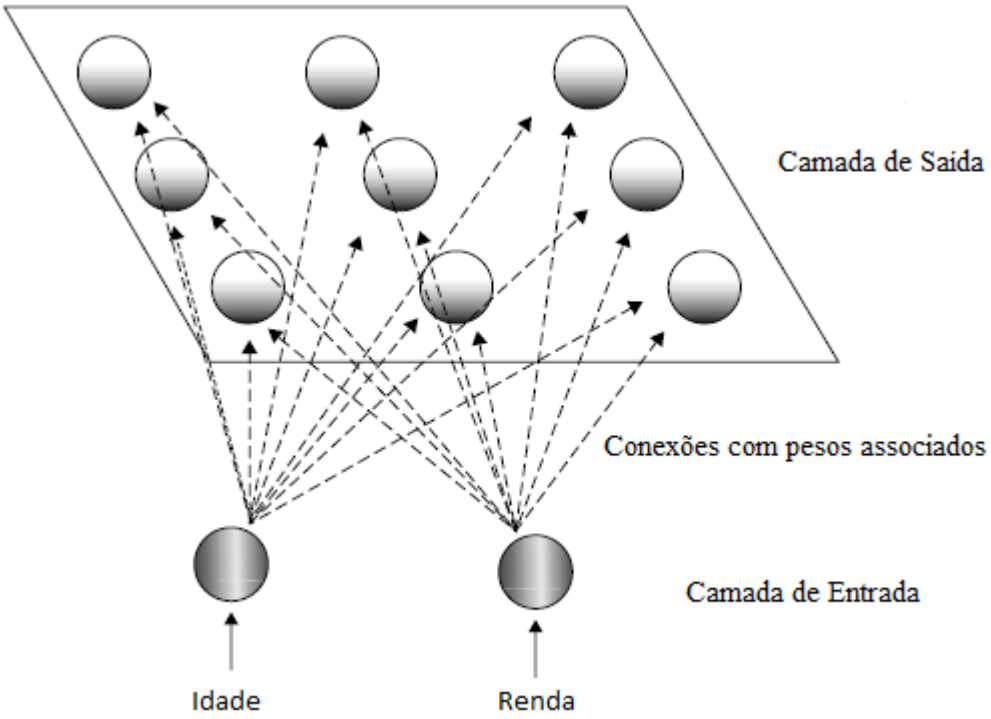

Figura 8 - Topologia de um mapa auto organizável simples.
Fonte: Adaptado de Larose, Daniel T. (2005).

A camada de saída é representada na forma de uma grade, que normalmente tem uma ou duas dimensões, e mais comumente tem a forma de um retângulo, porém outras formas, como a hexagonal ou até mesmo formatos como triangular, cilíndrico e toróide podem ser usadas. A camada de saída da Figura 8, por exemplo, tem um formato quadrado de 3x3 neurônios.

Para um dado registro, ou amostra, o valor específico de cada campo é enviado de um nó de entrada para os nós na camada de saída. Estes valores quando comparados aos pesos das conexões no mapa, retornarão aquele neurônio que apresentar maior semelhança com o registro atual. O nó de saída que tiver o melhor resultado na função de escolha do vencedor será designado então como o vencedor daquele registro.

Mapas Auto Organizáveis realizam três processos para a aprendizagem da rede: Competição, Cooperação e Adaptação:

> **Competição**: Os nós de saída competem entre si para produzir o melhor valor para uma função particular de “pontuação” ou aprendizagem, geralmente a distância euclidiana. Neste caso, o nó de saída que tem a menor distância euclidiana entre os campos de entrada e as conexões de pesos, seria então declarado o vencedor.
> **Cooperação:** O nó vencedor, portanto, se torna o centro de uma vizinhança de neurônios excitados. Este processo emula o comportamento de neurônios humanos, que são sensíveis à saída de outros neurônios em sua vizinhança imediata. Em Mapas Auto Organizados, todos os nós na vizinhança compartilham da “emoção" ou "recompensa" ganhas pelo nó vencedor, que se adapta. Assim, embora os nós na camada de saída não estarem diretamente ligados, eles tendem a compartilhar características comuns, devido a estes parâmetros de vizinhança.
> **Adaptação:** Os nós na vizinhança do nó vencedor participam na adaptação, ou seja, aprendem. Os pesos desses nós são ajustados de forma a melhorar ainda mais a função de “pontuação” ou aprendizagem. Em outras palavras, esses nós estarão assim tendo uma chance maior de ganhar na competição, mais uma vez, por um conjunto semelhante dos campos de valores. (LAROSE, 2005, p.165).

Os pesos devem ser inicializados de forma distribuída em relação à densidade dos vetores de entrada. Desta forma, pretende-se conseguir uma maior disponibilidade de pesos na região com mais amostras de vetores de entrada. A inicialização aleatória dos pesos pode gerar processadores inúteis, que nunca vençam a competição, assim como processadores que representem mais de uma classe, e, eventualmente, mais de um processador para representar uma única classe.

No mapa de Kohonen, onde o vencedor ganha a maioria, os neurônios da camada de saída são organizados em uma estrutura bidimensional, conforme mostra a Figura 8. A atualização dos pesos não é feita somente para o neurônio vencedor k, mas também para os neurônios pertencentes a determinada vizinhança $NEk(t)$, que deve ser reduzida com o tempo t.

2.5 CONSIDERAÇÕES

Tendo apresentado diversos temas neste capítulo a fim de situar o contexto em que se apresentam os algoritmos de redes neurais, principalmente as de aprendizado competitivo, este trabalho fará utilização das medidas de avaliação apresentadas na seção 2.2 deste trabalho, juntamente com a Rede Neural de Mapas Auto Organizáveis, a través utilização e extensão da biblioteca JKNNL.

3 METODOLOGIA

Neste capitulo serão apresentados os métodos utilizados para se alcançar os objetivos propostos no primeiro capítulos.

A Seção 3.1 demostrará as diversas formas de se criar e configurar uma nova rede SOM. A Seção 3.2 demonstrará como iniciar o aprendizado da rede SOM criada e configurada. A Seção 3.3 exibirá os métodos utilizados para se extrair conhecimento a partir de cada execução do SOM. A Seção 3.4 por fim mostra as configurações que foram definidas como padrão, bem como explica quais os experimentos a serem realizados.

Para este trabalho foi utilizado a biblioteca *Java Kohonen Neural Network Library* (JKNNL) (RYBARSKI e HABDANK, 2011), que é desenvolvida para uso em processos de mineração de dados e descoberta de conhecimentos em bases de dados. Esta biblioteca é um conjunto de classes e funções para projetar, treinar e calcular resultados de uma Rede de Mapas Auto Organizáveis (SOM). A biblioteca é escrita em Java, o que a torna altamente configurável e extensível. Existem definições de *containers* para os neurônios além das topologias de rede, como por exemplo, as topologias em matriz e hexagonal, que são de grande importância no processo de treinamento. Usando modelos de construção de objetos de função na forma de interfaces, a biblioteca completa o paradigma funcional, fornecendo uma ótima padronização de desenvolvimento e extensão da biblioteca.

Neurônios podem ser construídos pela parametrização das classes modelo, e então a rede pode ser criada. Algoritmos de treinamento podem ser criados de algumas partes, como a parametrização de objetos de função.

A biblioteca JKNNL tem a limitação de tratar somente dados numéricos, uma vez que a mesma utiliza modelos de objetos de entrada na forma de vetores de valores em ponto flutuante. Para bases de dados que trabalham com valores em caracteres, existe a possibilidade de se transformar os dados de forma a serem representados em formato numérico, porém a atual versão da biblioteca não conta nenhuma ferramenta ou algoritmo que auxilie nestas transformações.

3.1 CRIANDO A REDE

Para as entradas da rede, existe um modelo de classe de controle de dados, a qual é utilizada para persistir os dados disponíveis durante todo o processo de treinamento. Esta classe é denominada *LearningDataModel*, que por ser uma interface contém apenas especificações básicas para a definição de classes que implementarão os modelos de funções nela contidos.

As implementações que controlam o carregamento dos dados são feitos na classe *LearningData*, e entre os principais métodos e variáveis desta classe podem ser citados:

- *LearningData*: Método construtor que realiza a instanciação e carregamento dos dados, que são obtidos através de arquivos de texto plano.
- *getSplitter*: Customização feita para automaticamente identificar o dígito separador utilizado na base de dados escolhida para treinamento, podendo este conter um caractere de tabulação (\t), uma virgula (,) ou um ponto e virgula (;).
- *getData*: Responsável pela recuperação de todos os dados obtidos inicialmente.
- *toString*: Transforma o vetor em caracteres para ser impresso em um formato visualmente interpretável.
- *getDataSize*: Utilizado para se retornar a quantidade total de dados obtidos através da base de dados.

Para a criação e definição de modelos de redes é necessário escolher qual será o formato da topologia utilizada. As implementações de topologia são baseadas na interface *TopologyModel*, que contém os modelos das principais e fundamentais funções que uma topologia deve ter, como por exemplo:

- *getConnectedNeurons*: Implementará como serão retornados os neurônio diretamente ligados ao neurônio em questão.
- *getNeighbourhood*: Implementará como serão retornados os vizinhos do neurônio em questão.
- *getNumbersOfNeurons*: Número total de neurônios da rede.
- *getNeuronNumber*: Retornar o número do neurônio em uma determinada coordenada da rede.

As topologias implementadas presentes na biblioteca são a *MatrixTopology*, para representar os dados de forma retangular, e a *HexagonalTopology* que tem formato hexagonal.

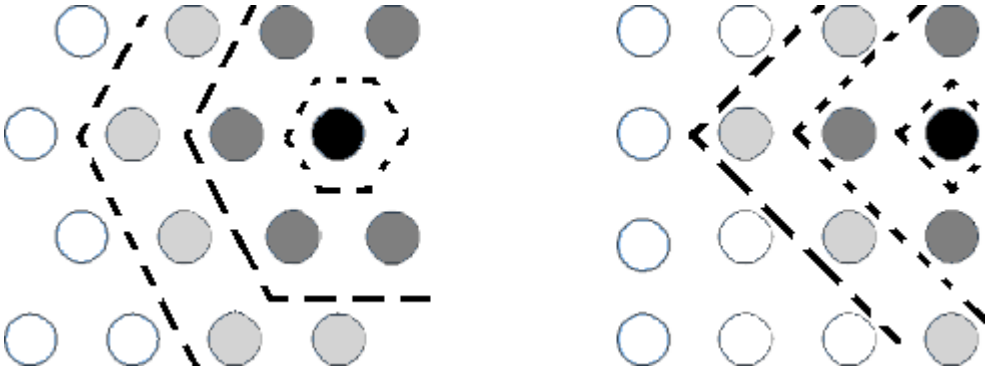

Figura 9 – Representação dos raios das topologias Hexagonal e em Matriz.
Fonte: Adaptado de Java Kohonen Neural Network Library (2011).

Tanto para instanciar uma rede retangular quanto uma hexagonal é preciso que sejam definidos os valores que definem o tamanho da rede, referentes à quantidade de linhas e quantidade de colunas, além do valor do raio de vizinhança utilizado. Um exemplo de ambas as topologias com tamanho de 4 linhas e 4 colunas pode ser visto na figura 9, onde em preto está o neurônio vencedor demarcado por traços ao seu redor estão as representações dos raios com valor igual a 1, onde a vizinhança está representada em cinza escuro e com raio igual a 2 a vizinhança representada em cinza claro.

Com a topologia corretamente definida, a rede propriamente dita pode ser criada, podendo ainda escolher se ela iniciará seus pesos com valores aleatórios ou se herdará os resultados de alguma execução anterior, de acordo com o método de declaração da classe *DefaultNetwork* selecionado.

Assim como para as classes anteriormente citadas, existe uma classe modelo que define as funções essenciais para a rede neural. *NetworkModel* é a interface que estabelece os padrões de desenvolvimento para redes, e que contém os padrões para criação das redes neurais.

A implementação de rede neural presente na biblioteca é a *DefaultNetwork*, que entre seus principais métodos e funcionalidades podem ser citados:

- *DefaultNetwork*(): Utilizado para definir uma nova rede, contém variações que permitem que seja definido por exemplo o tamanho dos vetores de entrada, os pesos máximos de seus atributos e o modelo de topologia

escolhido, ou ainda uma variação que permite que a rede seja recuperada de um arquivo, tornando-a uma rede com consciência para a determinada base de dados.

- *getNeuron*: Retorna um neurônio e seu vetor de pesos.
- *networkToFile*: Imprime a rede em um arquivo de saída caso seja desejado que a próxima execução da rede recupere os valores de um treinamento anterior em sua instanciação.

Tendo todos os elementos da rede neural devidamente definidos, é preciso agora especificar os elementos que controlarão o processo de aprendizado da rede neural.

3.2 INICIANDO O APRENDIZADO

Para iniciar o processo de treinamento da rede um dos principais itens para o treinamento, e que controlarão a porcentagem de atualização dos pesos da rede, e de seus vizinhos, é a definido como constante de aprendizado, que representa o índice de aprendizado da rede.

Para o melhor funcionamento da rede é recomendável que a rede tenha este índice decrescido durante o processo de aprendizado, e desta forma possa aos poucos se adaptar aos padrões de entrada. Esta atualização busca evitar tanto a generalização excessiva dos grupos no início do treinamento, quanto o excesso de dispersão ao final do treinamento.

A fim de garantir que um padrão seja definido para as classes que implementarão funções de aprendizagem, a biblioteca JKNNL conta com a interface *LearningFactorFunctionalModel*, contém apenas as especificações básicas para:

- *getValue* - Pegar o valor para a k iteração do aprendizado.
- *setParameters* - Definir os parâmetros essenciais da função a ser utilizada.
- *getParameters* - Retornar os valores vigentes da função especificada.

As funções de atualização da constante de aprendizado inicialmente implementadas na biblioteca utilizada são:

- *ConstantFunctionalFactor*: Define uma constante que não se altera durante o processo de aprendizagem, sua utilização não é recomendada.
- *ExponentionalFunctionFactor*: Utiliza uma função exponencial para calcular a constante de aprendizado de acordo com a iteração em que se encontra o aprendizado, a qual se atualizará seguindo o comportamento semelhante ao de um gráfico decrescente de função exponencial. Descrita pela função $n0 * exp(-c * k)$ onde:
    - $n0, c$ - são constantes.
    - k - número da iteração.
- *HiperbolicFunctionalFactor*: Utiliza uma função hiperbólica para calcular a constante de aprendizado de acordo com a iteração em que se encontra o aprendizado. Descrita pela função:

$$y = c1/(c2 + k) \quad (14)$$

Onde:

- $c1, c2$ - são constantes.
- k - número da iteração.

- *LinearFunctionalFactor*: Utiliza uma função linear para calcular a constante de aprendizado de acordo com a iteração em que se encontra o aprendizado. Descrita pela função:

$$y = (n0/maxIter) * (maxIter - iter) \quad (15)$$

Onde:

- y - valor de saída.
- $n0$ - fator máximo.
- $maxIter$ – número máximo de iterações.
- $iter$ - número da iteração.

- *GaussFunctionalFactor*: Utiliza uma função Gaussiana para calcular a constante de aprendizado de acordo com a iteração em que se encontra o aprendizado e o raio atual da vizinhança. Descrita pela função:

$$y = exp(-(k\text{^}2))/(2 * r\text{^}2)) \quad (16)$$

Onde:

- k - número da iteração.
- r - raio.

Durante o processo de treinamento, o algoritmo precisará de um modelo de medida que possa calcular a similaridade entre os registros. Em redes SOM esta similaridade é obtida através do cálculo da distância entre o registro de entrada e o vencedor da rede neural. Para a definição do neurônio vencedor será utilizada uma métrica que possa calcular a distâncias entre pares de vetores de pesos, e retornar o resultado do neurônio que tem a maior semelhança com o vetor de pesos do registro apresentado a rede.

Na biblioteca JKNNL encontra-se a implementação da interface modelo "*MetricModel*", que contem unicamente o modelo de um método para pegar o valor da distância entre dois vetores:

- *getDistance* - Modelo de método para retornar a distancia de pares de vetores.

A partir do modelo de medidas da biblioteca encontram-se implementadas as medidas *EuclidesMetric*, *MinkowskiMetric* e *CityBlockMetric*.

Na implementação da classe *EuclidesMetric* é utilizada a medida de distância euclidiana, que é definida por:

$$Medida\ Euclidiana = \sum \sqrt{(x_i - y_i)^2} \tag{17}$$

Onde:

- x_i – Primeiro vetor de entrada.
- y_i – Segundo vetor de entrada.

A classe *MinkowskiMetric*, utiliza a medida de distância Minkowski, que é definida por:

$$Medida\ Minkowski = \left\{\sum (x_i - y_i)^p\right\}^{(1/p)} \tag{18}$$

Onde:

- x_i – Primeiro vetor de entrada.
- y_i – Segundo vetor de entrada.
- p – Parâmetro que por padrão é iniciado com valor = 1.

A classe CityBlockMetric é a classe que contém a função mais simples das presentes na biblioteca JKNNL, considerando apenas a diferença das distâncias diretas. A medida utilizada para o cálculo de distância é a *City Block*, que é definida pela função:

$$CityBlock = \sum(x_i - y_i) \quad (19)$$

Onde:

- x_i – Primeiro vetor de entrada.
- y_i – Segundo vetor de entrada.

O processo de treinamento em si é o processo mais importante para a rede SOM. Quanto mais passagens forem aplicadas durante este processo, maior será o efeito do treinamento. Após ele, os agrupamentos resultantes da rede neural podem trazer muitas propriedades e características importantes dos dados. Por exemplo, como os dados podem ser generalizados, como os dados podem ser representados, quais agrupamentos podem ser encontrados nos dados, quais as semelhanças dos itens em cada grupo entre outras coisas.

Há dois algoritmos de treinamento para a Rede SOM, o primeiro é chamado de *Winner Takes All* (WTA) e o segundo *Winner Takes Most* (WTM).

No primeiro algoritmo, WTA, o vencedor ganha tudo, e apenas os pesos do neurônio vencedor são atualizados, de acordo com a fórmula:

$$\mathrm{w(k+1)} = \mathrm{w(k)} + \mathrm{n} * \mathrm{(x - w)} \quad (20)$$

Onde:

- $w(k+1)$ – Peso do neurônio em k +1 interações
- $w(k)$ – Peso do neurônio para a iteração k
- n – Valor do fator da função de aprendizagem para k interações
- x - aprendizado do vetor de dados
- w - peso do neurônio

Já no segundo, WTM, o vencedor ganha a maioria, e desta forma o neurônio vencedor e os neurônios que estão em nós vizinhos são atualizados de acordo com a fórmula:

$$w(k+1) = w(k) + n * N(i,x) * (x - w) \quad (21)$$

Onde:

- $w(k+1)$ - É o peso do neurônio na interação k+1.
- $w(k)$ - É o peso do neurônio para a interação k.
- n - Valor do fator da função de aprendizagem para k iterações.
- $N(i,x)$ - valor da função de vizinhança para o i-neurônio específico.
- x - aprendizado do vetor de dados.
- w - peso do neurônio.

A restrição quanto à atualização dos pesos apenas do neurônio vencedor, no caso do WTA, ou dele e de seus vizinhos, como no WTM, justifica-se porque, no caso de vetores normalizados, o processador vencedor é o que possui o vetor W mais próximo do vetor de entrada. A atualização dos pesos do processador vencedor faz com que o vetor W fique ainda mais próximo do vetor de entrada vigente. Desta forma, o vetor W fica cada vez mais representativo do grupo ao qual o padrão de entrada vigente pertence.

A constante da regra de aprendizado deve ser sempre inferior a 1, sendo adaptativo e decrescente com o tempo. Assim, os pesos são ajustados para representar a classe ou grupo de padrões semelhantes e não apenas um padrão de entrada específico.

Quando o algoritmo utilizado para treinamento da rede é o WTM, é preciso definir também qual a função que irá controlar o índice de atualização da vizinhança.

A classe *GaussNeighbourhoodFunction*, também presente por padrão na biblioteca JKNNL, define uma função para calcular a constante de atualização da vizinhança do neurônio vencedor.

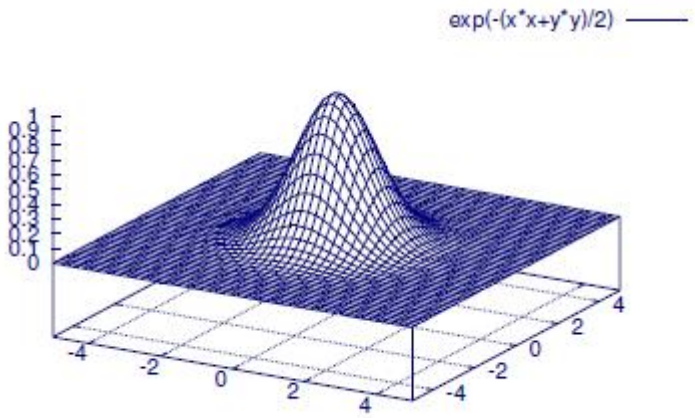

Figura 10 - Gráfico da função Gaussiana.
Fonte: Choe, Yoonsuck (2008).

Esta classe faz com que os neurônios vizinhos sejam também atualizados durante o processo de treinamento respeitando a definição do raio e a topologia utilizada. Na figura 10 é possível observar o gráfico resultante da aplicação da função Gaussiana em um plano em três dimensões.

3.3 EXTRAINDO CONHECIMENTO

A biblioteca JKNNL fornece todas as classes citadas até agora, algumas tiveram pequenas modificações durante o processo de alteração e ampliação do código fonte, e várias outras classes foram implementadas a fim de suprir a falta de algumas funcionalidades da biblioteca.

Algumas dessas funcionalidades são de extrema importância para se extrair conhecimento útil, e mesmo para definir uma base de métodos que possam no futuro serem reutilizados ou adaptados de acordo com a necessidade de utilização e ou extensão dos métodos da biblioteca.

Dentre um dos principais comportamentos que uma ferramenta de extração de conhecimento de bases de dados, principalmente se tratando de redes neurais como a SOM, é a criação de grupos dos registros presentes nas bases de dados. Com procedimentos para realizar este procedimento é possível que se extraiam exatamente que grupos que foram formados durante o treinamento, e quais membros pertencem a ele.

Para esta finalidade foi desenvolvida a classe *Clusters*, que implementa métodos e objetos para controlar e manter ao final da execução do treinamento, os agrupamentos formados. Esta classe tem como seu principal método implementado:

- *handleGroups* – Contém o algoritmo para definir e criar os agrupamentos de acordo com o número do neurônio vencedor, o número do registro pertencente ao grupo formado e o vetor de pesos referente aos pesos finais do registro na última iteração do total de n iterações.

A classe *Group* representa cada agrupamento formado propriamente dito. Nela serão mantidas informações do agrupamento como o número do neurônio vencedor, e comumente irá conter n elementos obtidos durante o treinamento. Dentre os principais métodos implementados por esta classe estão:

- *Group* – Construtor que instancia um novo grupo com o número do neurônio vencedor e seus registros pertencentes.
- *getCentroid* – Retorna um vetor de pesos que representa a média dos pesos de todos os registros pertencentes ao grupo.

A classe abstrata *InputData* que contém métodos e propriedades comuns às classes *Member* e *Centroid*, como por exemplo:

- *getWeights* – Responsável por manter os pesos dos registros.
- *getRecNumber* – Retorna o número do registro de acordo com a sua posição na base de dados utilizada para o treinamento.
- *getValue* – Retorna os pesos correspondentes do registro n.

A classe Member representa cada membro específico do agrupamento formado. O principal método implementado para controle dos membros de um agrupamento é:

- *Member* – Construtor da classe que instancia um novo membro com seu número de registo e seu vetor de pesos.

Com intuito de definir uma classe que pudesse controlar e extrair com precisão o centróide de cada grupo foi desenvolvida a classe *Centroid*. Esta classe visa definir um padrão de centróide para ser posteriormente utilizado nos algoritmos de validação de agrupamentos, e entre os principais métodos implementados para um centróide estão:

- *getCentroid* – Retorna a o conjunto de pesos que representam o centro do agrupamento.
- *increaseValue* – Controla o incremento dos valores de cada atributo de um centróide durante o cálculo do centróide.
- *setCentroid* – Define os pesos do centróide do agrupamento n.

Para os índices de validação, foi criada a classe abstrata *ValidationMetric*, a qual serve de base para a construção dos índices de validação e contém o método:

- *getMemberWeights* – Transforma um objeto genérico *InputData* em um *array* de *double*.

A classe do índice *DaviesBouldin* contém os métodos:

- *daviesBouldinIndex* - Implementação do algoritmo para calcular o índice de Davies-Bouldin
- *getAvgClusterDistance* – Retorna a distância média para um determinado agrupamento.

A classe do índice *Silhouette* contém:

- *silhouetteIndex* – Implementação do algoritmo para calcular o índice de Silhuetas.
- *getAvarageDissimilarity* – Retorna a dissimilaridade média para um determinado agrupamento.
- *getMinimunAvgDistance* – Retorna a menor dissimilaridade média para outro agrupamento.

Durante o processo de coleta de dados da base de dados, pode ser necessário que estes sejam normalizados (HEATON, 2005) a fim de reduzir os valores representativos dos atributos de cada registro contido na base de dados. Para este fim foi implementada a classe *Normalize*, que tem como principais métodos implementados:

- *Normalize* – Construtor da classe que irá predefinir o tamanho dos vetores de entrada a serem normalizados.
- *linearNormalization* – Função para retornar o valor do atributo de acordo com a normalização linear dos dados, transformando-os em uma faixa de 0 a 1, sendo 0 para o menor valor encontrado do atributo em questão em toda a base, e 1 o maior valor encontrado.

Para a o processo de aprendizagem é importante haver uma atualização da vizinhança, e para este propósito algoritmo WTM foi modificado a fim de realizar a atualização do raio da vizinhança com o decorrer do treinamento. Para alcançar tal funcionalidade foi definida uma instância de *LinearFunctionFactor* onde foram

passados os parâmetros do valor inicial do raio, pego através da já instanciada rede que é passada por parâmetro para inicialização do treinamento, e o número total de iterações que o treinamento irá realizar.

Para atenuar a mudança do valor retornado pela função linear no final do total de iterações, o valor foi tratado para retornar um valor inteiro que é arredondado em caso de números com casas decimais serem retornados. Alguns exemplos do tratamento dos valores podem ser visto na Tabela 1.

Valor da função linear	Valor inteiro utilizado
5.55	6
5.52	6
5.49	5
5.46	5

Tabela 1 – Exemplos de valores retornado pela função linear e o valor inteiro utilizado para o raio.
Fonte: Autoria própria.

Até agora, várias classes ajudantes foram definidas de modo a permitir que seja possível extrair mais informações do resultado do processo de treinamento. A partir delas, será possível utilizar os resultados de cada execução do algoritmo de treinamento para posterior comparação e averiguação de resultados.

Na próxima seção serão definidos os valores padrões para a execução do algoritmo de treinamento.

3.4 CONFIGURAÇÃO E EXPERIMENTOS

Tendo disponíveis todas as classes e funções necessárias para a execução e extração de conhecimento, é necessário escolher em que base de dados os algoritmos serão treinados. As bases escolhidas para análise foram a *Iris* e a *Wine*:

- ***Iris Dataset*:** considerado um dos melhores bancos de dados por ser encontrada em diversas literaturas a respeito de reconhecimento de

padrões, contém três classes de plantas Iris, sendo elas *Iris-Setosa, Iris-Versicolour* e *Iris-Virginica*, cada uma com 50 instâncias (33.3%), cada uma com quatro atributos numéricos (comprimento sepal em cm, largura de sepal em cm, comprimento de pétala em cm, e largura de pétala em cm).

- ***Wine Dataset*:** contém dados resultantes de uma análise química de vinhos produzidos na mesma região, na Itália, mas derivados a partir de três diferentes culturas. A análise determinou as quantidades de 13 componentes encontrados em cada uma das três classes de vinhos, sendo que a primeira classe contém 59 registros, a segunda tendo 71 registros e por fim a terceira com 48 registros.

Base de dados	Número de Registros	Número de Atributos
Iris Dataset	150	4
Wine Dataset	178	13

Tabela 2 - Dimensão dos dados de entrada
Fonte: Autoria própria.

Com a finalidade de se obter um padrão de configuração a ser utilizado para a biblioteca JKNNL durante toda a bateria de testes, os seguintes parâmetros foram estabelecidos:

1. Os valores referentes aos índices correspondem ao valor médio apresentado para o total de dez execuções do algoritmo de treinamento para cada uma das quantidades de *clusters* desejada.
2. Para cada execução foi definido a quantidade total de 2000 iterações do algoritmo.
3. Raio inicial da rede definido para ter sempre o mesmo valor da quantidade de colunas definidas.
4. Para as simulações foram utilizados mapas com os tamanhos de linhas e colunas iguais a: 1x2, 1x3, 2x2 e 3x3.
5. A topologia hexagonal foi definida como base para os experimentos.

6. A constante de aprendizado e o valor de decaimento do índice de aprendizado foram definidos em:
    a. 0.01 e 0.0001, para a base de dados *Iris*.
    b. 0.00007 e 0.001, para a base de dados *Wine*.
7. Todas as execuções foram feitas utilizando o algoritmo de aprendizado WTM, onde o neurônio vencedor e seus vizinhos são atualizados.
8. Execuções em que houve demasiada generalização, ou seja, em que todos os registros foram agrupados em um único grupo, foram desclassificadas para questões de cálculo das médias dos índices de validação de agrupamentos, uma vez que anulam a execução dos algoritmos que retornam os seus cálculos.

Para fins de comparação, foram executados experimentos com a aplicação de mineração de dados WEKA, da qual se puderam extrair resultados de algoritmos de agrupamentos de dados comumente usados em estudos referentes validação de agrupamentos. Entre os algoritmos escolhidos estão:

- O algoritmo COBWEB que é um algoritmo de agrupamento conceitual que organiza os dados de forma a maximizar a habilidade de inferência dos dados (FISHER, 1986).
- O algoritmo *k-Means* que é um dos mais simples e populares algoritmos de agrupamento de dados (ŠILIĆ et al, 2008).
- O algoritmo LVQ que é uma versão supervisionada do algoritmo de quantização vetorial de Borgelt (2004) que pode também ser usado quando há dados de entrada rotulados.
- Uma implementação do SOM que foi obtido através do gerenciador de pacotes da própria aplicação WEKA.

A configuração para cada um dos algoritmos citados ficou definida da seguinte forma para todos os experimentos:

Algoritmo *k-Means*:

- Número máximo de iterações: 2000
- Número de *clusters*: 2,3,4 e 9, de acordo com a bateria de testes em questão.
- Sementes: 10.

Algoritmo LVQ:

- Épocas: 2000.
- Taxa de aprendizado: 0.1.
- Número de *clusters*: 2,3,4 e 9, de acordo com a bateria de testes em questão.

Algoritmo COBWEB:

- Acuidade: 1.0, 0.7, 0.6 e 0.38 com a quantidade de *clusters* desejada.
- Corte: 0.0028209479177387815
- Sementes: 2

Algoritmo *SelfOrganizingMap* (WEKA):

- Taxa de aprendizado: 0.1.
- Tamanho do Mapa: 2x1, 3x1, 2x2 e 3x3.
- Épocas de Ordenação: 2000.
- Épocas de Convergência: 2000.

4 RESULTADOS

Neste capitulo serão apresentados os resultados alcançados a partir da realização dos experimentos definidos no capítulo 3.

A Seção 4.1 apresentará os resultados referentes à base de dados *Iris Dataset*. A Seção 4.2 apresentará os resultados referentes à base de dados *Wine Dataset*. A Seção 4.3 apresenta por fim a eficiência computacional de cada método de validação implementado e utilizados nos experimentos.

Nos experimentos aqui apresentados, os algoritmos foram induzidos a produzir o número de agrupamentos propositalmente designados de forma que fosse possível realizar a análise da qualidade dos agrupamentos de cada algoritmo através dos resultados obtidos das medidas de Silhuetas Simplificadas, Silhuetas padrão e *Davies-Bouldin*, e também a porcentagem referente ao erro médio (E.M.) de cada um dos resultados dos treinamentos.

As bases de dados *Iris* e *Wine* contém três classes cada uma, no entanto elas apresentam características muito diferentes, mesmo apesar de as duas bases terem a mesma quantidades de classes. Na base de dados *Iris* somente a primeira classe (*Iris-Setosa*) é linearmente separável das outras duas classes (*Iris-versicolor, Iris-virginica*) (FISHER, 1991). A base de dados *Wine* apresenta um problema de estruturas de classes "bem representadas" sendo desta forma uma base de dados adequada para testar novos classificadores (FORINA et al, 1991).

Os resultados foram obtidos através da média das execuções de cada um dos algoritmos selecionados, que tiveram suas configurações previamente definidas de forma a tornar compatível a comparação entre todos eles. Com o objetivo de extrair resultados próximos da realidade de cada base de dados, os algoritmos foram então induzidos a extraírem 2,3,4 e 9 *clusters* para ambas as bases de dados.

Para o melhor entendimento a tabela 3 apresenta os valores obtidos através da média do valor dos agrupamentos resultantes, além de trazer os valores médios globais dos índices calculados de cada um dos grupos originalmente pertencentes às bases de dados. Esta tabela exibe de forma básica os índices calculados para cada agrupamento seguindo a definição padrão das bases de dados utilizadas.

Índices de Validação		Base de Dados Iris	Wine
Silhueta Simplificada	Classe - 1	0.57829	0.35353
	Classe - 2	0.31582	0.28344
	Classe - 3	0.17951	0.41789
	Média Global	0.35788	0.35162
Silhueta	Classe - 1	0.79233	0.41407
	Classe - 2	0.43337	0.15925
	Classe - 3	0.33050	0.40578
	Média Global	0.51873	0.32637
Davies-Bouldin	Classe - 1	0.27754	0.74587
	Classe - 2	0.88403	0.74587
	Classe - 3	0.88403	0.70739
	Média Global	0.68186	0.73304

Tabela 3 - Índices calculados dos agrupamentos originais.
Fonte: Autoria própria.

Como descrito no capítulo 2, as duas técnicas de validação de agrupamentos utilizadas neste trabalho diferem nos valores apresentados devido ao fato em que os índices de Silhuetas tem seus melhores resultados quando os valores são maximizados chegando o mais próximo de 1, e os valores dos índices de Davies-Bouldin por sua vez representam seus melhores resultados quando os resultados calculados tendem a zero.

4.1 RESULTADOS DA BASE DE DADOS IRIS

Os resultados extraídos para a base de dados Iris observados na tabela 3 mostram que, para todos os algoritmos executados no WEKA os melhores índices (destacados em cinza escuro) foram obtidos com a quantidade de agrupamentos igual a dois, pois de acordo com os índices extraídos resultou também em uma melhor média de qualidade de agrupamentos (0,78 para Silhueta Simplificada, 0,70 para Silhueta e 0,46 para Davies-Bouldin), onde os agrupamentos estão bem separados entre si, e ao mesmo tempo bem agrupados internamente.

Base de Dados	Exp.	Algoritmo	Grupos	Valores das Medidas de Avaliação*			E.M.
				Silhuetas Sim.	Silhuetas	DB	
Iris	1.1	SimpleKMeans	2	**0.80199**	**0.72746**	**0.42668**	**33.33% (50)**
	1.2	SelfOrganizingMap		0.78619	0.70635	0.46022	**33.33% (50)**
	1.3	LVQ		0.76023	0.67710	0.51698	**33.33% (50)**
	1.4	COBWEB		0.78619	0.70635	0.46022	**33.33% (50)**
	1.5	JKNNL		**0.80199**	**0.72746**	**0.42668**	**33.33% (50)**
	2.1	SimpleKMeans	3	**0.37928**	**0.57077**	**0.54925**	**11.33% (17)**
	2.2	SelfOrganizingMap		0.36725	0.52275	0.63389	**11.33% (17)**
	2.3	LVQ		0.37549	0.55914	0.58432	13.33% (20)
	2.4	COBWEB		0.35807	0.52860	0.55839	26.00% (39)
	2.5	JKNNL		0.31604	0.45564	0.67010	12.00% (18)
	3.1	SimpleKMeans	4	**0.24844**	**0.46347**	**0.56330**	29.33% (44)
	3.2	SelfOrganizingMap		**0.24844**	**0.46347**	**0.56330**	29.33% (44)
	3.3	LVQ		0.24674	0.46249	0.56922	28.00% (42)
	3.4	COBWEB		0.21629	0.36961	0.96811	**19.33% (29)**
	3.5	JKNNL		0.20949	0.42305	0.57129	**19.33% (29)**
	4.1	SimpleKMeans	9	0.07517	0.28981	0.56644	59.33% (89)
	4.2	SelfOrganizingMap		0.12559	**0.41568**	0.57194	40.66% (61)
	4.3	LVQ		0.12294	0.38829	0.54268	42.66% (64)
	4.4	COBWEB		0.09002	0.36149	**0.46592**	33.66% (51)
	4.5	JKNNL		**0.14673**	0.36525	0.81062	**9.33% (14)**

Tabela 4 - Resultados base de dados Iris
Fonte: Autoria própria.

Os resultados dos experimentos realizados com a JKNNL (1.5, 2.5, 3.5 e 4.5) mostram que devido a constante atualização dos neurônios vencedores e dos registros de entrada durante o processo de treinamento, o algoritmo se mostra condizente com os demais algoritmos, apresentando uma boa média de resultados finais, que inclusive se aproximam da média dos resultados no algoritmo SOM extraídos no WEKA.

Como ponderador dos valores obtidos dos índices de validação, é possível observar a taxa de erro médio (E.M.) dos algoritmos para cada uma das quantidades de agrupamentos predefinidas. Este valor serve de base para uma análise suplementar dos resultados, uma vez que a taxa representa a porcentagem de registros que foram erroneamente agrupados nos processos de treinamento de cada um dos algoritmos. Diferentemente dos valores obtidos através dos índices de

validação, este valor se baseia na classe original do registro em questão e, portanto, mostra efetivamente o E.M. resultante dos diferentes algoritmos de agrupamento.

Devido ao fato de a base de dados Iris ter originalmente três classes, os melhores resultados de E.M. extraídos pelo WEKA aparecem quando os algoritmos foram configurados para formarem três agrupamentos. O algoritmo utilizado na JKNNL segue esta lógica para agrupamentos de 2, 3 e 4 *clusters*, porém o melhor resultado de E.M. (9,33%) aparece quando o mapa é definido com nove neurônios.

Isto ocorre devido ao fato das propriedades do algoritmo SOM, o qual agrupa os registros baseando-se no menor valor de distâncias entre o registro e os neurônios da rede. Esta característica, aliada ao maior número de possíveis vencedores resulta em um uma taxa de erro significativamente menor, uma vez que as classes não são mais sobrepostas e os registros de cada classe em questão estão em sua maioria corretamente agrupados.

Dos resultados observados na tabela 3 é possível ainda observar que os menores valores de E.M. para 4 e 9 *clusters* são obtidos pelo algoritmo da JKNNL. Estes resultados realçam as propriedades adaptativas dos Mapas Auto Organizáveis, os quais trabalham melhor quando há maior possibilidade de escolha para o neurônio vencedor. Como consequência da maior variedade de escolha, cada um dos neurônios ficará mais especializado para determinado tipo de entrada. Esta especialização resulta na melhor identificação dos diferentes agrupamentos, onde mesmo havendo certa dispersão das classes originais da base de dados (i.e. maior quantidade de *clusters*), implica em menor taxa de erro na escolha do agrupamento do registro apresentado à rede. Ou seja, havendo maior possibilidade de escolha nos neurônios no mapa de entrada, o algoritmo SOM tem menor chance de errar a classe a qual o registro deveria ser atribuído.

O gráfico 1 mostra detalhadamente os valores específicos da JKNNL para a base de dados Iris, onde estão incluídos os valores dos índices de Silhuetas Simplificadas, Silhuetas, Davies-Bouldin e o Erro Médio (E.M.) que teve seus valores adaptados para a fração de 1/100 para melhor visualização.

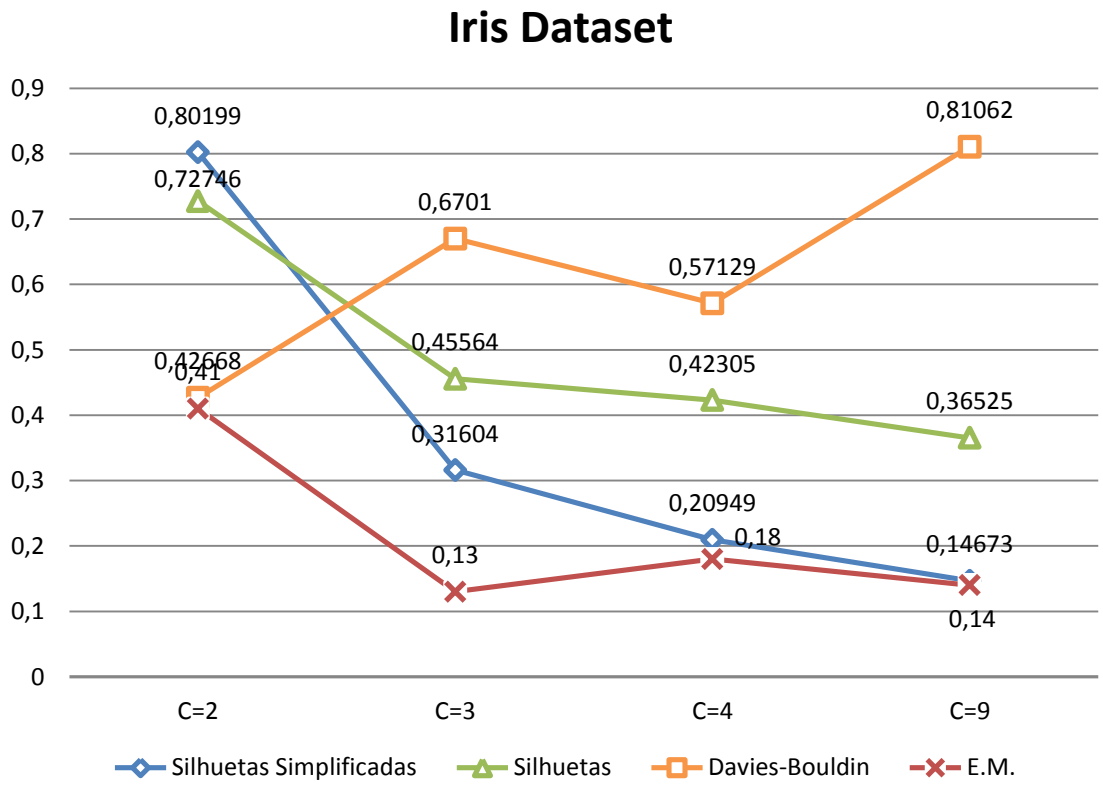

Gráfico 1 - Resultados da JKNNL para a bade de dados Iris.
Fonte: Autoria própria.

É possível perceber a evolução obtida no E.M. para as quantidades de agrupamentos formados. Nele é possível observar que para nove agrupamentos, ao contrário do que acontece para os outros algoritmos utilizados, o índice melhora perceptivelmente chegando quase ao mesmo valor obtido para três agrupamentos, que de acordo com a definição da base de dados, é a quantidade de agrupamentos corretos desta base de dados. O resultado para nove agrupamentos destacou ainda o resultado do índice de Silhuetas Simples, o qual teve o maior valor dentre os algoritmos utilizados. Para os demais índices porém os valore não apresentou valor satisfatório como, por exemplo, o índice de Davies-Bouldin que teve o segundo pior resultado geral, indicando certa sobreposição de classes.

4.2 RESULTADOS DA BASE DE DADOS WINE

Nos experimentos realizados utilizando o WEKA para a base de dados Wine, o algoritmo COBWEB mesmo apesar da configuração manual de seus parâmetros, não possibilita a seleção da quantidade de agrupamentos desejada. Por não apresentar resultados satisfatórios, chegando até a apresentar mais de cem agrupamentos como solução final nos seus treinamentos, o resultado das execuções deste algoritmo não trariam nenhum benefício para a análise dos resultados, e por este motivo teve seus valores desconsiderados na tabela 5.

Base de Dados	Exp.	Algoritmo	Grupos	Valores das Medidas de Avaliação*			E.M.
				Silhuetas Sim.	Silhuetas	DB	
Wine	5.1	SimpleKMeans	2	0.43689	0.31386	0.81857	39.88% (71)
	5.2	SelfOrganizingMap		0.44864	0.32460	0.79118	39.88% (71)
	5.3	LVQ		**0.46652**	**0.34287**	**0.75469**	**35.39% (63)**
	5.4	COBWEB		-	-	-	-
	5.5	JKNNL		0.44457	0.32165	0.80399	41.01% (73)
	6.1	SimpleKMeans	3	**0.35872**	**0.32372**	**0.71249**	5.61% (10)
	6.2	SelfOrganizingMap		0.35560	0.32083	0.72307	**5.05% (9)**
	6.3	LVQ		0.25767	0.25767	1.33275	33.14% (59)
	6.4	COBWEB		-	-	-	-
	6.5	JKNNL		0.20662	0.17007	1.25457	13.48% (24)
	7.1	SimpleKMeans	4	0.24129	0.25662	0.89235	20.22% (36)
	7.2	SelfOrganizingMap		0.24487	**0.25331**	0.88717	**18.53 (33)**
	7.3	LVQ		**0.25065**	0.22721	0.90536	22.47% (40)
	7.4	COBWEB		-	-	-	-
	7.5	JKNNL		0.19348	0.19775	**0.68857**	**18.53% (33)**
	8.1	SimpleKMeans	9	0.17278	0.24397	0.71899	50.00% (89)
	8.2	SelfOrganizingMap		0.15763	0.19717	0.916471	51.12% (91)
	8.3	LVQ		**0.20768**	**0.24846**	**0.71673**	44.94% (80)
	8.4	COBWEB		-	-	-	-
	8.5	JKNNL		**0.15398**	**0.20927**	**0.97891**	**14.04% (25)**

Tabela 5 - Resultados base de dados Wine.
Fonte: Autoria própria.

Assim como para a base de dados Iris, e diferentemente do que se encontra na descrição original da base de dados (FORINA et al, 1991), a média dos valores dos índices extraídos a partir do WEKA indica que os agrupamentos formados por dois clusters obtiveram os melhores valores (extraídos a partir da média de todos valores encontrados) para os algoritmos utilizados no experimento (0,45 para Silhueta Simplificada, 0,32 para Silhueta e 0,78 para Davies-Bouldin). Ao contrário dos resultados da base de dados Iris, entretanto, os valores extraídos do WEKA para a base de dados Wine não apresentam uma diferença muito acentuada dos valores médios comparando-se os resultados para dois ou três clusters. Isto claramente se deve à forma como os valores estão distribuídos na base de dados.

Os melhores valores gerais foram obtidos nos experimentos 5.3 e 6.1, onde o algoritmo LVQ obteve os melhores índices de silhuetas para dois clusters, e o algoritmo JKNNL obteve o melhor índice de Davies-Bouldin para quatro clusters. Isto indica que para o LVQ, os agrupamentos formados tiveram uma boa silhueta média geral, não entrelaçando muito as classes em seu treinamento com dois agrupamentos. Já o algoritmo da JKNNL teve uma ótima união intra-cluster e um bom espaçamento dos demais agrupamentos quando trabalhando com os quatro agrupamentos originais da base de dados *Wine*.

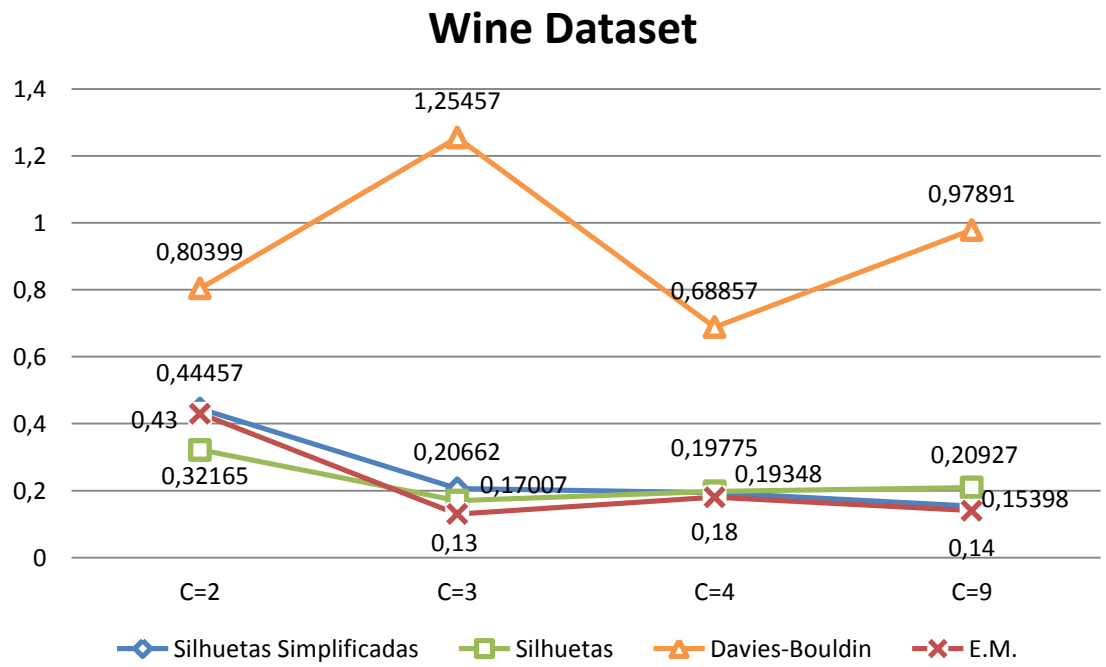

Gráfico 2 - Resultados da JKNNL para a bade de dados Wine.
Fonte: Autoria própria.

O erro médio, mais uma vez seguiu o comportamento anteriormente visto obteve seu melhor valor trabalhando com três agrupamentos, e em comparação com os demais algoritmos, ele se mostra muito superior na taxa de erro quando é utilizado um mapa com maior proporção, que permite à rede neural se adaptar e atualizar durante o processo de treinamento, especializando cada um dos seus neurônios e aprendendo a reconhecê-los melhor com o passar das épocas.

4.3 EFICIÊNCIA COMPUTACIONAL DOS ÍNDICES

Uma das motivações para a aplicação de diferentes técnicas de avaliação de agrupamentos foi além das características próprias de cada índice, o possível ganho de *performance* na avaliação dos resultados do agrupamento. O gráfico Z a seguir apresenta os valores do tempo de resposta, isto é, o tempo gasto para se criar a instância do algoritmo de validação de agrupamentos em questão e retornar o resultado calculado para todos os registros agrupados durante o processo de treinamento:

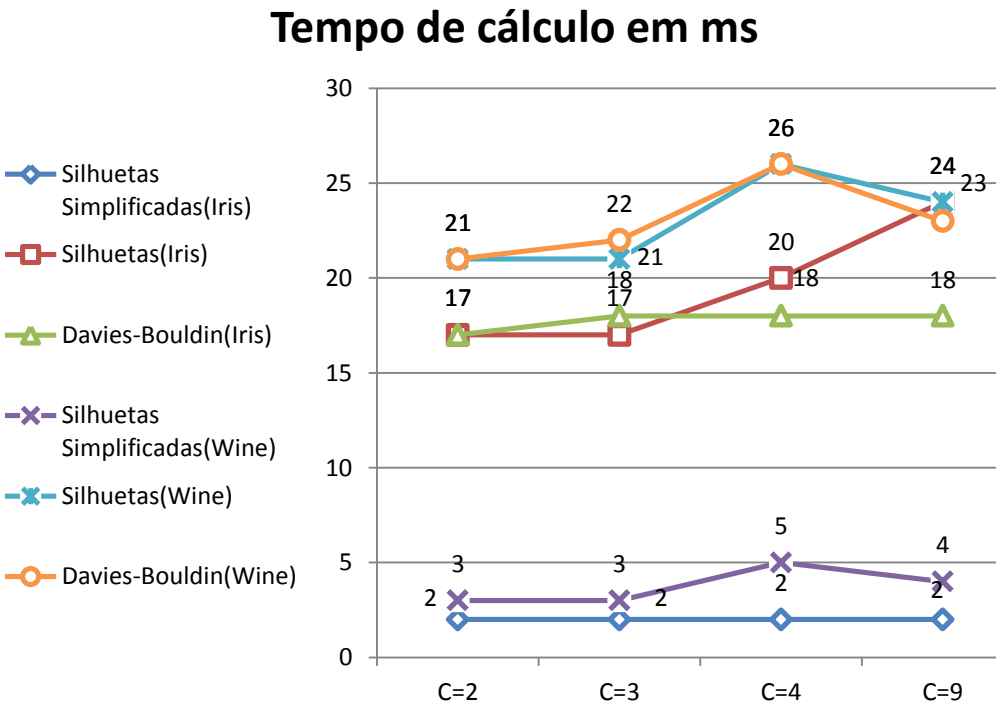

Gráfico 3 - Tempo gasto em milissegundos para o calculo do índice.
Fonte: Autoria própria.

O gráfico mostra claramente a vantagem computacional trazida pelo índice de Silhueta Simplificada, o qual calcula a distância intra-*cluster* através do centróide, em vez de realizar o cálculo para todos os itens do outro agrupamento em questão.

As Tabelas 6 e 7 apresentam os resultados encontrados em artigos oficialmente publicados que fazem referência tanto para as bases de dados *Iris* e *Wine*, quanto aos índices utilizados neste trabalho, fornecendo assim uma base para comparação da precisão do algoritmo SOM e também para validação dos próprios índices implementados.

Autor	Índice de Validação	Base de dados Iris			
		2 Classes	3 Classes	4 Classes	9 Classes
Saitta et al.	Davies-Bouldin	0.687	0.716	0.739	0.752
	Silhuetas	0.771	0.673	0.597	0.535
Pavan et al.	Davies-Bouldin	-	0.463	-	-
	Silhuetas	-	0.804	-	-
Kennan e Chung	Silhuetas	-	0.620	-	-
Liu et al.*	Silhuetas	0.62	0.555	0.445	0.250

Tabela 6 - Índices em artigos oficialmente publicados: Iris.
Fonte: Autoria própria.

Autor	Índice de Validação	Base de dados Wine			
		2 Classes	3 Classes	4 Classes	9 Classes
Saitta et al.	Davies-Bouldin	1.505	1.257	1.499	1.490
	Silhuetas	0.426	0.451	0.416	0.340
Puma-Villanueva*	Davies-Bouldin	0.110	0.001	0.150	0.400
	Silhuetas	0.999	0.899	0.750	0.050
Pavan et al.	Silhuetas	-	0.694	-	-
	Davies-Bouldin	-	0.569	-	-
Kannan e Chung	Silhuetas	-	0.633	-	-

Tabela 7 - Índices em artigos oficialmente publicados: Wine.
Fonte: Autoria própria.

Em Saitta et al. (2007) é apresentado um índice específico criado por ele, chamado de *Score Function* (SF), este índice é comparado com diversos índices, dentre os quais os índices de Silhuetas e Davies-Bouldin.

Pavan et al. (2011) apresenta um algoritmo de agrupamentos de dados automático, chamado de *Automatic Merging for Optimal Clusters* (AMOC), o qual é

uma extensão do algoritmo de *k-Means*, e faz comparação entre diversos algoritmos de validação e agrupamento de dados.

Em Puma-Villanueva (2008), é encontrado um estudo inicial focado em índices de validação de agrupamentos, onde não é especificado o algoritmo usado.

Kannan e Chung (2009) apresentam outro algoritmo de agrupamento de dados chamado de *Extended Hyper Tangent Version of Fuzzy C-Means* (EHTVFCM), o qual traz o resultado da medida de silhueta apenas para três agrupamentos.

Liu et al. Apresenta também um algoritmo que é *denominado Multi-Objective k-Means Genetic Algorithm* (MOKGA), e em seu estudo apresenta os resultados de vários algoritmos de validação de agrupamentos calculando a eficiência do algoritmo proposto.

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou a utilização de Redes Neurais de Mapas Auto Organizáveis na tarefa de mineração e agrupamento de dados, comparando os resultados obtidos através da extensão da biblioteca JKNNL, com outros algoritmos presentes na popular ferramenta de mineração de dados WEKA. Para a verificação da eficácia, os resultados obtidos foram também comparados com outros trabalhos oficialmente publicados, e que foram selecionados devido ao fato de apresentarem novas e mais eficientes técnicas de mineração e agrupamento de dados.

A biblioteca JKNNL faz parte de uma classe de redes neurais de aprendizado não supervisionado, que se mostram muito eficaz na descoberta de dados não rotulados, especialmente se mais neurônios estão disponíveis durante o treinamento, que é quando são capazes de conseguir níveis de precisão satisfatórios.

Para se alcançar principalmente os objetivos de análise e comparação dos resultados, diversas implementações na biblioteca JKNNL original se tornaram necessárias. As alterações implementadas tornaram a biblioteca apta a guardar os agrupamentos formados durante o processo de treinamento, e a posse dos dados resultantes deste treinamento permitiu que outros algoritmos implementados para o cálculo e validação dos agrupamentos pudessem então extrair de forma mensurável a qualidade dos agrupamentos que foram formados.

Os resultados de cada algoritmo de validação de agrupamentos pode ser também avaliado nos quesitos referentes a validade e eficiência dos próprios índices, uma vez que tanto os resultados obtidos através da biblioteca JKNNLX (FERREIRA, 2012), quanto os resultados em artigos ou os extraídos através do WEKA, apresentaram concordância de valores, não destoando em seus resultados, e confirmando assim a eficácia de seus métodos.

Os agrupamentos formados durante a fase de experimentos se mostraram condizentes com as demais técnicas utilizadas e comparadas, atestando desta

forma a validade e eficiência do algoritmo SOM no processo de agrupamento de dados.

Os resultados obtidos, entretanto apontaram a tendência de agrupamentos formados por apenas dois grupos, mesmo apesar de as bases de dados *Iris* e *Wine* apresentarem originalmente três classes cada uma.

Esta tendência foi também observada na maioria das outras técnicas comparadas, reforçando desta forma os resultados obtidos neste trabalho e indicando que devido a semelhança das classes presentes nas bases de dados, o número que representa mais fielmente os dados presentes em cada base de dados, não seria necessariamente o mesmo número de classes oficialmente informado.

A avaliação dos resultados de cada bateria pôde então confirmar estes resultados, sendo que a regularidade dos valores dos índices extraídos dos agrupamentos apontou a quantidade de agrupamentos que melhor representariam os dados.

É importante citar, entretanto, que nos resultados obtidos tornou-se visível a capacidade de comparação e aprendizado dos Mapas Auto Organizáveis, que conseguem manter uma baixa taxa de erro quando se trabalha com mapas maiores que o formato padrão dos dados apresentados à rede. Pode-se também analisar a vantagem e economia computacional alcançada ao utilizar um método simplificado de um índice de muita credibilidade na comunidade científica, o qual pode trazer grandes benefícios ao se trabalhar com sistemas em tempo real que necessitam responder a um estímulo no menor tempo possível.

A partir do balanço dos resultados obtidos é possível afirmar que a maior contribuição deste trabalho não parte da premissa de que a biblioteca utilizada superou todas as demais técnicas comparadas, mas sim do fato que ela atende satisfatoriamente os requisitos de um bom algoritmo para mineração de dados.

Como resultado das diversas implementações na biblioteca JKNNL, originou-se a biblioteca JKNNLX (Java Kohonen Neural Network Library eXtended), uma ferramenta para análise de agrupamentos muito útil ao trazer informações importantes da qualidade dos agrupamentos formados, ajudando no processo de obtenção de conhecimento a partir dos índices qualitativos gerados.

5.2 TRABALHOS FUTUROS

Como trabalhos futuros perceptivelmente possíveis de se alcançar com as implementações já presentes na biblioteca, se destacam a possibilidade de unir a boa precisão do SOM em mapas de maior dimensão, com um método de correção ou limitação da quantidade de agrupamentos totais. Podendo tornar o algoritmo SOM em um algoritmo “semi-supervisionado” através da união de agrupamentos muito próximos. Esta análise pode ser baseada nos já implementados índices de validação, os quais retornam valores “pobres” para agrupamentos muito parecidos, mas que se encontram separados.

REFERÊNCIAS

AMORIM, R. C., **Learning feature weights for K-Means clustering using the Minkowski metric**. Department of Computer Science and Information Systems, Birkbeck, University of London. 2011.

BLINKOV, S.M; GLEZER, I.I. **The Human Brain in Figures and Tables**: A Quantitative Handbook, New York: Plenum Press, 1968.

BORGELT, C. et al**. Learning vector quantization: cluster size and cluster number**. School of Computer Science, University of Magdeburg, Magdeburg, Germany. 2004.

CHOE, Yoonsuck. **Haykin Chapter 9**: Self-Organizing Maps. 2008. Disponível em: <http://courses.cs.tamu.edu/choe/08spring/lectures/slide09.pdf>. Acesso em: 22 out. 2011.

DAVIES, D.L.; BOULDIN, D.W.; **A cluster separation measure**. IEEE Transactions on Pattern Recognition and Machine Intelligence, Vol. 1, No. 2, 1979, pp. 224-227.

FACELI, Katti; CARVALHO, André C.P.L.F.; SOUTO, Marcílio C.P.; **Validação de Algoritmos de Agrupamento**. São Carlos: Instituto de Ciências Matemáticas e de Computação, RELATÓRIO TÉCNICO DO ICMC, 2005.

FAYYAD , Usama M.; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. **From Data Mining to Knowledge Discovery**: a Overview. In: Advances in Knowledge Discovery and Data Mining. Usama M. Fayyad, et.al. 9Editors). AAAI Press/MIT Press, 1996.

FERREIRA, André L. **JKNNLX** - Self Organizing Maps with Clustering Evaluation. Disponível em: <http://code.google.com/p/java-kohonen-neural-network-extended/downloads/list>. Acesso em 20 mai. 2012.

FISHER, Douglas H. **Knowledge Acquisition Via Incremental Conceptual Clustering**. Irvine Computational Intelligence Project, Department of Information and Computer Science, University of California, Irvine, California 92717, U.S.A. 1986.

FISHER, R.A. **The use of multiple measurements in taxonomic problems**. Annual Eugenics, 7, Part II, 179-188. 1936. Encontrado em: <http://archive.ics.uci.edu/ml/machine-learning-databases/iris/>.

FORINA, M. et al**. PARVUS - An Extendible Package for Data Exploration, Classification and Correlation**. Institute of Pharmaceutical and Food Analysis and Technologies. Genoa, Italy. 1991. Encontrado em: <http://archive.ics.uci.edu/ml/machine-learning-databases/wine/>.

GOLDSCHMIDT, R; Passos, E. **Data Mining**: Um guia prático. Elsevier, 2005.

GONÇALVES, A. L.; PACHECO, R. C. S.; MORALES, A. B. T.; **Utilização de Técnicas de Mineração de Dados em Bases de C&T**: Uma Análise Dos Grupos De Pesquisa No Brasil. Universidade Federal de Santa Catarina, Santa Catarina, Brasil. 2001.

HEATON, Jeff. **Introduction to Neural Networks with Java, 1st Edition**. Heaton Research, Inc. 2005.

HOLMES, Geoffrey; DONKIN, Andrew; WITTEN, Ian H. **WEKA: A Machine Learning Workbench**. Department of Computer Science. University of Waikato, Hamilton, New Zealand. 1994.

JAIN, A. K.; DUBES, R. C. **Algorithms for Clustering Data**, Prentice Hall, 1988.

JAIN, A. K.; MURTY, M.N.; FLYNN, P.J. **Data Clustering: A Review**, ACM Computing Reviews, Nov 1999.

KANNAN, S R; CHUNG, Pau-Choo.; **An Effective Objective Function of Fuzzy C-Means with an Effective Cluster Center Initialization**. Department of Electrical Engineering, National Cheng Kung University, Taiwan. 2009.

KARYPIS, G.; HAN, E.; KUMAR, V.; **CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling**. IEEE Computer 32(8): 68-75. 1999.

KAUFMAN, L.; ROUSSEEUW, P. **Finding Groups in Data: an Introduction to Cluster Analysis**. John Wiley & Sons (1990).

KOHONEN, T. **The self-organizing map**. Proceedings of the IEEE, v. 78, n. 9, p. 1464–1480. 1990.

LAROSE, Daniel T., **Discovering Knowledge in Data**: An Introduction to Data Mining, Wiley-Interscience, 2005.

LIU, Yimin; ÖZYER, Tansel; ALHAJJ, Reda; BARKER, Ken; **Integrating Multi-Objective Genetic Algorithm and Validity Analysis for Locating and Ranking Alternative Clustering**. Advanced Database Systems and Applications Lab, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada. 2004.

Machine Learning Repository, Center for Machine Learning and Intelligent Systems, University of California, Irvine, California, United States. Encontrado em <http://archive.ics.uci.edu/ml/about.html>.

MCCULLOCH, W.; PITTS, W. **A logical calculus of the ideas immanent in nervous activity**. Bulletin of Mathematical Biophysics, 1943.

MELSSEN *, Willem; WEHRENS, Ron; BUYDENS, Lutgarde. **Supervised Kohonen networks for classification problems, Institute for Molecules and Materials**. Elsevier Radboud University Nijmegen, The Netherlands, mar 2006. Disponível em: <http://www.cac.science.ru.nl/research/publications/PDFs/melssen2006.pdf>. Acesso em: 07, mai, 2011.

NAGI, J. et al. **Electrical Power Load Forecasting using Hybrid Self-Organizing Maps and Support Vector Machines**. The 2nd International Power Engineering and Optimization Conference (PEOCO 2008), Shah Alam, Selangor, MALAYSIA. 4-5 June 2008.

PAKHIRA, M.K.; BANDYOPADHYAY, S; MAULIK, U; **Validity index for crisp and fuzzy clusters**. In Proceedings of Pattern Recognition. 2004, 487-501.

PAVAN, K. K.; RAO, A.A.; RAO, A.V.D.; **An Automatic Clustering Technique for Optimal Clusters**, Department of Computer Applications, Rayapati Venkata Ranga Rao and Jagarlamudi Chadramouli College of Engineering, Guntur, India, 2011.

Portal São Francisco. **Sistema Nervoso**: Anatomia, Estrutura, Classificação: Neurônios. p.6. Disponível em <http://www.portalsaofrancisco.com.br/alfa/corpo-humano-sistema-nervoso/neuronios-6.php>. Acessado em: 27 Out. 2011.

PUMA-VILLANUEVA, W.J.; VON ZUBEN, F.J., **Índices de validação de agrupamentos**, Departamento de Engenharia de Computação e Automação Industrial (DCA). Faculdade de Engenharia Elétrica e de Computação (FEEC). Universidade Estadual de Campinas (Unicamp). Campinas, SP, Brasil. 2008

REZENDE, S. O. **Sistemas Inteligentes**: Fundamentos e aplicações. Editora Manole, 2002.

RAMÓN Y CAJAL, S. **Histologie du système nerveux de l'homme & des vertébrés**. Maloine, Paris, França. 1909.

RITTER, H.; MARTINEZ, T.; SCHULTEN, K. **Neural Computation and Self-Organizing Maps**: An Introduction. Reading: Addison-Wesley. 1992.

ROSENBLATT, Frank, **The Perceptron**: A Probabilistic Model for Information Storage and Organization in the Brain, Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, pp. 386–408. 1958.

RYBARSKI,Janusz; HABDANK-WOJEWÓDZKI, Seweryn. **Java Kohonen Neural Network Library**: (JKNNL). Disponível em: <http://jknnl.sourceforge.net/>. Acesso em: 22 out. 2011.

SAITTA, S.; RAPHAEL, B; SMITH, I.F.C., **A Bounded Index for Cluster Validity**, 2007.

SCHATZMANN, Jost. **Using Self-Organizing Maps to Visualize Clusters and Trends in Multidimensional Datasets**. Imperial College, London. 2003.

SILVA, M. P. S. **Mineração de Dados – Conceitos, Aplicações e Experimentos com o Weka**. Livro da Escola Regional de Informática Rio de Janeiro - Espírito Santo Vitória - ES, Rio das Ostras – RJ, 2004.

SHIN, H.W; SOHN, S.Y. **Segmentation of stock trading customers according to potential value**. Samsung Economy Research Institute, Seoul, South Korea. 29 dec. 2003.

ŠILIĆ, Artur et al. Comparing Document Classification Schemes Using K-Means Clustering. Knowledge-Based Intelligent Information and Engineering Systems. Lecture Notes in Computer Science. Springer Berlin. 2008.

TAKATSUKA, M. **An application of the Self-Organizing Map and interactive 3-D visualization to geospatial data**. Proceedings of the 6 th International Conference on GeoComputation. 2001.

TATIBANA, Cassia Yuri; KAETSU, Deisi Yuki. **Uma Introdução às Redes Neurais**. Disponível em <http://www.din.uem.br/~ia/neurais> Acesso em: 5 mai. 2011.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. 1st Edition. Academic Press, USA, 1999.

VENDRAMIN, L., CAMPELLO, R. J. G. B., HRUSCHKA, E. R. **On the Comparison of Relative Clustering Validity Criteria**, In: 2009 SIAM International Conference on Data Mining (SDM 09), Sparks, Nevada, v. 1. p. 733-744, 2009.

WADE, N. **The Science Times Book of The Brain**. The Lyons Press, New York, 1998.

ZUCHINI, M. H. **Aplicações de Mapas Auto-Organizáveis em Mineração de Dados e Recuperação de Informação** – Mestrado em Engenharia da Computação – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2003.