Pós-Graduação em Ciência da Computação

“Otimização de Algoritmo de Agrupamento de Dados para a Classificação Supervisionada de Padrões”

Por

Evandro José da Rocha e Silva

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, FEVEREIRO/2014

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EVANDRO JOSÉ DA ROCHA E SILVA

"Otimização de Algoritmo de Agrupamento de Dados para a Classificação Supervisionada de Padrões"

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR(A): Teresa Bernarda Ludermir.

CO-ORIENTADOR(A): Leandro Maciel Almeida.

RECIFE, FEVEREIRO/2014

Catalogação na fonte
Bibliotecária Joana D'Arc L. Salvador, CRB 4-572

Silva, Evandro José da Rocha e.
Otimização de algoritmo de agrupamento de dados para a classificação supervisionada de padrões / Evandro José da Rocha e Silva. – Recife: O Autor, 2014.
100 p.: fig., tab.

Orientador: Teresa Bernarda Ludermir.
Dissertação (Mestrado) - Universidade Federal de Pernambuco. CIN. Ciência da Computação, 2014.
Inclui referências.

1. Inteligência computacional. 2. Sistemas de reconhecimento de padrões. I.Ludemir, Teresa Bernarda (orientadora). II. Título.

006.3 (22. ed.) MEI 2014-69

Dissertação de Mestrado apresentada por **Evandro José da Rocha e Silva** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Otimização de Algoritmo de Agrupamento de Dados para a Classificação Supervisionada de Padrões**" orientada pela **Profa. Teresa Bernarda Ludermir** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Cleber Zanchettin
Centro de Informática / UFPE

_____

Profa. Aida Araújo Fereira
Depto. de Tecnologia em Análise e Desenvolvimento de Sistemas/IFPE

_____

Prof. Leandro Maciel Almeida
Centro de Informática / UFPE
(Co-orientador)

Visto e permitida a impressão.
Recife, 25 de fevereiro de 2014.

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Aos meus pais, a quem amo muito.

Agradecimentos

Agradeço primeiramente a Deus, pois reconheço que sem Ele nada seria, nada faria. Obrigado Deus, por mais uma conquista. Obrigado, porque Tuas misericórdias se renovam a cada dia e não têm fim. Obrigado por ter me trazido até aqui.

Ao meu pai, minha mãe e irmãos por todo o apoio e ajuda que me proporcionaram durante todo este período em que estive longe deles.

Aos meus irmãos em Cristo, da Ibamsol e da Maná, que nunca cessaram de orar por mim.

Aos meus amigos que dividem ou dividiram o apartamento e também aos que conheci no CIn. Sua amizade foi fundamental.

À Professora Teresa, por sua enorme paciência e por ter me ensinado muita coisa. Seu conhecimento e experiência foram decisivos no decorrer do mestrado. Sou eternamente grato.

Ao Professor Leandro, por todo seu tempo dedicado a me acompanhar e me ensinar. Este trabalho não teria sido completado sem sua ajuda.

Por fim, agradeço a todos os professores que tive durante a graduação no CEUT – em especial ao Professor Harilton – como também aos professores que tive no CIn.

Resumo

O reconhecimento de padrões é uma atividade frequente do ser humano. Entretanto muitas vezes não somos capazes de lidar com o volume de informações disponíveis. Para isso podemos recorrer às técnicas de Aprendizagem de Máquina, cujos algoritmos permitem a um computador aprender e classificar padrões de forma segura e veloz.

Dentre os algoritmos que podem ser utilizados, existem aqueles que fazem parte dos sistemas de múltiplos classificadores. Nesses sistemas, vários classificadores trabalham em conjunto para a classificação dos padrões. O trabalho em conjunto pode ser realizado através da abordagem de seleção de classificadores.

Neste trabalho foi desenvolvida uma metodologia para a construção de sistemas de múltiplos classificadores. Inicialmente o método usa os dados de treinamento para encontrar um mapa do agrupamento dos dados. Com isso, os dados de validação e teste pertencentes a cada grupo são encontrados. Então os classificadores são criados e treinados para cada grupo de dados. Através da abordagem de seleção de classificadores, o melhor classificador para cada agrupamento é encontrado. Os classificadores selecionados são usados para classificar os padrões não vistos que pertencem aos seus respectivos grupos.

Foram implementadas duas versões do método proposto. A primeira, chamada BMGGAVS, conseguiu um bom desempenho, superando, na maioria das vezes, todos os outros métodos utilizados na comparação. A segunda versão do método, chamada BMG2GA, possui uma maior automatização. O BMG2GA não conseguiu resultados tão bons quanto os do BMGGAVS. Entretanto, em algumas situações, o BMG2GA conseguiu resultados próximos ou até melhores que os resultados de alguns dos métodos usados para comparação. Por causa desses últimos resultados, uma série de diretrizes são apresentadas para trabalhos futuros.

Palavras-chave: Reconhecimento de Padrões, Aprendizagem de Máquina, Sistemas de Múltiplos Classificadores (SMC), Comitês de Classificadores (CC), Seleção de Classificadores (SC), *Blockmodeling*, BM-GGA

Abstract

Pattern recognition is a common human activity. However we are not often able to handle with the volume of available information. In this case we can use Machine Learning techniques. Its algorithms allow a computer to learn and classify patterns safely and fast.

Among available algorithms, there are those which are part of multiple classifier systems. In these systems several classifiers work together to classify patterns. This work can be performed through the selection of classifiers approach.

In this work a methodology for the construction of multiple classifier systems was developed. Initially the method uses training data to find a clustering map. Then validation and test data belonging to each cluster are also found. Thereafter classifiers are constructed and trained for each cluster. Through classifier selection approach the best classifier for each cluster is found. Selected classifiers are used to classify unseen patterns that belong to their respective clusters.

Two versions of the proposed method were implemented. The first version, called BMGGAVS, achieved a good performance overcoming in most cases all other methods used for comparison. The second version, called BMG2GA, is more automatic. BMG2GA did not achieve results as good as BMGGAVS's results. However in some occasions BMG2GA achieved close or even better results than some other methods results. Because of BMG2GA results a set of guidelines are presented for future works.

Keywords: Pattern Recognition, Machine Learning, Multiple Classifier Systems (MCS), Classifier Committees (CC), Classifier Selection (SC), Blockmodeling, BM-GGA

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AG Algoritmos Genéticos

AM Aprendizagem de Máquina

BM-GGA *Grouping Genetic Algorithm for Blockmodeling*

BMGGAVS *Grouping Genetic Algorithm for Blockmodeling and Vote System*

BMG2GA *Grouping Genetic Algorithm for Blockmodeling and Ensemble Genetic Algorithm*

CC Comitê de Classificadores

CSEA *Clustering-and-Selection with Evolutionary Algorithms*

CSJADE *Classifier Selection and JADE*

DECORATE *Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples*

ED Evolução Diferencial

ELM *Extreme Learning Machine*

EM *Expectation-Maximization*

FC Fusão de Classificadores

GGA Algoritmo Genético de Agrupamento

HHI *Herfindhl-Hirshman Index*

MLP *Multilayer Perceptron*

MoE *Mixture of Experts*

PCA *Principal Component Analysis*

PSO *Particle Swarm Optimization*

RNA Redes Neurais Artificiais

SC Seleção de Classificadores

SD Seleção Dinâmica

SE Seleção Estática

SFDEGL *Selection and Fusion via Differential Evolution with Global and Local neighborhoods*

SMC Sistemas de Múltiplos Classificadores

SOM *Self-Organizing Maps*

WAVE *Weight-Adjusted Voting for Ensembles of classifiers*

Sumário

1 Introdução

A Aprendizagem de Máquina (AM) é uma área de estudo onde cientistas desenvolvem técnicas que possibilitam às máquinas a capacidade de aprenderem após um processo de observação. Uma das aplicações da aprendizagem é a classificação de instâncias seguindo padrões observados. Esta aplicação é conhecida como reconhecimento de padrões.

Dentre as técnicas desenvolvidas em AM, muitas focam no aprendizado de uma única máquina, ou classificador. Entretanto, um único classificador, em vários casos, não consegue atingir um aprendizado adequado. Uma solução encontrada é a utilização de várias máquinas que, em conjunto, aprendem e classificam as amostras. Entretanto há a necessidade de pesquisas nesta área para que novas técnicas contribuam para a eficiência do aprendizado das máquinas.

Este trabalho foca na construção de um método que melhore o aprendizado de múltiplas máquinas.

1.1 Motivação

O reconhecimento de padrões é bastante abordado em AM. Muitas vezes Redes Neurais Artificiais (RNAs) são utilizadas para tal. Através do reconhecimento de padrões muitos trabalhos podem ser acelerados e/ou até melhorados. O diagnóstico médico é um exemplo: na análise de exames, características de padrões de câncer de mama e nódulos pulmonares

podem ser aprendidas por uma máquina. Devido a sua capacidade de processamento a máquina acelerará a análise de exames feita por um médico especialista. Ao mesmo tempo, a análise pode ser melhorada com a ajuda da máquina, pois esta pode mostrar alguma característica que tenha passado desapercebidamente pelo especialista [LLFM98].

Para alguns casos uma única RNA é suficiente para resolver problemas, contudo nem sempre um classificador é suficiente para obter um bom resultado, ou o resultado desejado [XKS92][HHS94][HS95]. Em contrapartida foi demonstrado formal e empiricamente que um Comitê de Classificadores (CC) pode obter melhores resultados em detrimento de um único classificador [CY05][ML06]. Isto é possível porque os membros do comitê possuem diferentes informações sobre o mesmo problema de modo que contribuem de formas diferentes para a solução do mesmo [Alm11].

Dentre as abordagens para construção de comitês existe a Seleção de Classificadores (SC). A ideia por trás desta abordagem é definir regiões de competência no espaço amostral e então tentar determinar o melhor ou um conjunto dos melhores classificadores para cada região [LSL12b].

A divisão do espaço amostral em várias regiões e, posteriormente, a seleção de classificadores pode ser realizada através de uma técnica chamada de *clustering-and-selection* (agrupamento-e-seleção). Com esta técnica é possível alcançar resultados melhores. Caso contrário é garantido, ao menos, a mesma precisão de treinamento do melhor classificador individual [Kun00].

Além de *clustering-and-selection*, o uso de Algoritmos Genéticos (AG) para a construção de comitês pode também melhorar o desempenho do conjunto de classificadores [ML06]. Logo, é possível que a junção de *clustering-and-selection* com AGs resulte em um comitê de classificadores que retorne uma boa generalização do reconhecimento de padrões.

Portanto, considerando os bons resultados que são alcançados pela metodologia de Seleção de Classificadores e o fato da mesma ser pouco estudada [LSL12a] é importante a realização de novas pesquisas na área. Assim, é interessante verificar se novas maneiras de se agrupar os dados, como também de selecionar os classificadores terão, como consequência, melhores resultados. Desta forma, será possível fornecer mais contribuições a área de reconhecimento de padrões.

1.2 Objetivos

O objetivo deste trabalho é verificar como AGs, uma técnica de agrupamento e um sistema para seleção de classificadores, em conjunto, contribuem para o sucesso de um comitê de classificadores, através da abordagem de SC. Individualmente, as técnicas utilizadas alcançam bons resultados onde são aplicadas. Mas é possível que, em conjunto, possam apresentar um resultado ainda melhor.

A técnica de agrupamento utilizada é o *blockmodeling*. Esta técnica foi usada neste trabalho devido a sua potencialidade para agrupar coerentemente instâncias com características parecidas em um mesmo *cluster*. O *blockmodeling* foi adaptado da área de ciências sociais. Apesar de bem sucedido, é pouco conhecido no mundo de agrupamentos não sociais [DBF04]. Através do agrupamento heurístico de atores sociais, o *blockmodeling* pode descrever a estrutura fundamental de qualquer rede social [BDSS13]. Como resultado, o pesquisador terá em posse a estrutura social em questão dividida em blocos coesos, facilitando o entendimento das relações, de modo que é possível até a visualização de fatos que antes eram de difícil percepção [DBF04].

Outro objetivo no uso do *blockmodeling* é verificar se o mesmo, extraído das ciências sociais, consegue um bom desempenho em sua generalização para dados não sociais. Caso positivo, o escopo do *blockmodeling* pode ser aumentado, recebendo uma importante contribuição e abrindo um novo campo de pesquisa dentro dele. Além disso, o estudo de como formar blocos de dados coesos com ideias semelhantes ou extraídas das ciências sociais pode ter resultados interessantes. Isto é possível pelo fato de que existe um princípio parecido. Ou seja, é esperado que padrões da mesma classe sejam mais tendenciosos a se relacionarem entre si -– dada alguma medida de similaridade. Da mesma forma pessoas de um mesmo grupo social, ou com ideias/personalidades parecidas, têm tendência a se relacionarem.

A divisão dos dados em grupos seguirá o objetivo de facilitar o trabalho dos classificadores. Isto quer dizer que não está no escopo executar uma divisão que mapeie corretamente cada padrão à sua classe. Em outras palavras, seja k o número de classes e n o número de padrões. O objetivo, neste ponto, não é agrupar os dados em k blocos, mas numa quantidade entre 2 e n que possa contribuir para o alcance de melhores resultados.

Quanto à seleção dos classificadores, o objetivo consistirá na simplicidade. Entretanto, se for necessário, o sistema simplificado deverá dar espaço a outro mais complexo, desde que este consiga uma seleção mais concisa, segura e automática da melhor combinação de classificadores únicos.

1.3 Organização da Dissertação

Este trabalho está dividido em seis capítulos. No Capítulo 2 são descritos os métodos de agrupamento e classificação dos dados. Será também apresentado um detalhamento sobre o *blockmodeling* e a ferramenta utilizada para o agrupamento dos dados. Os métodos usados para classificação dos dados também serão explicados.

No Capítulo 3, a abordagem será sobre os comitês de classificadores. Serão abordados alguns métodos clássicos como também outros métodos propostos na literatura. Todos estes métodos foram usados em comparação ao método proposto neste trabalho.

O Capítulo 4 contém a descrição completa do método proposto. O capítulo é dividido em duas seções, de forma que cada versão do método é detalhada em uma seção diferente.

No Capítulo 5, serão apresentados os detalhes dos experimentos realizados. Além dos experimentos, discussões sobre os resultados obtidos são apresentadas.

O Capítulo 6 contém as conclusões do trabalho e propostas para trabalhos futuros.

2

Métodos de agrupamento e classificação de dados

Os métodos de agrupamento e de classificação usados neste trabalho serão descritos neste capítulo.

Inicialmente descreveremos a técnica utilizada para o agrupamento dos padrões. A técnica utilizada é o BM-GGA (*Grouping Genetic Algorithm for Blockmodeling*), proposta por [JBR10]. Além da técnica, será apresentada também a teoria necessária para o melhor entendimento do BM-GGA.

Na Seção 2.2 descreveremos os métodos utilizados neste trabalho para a classificação das instâncias. Os métodos utilizados são:

- **Perceptron**, devido à sua simplicidade e por ser suficiente para *clusters* que possuam instâncias de classes linearmente separáveis;
- **MLP** (*Multilayer Perceptron*), devido à sua capacidade de resolver problemas não linearmente separáveis [BLC00] e, dependendo de sua arquitetura, é possível a aproximação de qualquer função [Cyb88]; e
- **ELM** (*Extreme Learning Machine*), devido à velocidade de seu treinamento aliado, normalmente, a bons resultados, além de ser relativamente recente.

2.1 Métodos de agrupamento

O agrupamento de padrões é uma técnica de aprendizado não supervisionado [CZZ09], ou seja, não utiliza conhecimento prévio sobre o problema como também não possui a ajuda de um especialista. O conjunto de padrões — normalmente vetores num espaço multidimensional -– é dividido para formar grupos (*clusters*) com características similares [CZZ09]. O agrupamento de padrões tem sido utilizado em várias áreas de estudo, tais como aprendizagem de máquina, reconhecimento de padrões, mineração de dados na *web* e segmentação de imagens [ELL01].

De uma maneira mais formal o agrupamento de padrões pode ser definido como segue. Dado um conjunto $X = (x_1, x_2, ..., x_N)$ de instâncias, encontrar $K(2 \leq K < N)$ partições de modo que cada partição possua um subconjunto categoricamente homogêneo [SYK97].

A classificação de algoritmos de agrupamento possui dois grupos: hierárquicos e particionais [CZZ09]. Os algoritmos hierárquicos constroem um dendograma a partir dos dados. Os algoritmos particionais produzem partições nos dados [BM93]. O BM-GGA faz parte do segundo grupo, dado que o mesmo particiona os dados em blocos.

A seguir, apresentaremos o *blockmodeling* e, posteriormente, o *community finding*, ambos necessários para o melhor entendimento do BM-GGA. Logo após, o BM-GGA será detalhado.

2.1.1 *Blockmodeling*

O *blockmodeling* foi introduzido em [WBB76] como um método de análise de dados baseado em permutação de matrizes. No início, o método consistia na manipulação de linhas e colunas com o propósito de encontrar alguma estrutura oculta na matriz [ABL78]. O objetivo do *blockmodeling* pode ser descrito como transformar uma matriz grande e potencialmente incoerente em uma estrutura menor e mais compreensível, de forma que possa ser interpretada mais facilmente [BMFD04] por psicólogos e sociólogos [BDSS13], por exemplo.

A ideia por trás deste método é entender os atores sociais como equivalentes de acordo com a distribuição de suas ligações. Atores sociais similares são agrupados em um mesmo *cluster* [BDSS13] ou bloco.

O *blockmodeling* vem sendo usado como uma ferramenta para estudos sobre redes sociais. Tais redes são formadas por atores sociais, os quais possuem uma ou mais relações com outros atores sociais [DBF04]. Os atores, através de suas mútuas relações, podem ser organizados em grupos sociais. Em redes sociais pequenas e simples, é possível encontrar vários grupos, muitos deles não óbvios. Em maiores escalas, i.e., em redes sociais maiores e mais complexas, é frequentemente impossível distinguir as reais estruturas presentes nos dados [DBF04]. Devido a isso, um dos principais objetivos da análise de redes sociais é identificar, em uma dada rede, *clusters* de atores que compartilham características estruturais em termos de uma ou mais relações [DBF04]. Desta forma, atores que estiverem dentro de um mesmo *cluster* [DBF04] devem ter o mesmo (ou similar) padrão de relacionamento (i.e., ligação com os demais atores).

As relações entre os atores, na literatura, são normalmente representadas através de grafos, e estes, por sua vez, são representados através de matrizes de adjacência [JBR10]. Essas matrizes podem ser manipuladas de forma que os atores pertencentes ao primeiro *cluster* vêm primeiro (nas primeiras linhas e colunas). Logo após vêm os atores do segundo *cluster*, e assim por diante [DBF04]. As matrizes podem, então, ser particionadas em vários blocos.

O *blockmodeling* provê um método e uma análise racional para refinar os *clusters* que compartilham (o máximo possível) uma padronização de ligação dentro dos *clusters* como também entre os mesmos. Em essência [DBF04], o *blockmodeling* monta um conjunto de blocos em um *blockmodel* (ou modelo de blocos). A abordagem do *blockmodeling* seleciona tipos de blocos, agrupa-os em um *blockmodel* e então encaixa o resultado nos dados da rede. As caracterizações dos blocos de um *blockmodel* são construídas das ligações existentes na rede.

De acordo com [DBF04], sejam dois *clusters* de atores $\{a, b, c\}$ e $\{d, e, f, g\}$. Um bloco é definido como a relação entre dois *clusters*. Se forem consideradas apenas as ligações dentro de um *cluster*, a atenção se retém apenas nos atores daquele *cluster*. Isto define um bloco diagonal, exemplificado na Figura 2.1.

	a	b	c
a	0	–	–
b	–	0	–
c	–	–	0

Figura 2.1 Bloco diagonal

As linhas denotam a localização onde a informação empírica sobre a ligação será gravada (no caso binário, ‘1’ significa que um ator se liga ao outro e ‘0’ indica ausência de ligação). Para este exemplo, foi adotado na tabela o valor ‘0’ para a diagonal principal. Entretanto na literatura também é possível encontrar o valor ‘1’, dependendo da relação que é mapeada entre os atores. Caso a atenção esteja retida às ligações entre dois diferentes *clusters*, os blocos passam a ser não-diagonais, como demonstrado na Figura 2.2.

	d	e	f	g
a	–	–	–	–
b	–	–	–	–
c	–	–	–	–

Figura 2.2 Bloco não-diagonal

Na Figura 2.2, cada elemento na matriz é uma localização onde os dados da rede podem ser gravados. Um *blockmodel* é construído ao se usar blocos diagonais e não-diagonais sistematicamente.

Além da classificação apresentada, existe também os tipos de bloco. Neste trabalho serão apresentados dois tipos de bloco: completo e vazio. Os blocos completos são compostos de ‘1’ em todas as ligações, como na Figura 2.3. Na Figura 2.3, a diagonal principal não possui ‘1’ pelo fato de se tratar de um bloco diagonal seguindo a convenção estabelecida na Figura 2.1.

Os blocos vazios, por sua vez, são compostos de ‘0’ em todas as ligações (Figura 2.4).

	a	b	c
a	0	1	1
b	1	0	1
c	1	1	0

Figura 2.3 Bloco completo

	d	e	f	g
a	0	0	0	0
b	0	0	0	0
c	0	0	0	0

Figura 2.4 Bloco vazio

Suponha que haja um conjunto de atores $\{a, b, c, d, e, f, g\}$, cuja matriz de adjacência é a apresentada na Figura 2.5

	a	b	c	d	e	f	g
a	0	1	1	0	0	0	0
b	1	0	1	0	0	0	0
c	1	1	0	0	0	0	0
d	1	1	1	0	0	0	0
e	1	1	1	0	0	0	0
f	1	1	1	0	0	0	0
g	1	1	1	0	0	0	0

Figura 2.5 Matriz de Adjacência

Ao observar as ligações, é possível perceber que: (1) $\{a, b, c\}$ têm ligações uns com os outros; (2) estes mesmos atores não possuem ligações com o *cluster* $\{d, e, f, g\}$; (3) no segundo *cluster* os atores não têm ligações entre si; e (4) $\{d, e, f, g\}$ possuem ligações com $\{a, b, c\}$. A partir dessas observações é possível particionar a matriz em quatro

blocos (Figura 2.6).

	a	b	c	d	e	f	g
a	0	1	1	0	0	0	0
b	1	0	1	0	0	0	0
c	1	1	0	0	0	0	0
d	1	1	1	0	0	0	0
e	1	1	1	0	0	0	0
f	1	1	1	0	0	0	0
g	1	1	1	0	0	0	0

Figura 2.6 Matriz de Adjacência particionada em blocos

O particionamento deixa claro que os quatro blocos obtidos podem ser descritos pelos seus tipos de bloco (Figura 2.7). Caso se substitua o bloco completo por '1' e o vazio por '0', é possível compactar ainda mais a informação obtida da rede (Figura 2.8).

completo	vazio
completo	vazio

Figura 2.7 Matriz de Adjacência descrita através dos tipos de bloco

1	0
1	0

Figura 2.8 Matriz de Adjacência descrita através dos tipos de bloco, simplificadamente

Esta forma compacta é vista como uma imagem da rede social original [DBF04]. Para esta representação da rede, é usado o termo *model* (ou modelo) e é dito que a representação é um *blockmodel*. A imagem pode ser usada como uma descrição da estrutura essencial da rede e, a partir dela, podem ser realizadas interpretações com mais facilidade.

Mais informações sobre *blockmodeling* podem ser encontradas em [DBF04].

2.1.2 *Community Finding*

O *community finding* é um método que pode ser visto como um caso particular do *blockmodeling* [CLH11]. Sua metodologia procura particionar uma dada rede social em estrututas (*community structures*), as quais são grupos cuja característica é a alta densidade de ligação entre seus membros, enquanto que as ligações entre membros de grupos diferentes são esparsas ou inexistentes [NG04].

	a	*b*	*c*	*d*	*e*	*f*	*g*
a	1	1	1	0	0	0	0
b	1	1	1	0	0	0	0
c	1	1	1	0	0	0	0
d	0	0	0	1	1	0	0
e	0	0	0	1	1	0	0
f	0	0	0	0	0	1	1
g	0	0	0	0	0	1	1

Figura 2.9 *Community Finding*

A motivação do *Community Finding* é a mesma do *blockmodeling*, i.e., encontrar e analisar grupos em uma dada rede social de forma que seja possível entender e visualizar sua estrutura [NG04]. A busca dos grupos é feita a partir de algum algoritmo de agrupamento, o qual é escolhido pelo pesquisador para a divisão dos dados. Em [NG04] alguns desses algoritmos são descritos.

Utilizando a terminologia apresentada na seção anterior (Seção 2.1.1), é possível descrever o *community finding* como uma ferramenta que produz um único tipo de *blockmodel*. Este modelo de blocos consiste em blocos diagonais completos e blocos não-diagonais vazios [CLH11], como exemplificado na Figura 2.9[1]. Através dos algoritmos de agrupamento, procura-se encaixar nesse modelo as estruturas encontradas.

[1]Note que desta vez o primeiro bloco possui a diagonal principal com o valor ‘1’, em vez de ‘0’ como anteriormente.

2.1.3 BM-GGA

O BM-GGA utiliza um Algoritmo Genético de Agrupamento (GGA) [Fal92] para formar blocos diagonais completos e blocos não-diagonais vazios. O GGA é uma modificação do Algoritmo Genético, especificamente para problemas de agrupamento. Esta modificação foi motivada pelo fato de o AG apresentar as seguintes deficiências no trabalho com problemas de agrupamento de dados [Fal92] : (1) o esquema de codificação padrão é altamente redundante; isto faz com que o espaço de busca seja maior que o necessário; (2) a aplicação do operador de cruzamento frequentemente resulta em uma prole com poucas, ou nenhuma característica em comum com seus pais, e (3) a mutação padrão pode ser muito destrutiva. Para contornar tais problemas, o GGA foi proposto, e com ele, operadores e cromossomo especializados.

O novo cromossomo possui duas partes, onde a primeira mapeia cada gene em um grupo, ou *cluster*, e a segunda parte lista os grupos. Abaixo uma ilustração utilizada em [JBR10].

$n = 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 2\ \ 3$
$g = 1\ 2\ 3$

Figura 2.10 Exemplo do novo cromossomo

Em n cada instância é mapeada para algum grupo. Os três primeiros padrões fazem parte do grupo 1, os quatro padrões seguintes fazem parte do grupo 2 e o último padrão faz parte do último grupo. Finalmente g lista os três grupos existentes.

O operador de cruzamento age diretamente na segunda parte do cromossomo. A sua atuação na primeira parte é indireta, pois é uma consequência da sua atuação na segunda parte [SLA13]. Uma vez que os pais são selecionados (no BM-GGA a seleção é feita através do método da roleta), dois pontos de corte são escolhidos na lista de grupos do primeiro pai. Todos os *clusters* que estiverem entre os pontos de corte serão os grupos contribuintes para a prole. Estes *clusters* são então inseridos na parte de grupos do segundo pai, e a sua primeira parte será então modificada para refletir a alteração feita, produzindo, então, a prole. O processo de cruzamento é mostrado na Figura 2.11.

De acordo com a Figura 2.11 é possível perceber que (1) apenas uma prole foi produzida e (2) foi feita uma alteração na prole. Originalmente, o cruzamento do GGA é

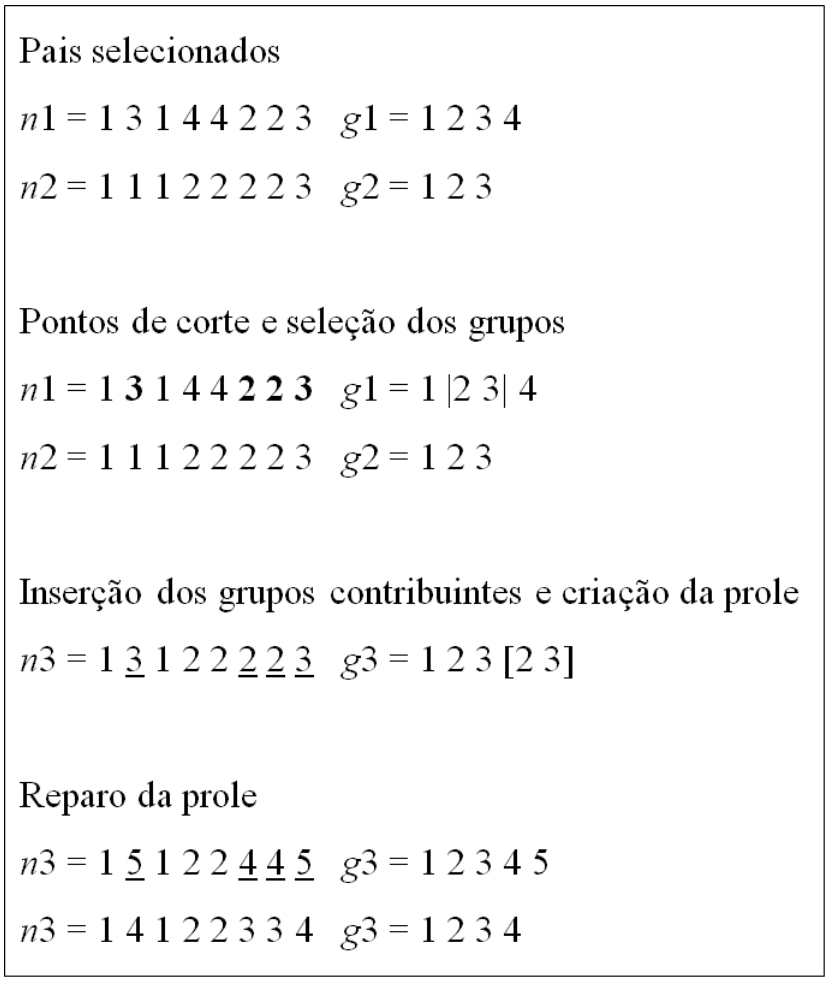

Figura 2.11 Exemplo de cruzamento

semelhante ao do AG simples. Isto quer dizer que dois pais produzem duas proles. Além disso, o GGA não prevê a alteração que foi feita na prole. Logo, a alteração da prole é uma das particularidades do BM-GGA.

Duas outras particularidades do BM-GGA são a exclusão do operador de mutação e a inserção do operador de reparo. O operador de reparo trata de padrões inviáveis, i.e., padrões que pertencem a grupos onde existe ao menos um membro ao qual o mesmo não se liga; em outras palavras, há o mapeamento de ligações inexistentes. Além de padrões inviáveis, o operador de reparo procura evitar grupos *singleton* -– i.e., grupos que tenham apenas um membro — alocando-os em outros grupos. Na Figura 2.12, o pseudocódigo do BM-GGA é apresentado.

A ordenação, no passo 2(a), é feita de acordo com um índice chamado HHI (*Herfindahl-*

Passo 1: Inicialização

1(a) Gere aleatoriamente uma população de *n* soluções

Passo 2: Seleção Parte 1

2(a) Ordene os cromossomos

2(b) Calcule a aptidão de cada cromossomo

2(c) Salve a melhor solução dentre todas (em todas as gerações)

Passo 3: Repita até que *n* filhos sejam criados

3(a) Seleção Parte 2

- Selecione com reposição o Pai 1
- Selecione com reposição o Pai 2

3(b) Execute o cruzamento

- Selecione dois pontos de corte
- Insira a porção do Pai 1 designada pelos pontos de corte no Pai 2
- Ajuste a prole para refletir os grupos inseridos pelo Pai 1

3(c) Remova grupos vazios

3(d) Renumere os grupos

3(e) Repare a prole

- Remova padrões inviáveis dos grupos
- Identifique *singletons* para realoca-los
- Realoque padrões inviáveis e *singletons*

Passo 4: Vá para o Passo 2 até que se alcance o número máximo de gerações

Figura 2.12 Pseudocódigo do BM-GGA

Hirshman Index) [Her50][Hir64]. O HHI foi desenvolvido na área de economia, onde é aplicado na análise do grau de concentração industrial [JBR10], mais especificamente para a análise de concorrência. Se o valor do índice é alto, significa que há poucas empresas em concorrência. Do contrário, se o valor é baixo, há muitas empresas em concorrência.

Maximizar o HHI, no contexto do BM-GGA, significa a busca por um pequeno número de grandes blocos, ou seja, haverá poucos blocos com o máximo possível de

membros em cada um. Formalmente a função objetivo do BM-GGA é:

$$\Sigma_{k=1}^{b}(\Sigma_{\lambda_{ik}}^{n})^2 \tag{2.1}$$

Na equação (2.1) , b é o número de blocos, n é o número de padrões e $\lambda_{ik} = 1$ se o padrão i pertencer ao bloco k, senão $\lambda_{ik} = 0$. Por definição [JBR10], um bloco permite a um padrão pertencer a somente um grupo.

Para exemplificar o HHI, será usado o cromossomo 1, da Figura 2.11, onde $n1 =$ 13144223. O HHI deste cromossomo seria calculado através do seguinte somatório:

$$2^2 + 2^2 + 2^2 + 2^2 = 16$$

No exemplo acima, cada bloco possui dois membros. Entretanto, se houvesse menos blocos, e, consequentemente, mais padrões em cada bloco, o valor do HHI seria maior.

A aptidão de cada cromossomo, no passo 2(b), leva em conta a ordenação dos mesmos.

$$f = \frac{2r}{N(N+1)} \tag{2.2}$$

Na equação (2.2) f é a aptidão do cromossomo, r é a posição do cromossomo na ordenação e N é o número de cromossomos ordenados, i.e., o tamanho da população de soluções.

Continuando o exemplo do cromossomo 1. Suponha que a população seja de 10 cromossomos e após a ordenação sua posição seja 10. Neste caso, quanto maior o HHI, maior o valor da posição[2], logo o cromossomo 1 tem o maior valor do HHI. Portanto, o valor de sua aptidão será:

$$f = \frac{2 \times 10}{10 \times 11} \quad \therefore \quad f = 0,1818 \tag{2.3}$$

[2]Essa abordagem leva em conta o operador de seleção, no caso, a roleta; portanto, as melhores soluções devem ter os maiores valores de aptidão e consequentemente terão mais chances de serem selecionadas.

Por fim, a melhor solução é encontrada e guardada de acordo com o passo 2(c), mostrado na Figura 2.12. A seleção e cruzamento dos pais foram descritos na Figura 2.11. Detalhes sobre o reparo da prole podem ser encontrados em [JBR10].

2.2 Métodos de classificação

Os métodos usados para classificação dos dados neste trabalho fazem parte do paradigma de aprendizado supervisionado. No aprendizado supervisionado [BLC00] as saídas desejadas da rede são conhecidas. Com esse conhecimento é possível ajustar os parâmetros necessários da RNA para que a mesma consiga retornar melhores resultados (i.e., diminuir seu erro) a cada iteração do treinamento. Neste trabalho, como dito anteriormente, três tipos de classificadores foram usados: Perceptron [Ros58], MLP [SSF10][BLC00] e ELM [HZS06].

2.2.1 Perceptron

O Perceptron foi introduzido por [Ros58], e consiste de uma estrutura simples com uma regra de aprendizado.

A topologia original do Perceptron [BLC00] era composta por unidades de entrada, por um nível intermediário – o qual é formado pelas unidades de associação – e por um nível de saída – que é formado pelas unidades de resposta. Ainda que tenha três níveis, essa topologia é conhecida como de uma só camada. Isto se deve ao fato de somente o nível de saída possuir propriedades adaptativas, necessárias para o aprendizado.

Dado que o Perceptron é um classificador linear, ele é incapaz de classificar corretamente instâncias de classes não linearmente separáveis. Devido a essa limitação este classificador recebeu duras críticas vindas de Minsky e Papert [MP69]. Como consequência houve um impacto significativo sobre as pesquisas em RNAs, resultando num grande desinteresse pela área durante os anos 70 e início dos anos 80 [BLC00].

Entretanto, devido à sua simplicidade e ao fato do mesmo sempre convergir caso o problema em questão seja linearmente separável [Ros62], o seu uso ocorreu devido a

possibilidade de, após o agrupamento, haver grupos com classes linearmente separáveis. Nestes casos, um Perceptron é completamente suficiente.

2.2.2 MLP

Uma rede MLP, ou Perceptron Multicamadas [SSF10] é uma rede *feedforward* – i.e., com fluxo de dados linear, no sentido da camada de entrada para a camada de saída –, que apresenta uma camada de entrada, uma ou mais camadas intermediárias (escondidas) e uma camada de saída. Os nodos de duas camadas adjacentes são completamente conectados e o valor da saída é obtido através de uma sequência de ativação de funções definidas em cada camada [SSF10].

Diferentemente do Perceptron, a MLP é capaz de tratar problemas não linearmente separáveis [BLC00]. Além disso, uma MLP com uma única camada escondida pode implementar qualquer função contínua [Cyb89] e uma MLP com duas camadas intermediárias permite a aproximação de qualquer função [Cyb88].

Da mesma forma que o Perceptron, as MLPs possuem regras de aprendizado; a diferença é que são capazes de ajustar os pesos de todas as camadas. O *Backpropagation* é o algoritmo de aprendizado mais utilizado e um dos responsáveis por renovar o interesse sobre RNAs na década de 80 [BLC00]. Neste trabalho, entretanto, foi utilizado em seu lugar o algoritmo de aprendizado *Levenberg-Marquardt* [Lev44][Mar63], devido ao seu bom desempenho ao mesmo tempo em que possui uma rápida convergência [Alm07].

Um dos problemas das redes MLP se refere à definição do número de nodos na(s) camada(s) intermediária(s). Este número influencia no desempenho final da rede. Entretanto não existe na literatura uma regra que defina o número de nodos. Devido a isso, neste trabalho foram utilizadas algumas configurações, cujas diferenças residem apenas no número de nodos da camada escondida.

2.2.3 ELM

Extreme Learning Machine (ELM) foi introduzido em [HZS06]. ELM é um algoritmo de aprendizado para redes *feedforward* com uma única camada escondida. Como motivação

para a criação do ELM, os autores apontam que o uso comum de algoritmos de aprendizado baseados em gradiente deixa a velocidade do aprendizado mais lenta que o exigido, além de os mesmos ajustarem todos os parâmetros da rede.

No ELM, os autores buscam a otimização da velocidade do aprendizado, ao mesmo tempo em que determinam analiticamente os pesos entre a camada escondida e a camada de saída. Os pesos entre as entradas e os neurônios da camada intermediária são escolhidos aleatoriamente.

As vantagens do ELM são [HZS06]: (1) aprendizado extremamente rápido, conseguindo muitas vezes finalizar sua execução em segundos ou mesmo em menos de um segundo; (2) uma melhor generalização que o *Backpropagation* em vários casos; (3) não sofre com mínimo local, taxa de aprendizado imprópria ou *overfitting* e (4) pode ser usado para treinar redes com várias funções de ativação não diferenciáveis.

2.3 Considerações finais

Neste capítulo foram abordados o método de agrupamento e os métodos de classificação utilizados.

O método de agrupamento é uma solução, *a priori*, para os estudos sociais. Entretanto, dada sua capacidade de produzir um bom agrupamento de acordo com o relacionamento entre os padrões, foi decidido verificar o quão bem tal método se aplicaria no reconhecimento de padrões. O BM-GGA, como mostrado, é apenas uma técnica de um caso simples e específico do *blockmodeling*.

A seção sobre os métodos de classificação apresentou a descrição dos três tipos de classificadores (Perceptron, MLP e ELM) utilizados no método proposto neste trabalho.

3

Comitês de Classificadores

Comitês de Classificadores (CC) – também conhecidos como Sistemas de Múltiplos Classificadores (SMCs) [Pon11] -– consistem na combinação de dois ou mais classificadores a fim de que se obtenham melhores resultados na classificação de padrões [KHDM98].

A motivação principal para o uso de CCs vem da ideia que um conjunto de diferentes classificadores pode oferecer informações complementares sobre os padrões a serem classificados. Isto, portanto, aumenta a efetividade geral do processo de reconhecimento [LSL12b][WBB76]. Outra motivação é o fato de que a construção de um único classificador *forte* -– i.e., que tenha um alto desempenho -– pode ser extremamente difícil [HS95]. A construção de vários classificadores *fracos*, além de ser mais fácil, pode retornar o desempenho desejado.

Teoricamente, um comitê de classificadores *fracos* pode vir a constituir um classificador *forte* [Rok10]. Entretanto, para que isso ocorra, o comitê precisa ser composto de classificadores independentes que classifiquem corretamente um padrão com uma probabilidade maior que 50% [Rok10]. Idealmente um comitê de classificadores deve se apropriar das vantagens de cada classificador individual. Ao mesmo tempo precisa evitar todas as suas fraquezas, acrescendo, assim, a precisão de sua classificação [HHS94].

Um comitê de classificadores típico contém normalmente três componentes [Rok10]:

1. Indutor base: ou classificador base, é um algoritmo de indução que para um dado conjunto de treinamento forma um classificador.

2. Gerador de diversidade: o qual é responsável por gerar classificadores diversificados.

3. Combinador: é responsável pela combinação dos classificadores.

Como exemplo de indutores, temos o Perceptron, o MLP e o ELM, que foram usados neste trabalho. Os mesmos são usados como base para a construção de vários classificadores.

Quanto à diversidade, esta é uma condição crucial para a obtenção de comitês precisos em sua classificação [Kun04][KW03]. Isto acontece porque classificadores diversificados levam a classificações não correlacionadas, as quais, por sua vez, aumentam a precisão da classificação [Hu01]. Para se alcançar uma maior diversidade, o pesquisador pode escolher dentre várias abordagens [Rok10]:

1. Manipular o conjunto de treinamento: a entrada usada pelo indutor é variada. Cada membro do comitê é treinado em um conjunto diferente de amostras.

2. Manipular o indutor: cada membro do comitê é treinado com um indutor que é diferentemente manipulado.

3. Modificando a representação do atributo desejado: cada classificador lida com um conceito diferente do resultado desejado.

4. Particionando o espaço amostral: cada membro do comitê é treinado em um subespaço diferente.

5. Hibridização: a diversidade é obtida ao se usar vários indutores base ou estratégias de construção de comitês.

Por fim, a combinação dos classificadores é feita normalmente através de duas abordagens [Kun00]: Fusão de Classificadores (FC) e Seleção de Classificadores (SC).

3.1 Fusão de Classificadores

A Fusão de Classificadores é uma mímica ao processo humano de buscar várias opiniões de especialistas antes de tomar uma decisão importante [Kun93]. Neste processo a opinião de vários especialistas sobre determinado assunto é observada em conjunto. Através de alguma função (e.g., voto majoritário ou alguma regra aritmética) sobre tais opiniões, uma decisão final é tomada.

As funções tentam combinar os resultados da classificação obtidos por vários classificadores individuais. O objetivo é atingir o consenso do grupo sobre a classe de uma determinada instância [LSL12b].

Seja $D = \{D_1, D_2, ..., D_L\}$ um conjunto de L classificadores e $\Omega = \{\omega_1, \omega_2, ..., \omega_C\}$ um conjunto de C classes. A matriz D_{ij}, onde $i = 1, ..., L$, e $j = 1, ..., C$, é uma representação do resultado da classificação e mostra os graus de suporte do classificador i para a classe j. A Figura 3.1 contém um exemplo simples que possui três classificadores e quatro classes.

	ω_1	ω_2	ω_3	ω_4
D_1	0,1	0,5	0,4	0,0
D_2	0,3	0,4	0,2	0,1
D_3	0,2	0,0	0,8	0,0

Figura 3.1 Exemplo de matriz D_{ij}

A instância em questão, por exemplo, recebe do classificador D_1 o rótulo da segunda classe. Do segundo classificador também o rótulo da segunda classe, enquanto que o terceiro classificador a rotula como pertencente à terceira classe.

As funções podem ser aplicadas sobre a matriz D_{ij} para obter uma fusão das decisões dos classificadores. Algumas das funções comumente usadas são: Voto com peso, voto majoritário, regra do mínimo, regra do máximo, regra do produto, regra da média e regra da mediana [KHDM98].

Votos ponderados e voto majoritário são similares. Em suma, a classe que tiver

o maior *apoio* dos classificadores será o rótulo escolhido para o padrão em questão [KHDM98]. A diferença nos votos ponderados é que a escolha de alguns classificadores terá uma influência maior que a de outros. Levando em consideração o exemplo da Figura 3.1, o voto majoritário escolheria a classe ω_2, dado que a mesma recebe dois dos três votos.

A regra do mínimo [KHDM98] seleciona o menor valor de suporte dado a cada classe. Destes valores mínimos, o maior será o escolhido como rótulo da instância em questão. Da Figura 3.1, os valores mínimos correspondem respectivamente a 0,1, 0, 0,2 e 0. Logo, a classe ω_3 seria a escolhida.

A regra do máximo [KHDM98] consiste em selecionar o maior valor de suporte dado a cada classe, e desses valores seleciona-se o maior. No exemplo da Figura 3.1, os valores selecionados seriam 0,3, 0,5, 0,8 e 0,1, respectivamente. Desta forma, a classe ω_3 seria a escolhida.

A regra do produto [KHDM98] realiza o produto dos suportes de cada classe, e então o maior valor é selecionado. Os produtos dos suportes de cada classe, da Figura 3.1, são, respectivamente, 0,006, 0, 0,064 e 0. Novamente a classe ω_3 é a escolhida.

A regra da média [KHDM98] aplica a média aritmética sobre os valores de suporte das classes. A classe que obtiver a maior média é escolhida. Na Figura 3.1, as médias são 0,2, 0,3, 0,47 e 0,03 respectivamente. Mais uma vez a classe ω_3 seria a escolhida.

Por fim, a regra da mediana [KHDM98] aplica a mediana sobre os valores de suporte das classes. A classe que obtiver o maior valor é selecionada como o rótulo da instância a ser classificada. As medianas das classes da Figura 3.1 são 0,2, 0,4, 0,4 e 0, respectivamente. Neste caso as classes ω_2 e ω_3 empataram.

Os métodos clássicos [Pol06], por exemplo, *bagging*, *boosting* e suas muitas variações utilizam a fusão de seus classificadores para uma decisão final. Boa parte dos demais métodos [LSL12a] também utilizam a fusão dos classificadores. As funções mais comumente usadas para fusão são a do voto majoritário, regra da média e regra do produto [Pon11].

3.2 Seleção de Classificadores

Na seleção, um ou mais classificadores [Kun00] são especialistas em alguma região do espaço amostral. Eles serão escolhidos para classificar um padrão desconhecido, caso o padrão pertença à sua região [LY01]. Referindo a um processo humano, a Seleção de Classificadores é semelhante a um grande projeto, que pode ser subdividido. Os especialistas não têm competência para inferir sobre todo o projeto, mas cada um pode dar uma valiosa contribuição dentro de sua área. Quanto melhores forem os especialistas em cada subdivisão do projeto, melhor será o projeto como um todo.

As técnicas de seleção de classificadores podem ser separadas em duas metodologias [RG05]. De acordo com o primeiro tipo, chamado de seleção estática (SE), a solução ótima encontrada é fixada e usada para a classificação de padrões não vistos. De acordo com o segundo tipo, chamado de seleção dinâmica (SD), a seleção é feita durante a classificação.

Uma técnica, chamada *clustering-and-selection* (agrupamento-e-seleção, a qual é usada neste trabalho) possui características tanto da SE quanto da SD. Nesta técnica o espaço amostral é incialmente particionado em $K > 1$ regiões disjuntas por algum algoritmo de agrupamento. O melhor classificador para cada grupo é identificado e selecionado para classificar todo novo padrão desconhecido que pertença àquela região [Kun00]. As regiões definidas nesta técnica são estáticas durante todo o processo de treinamento enquanto que os classificadores são selecionados dinamicamente [RG05].

3.3 Modelos Clássicos

Neste trabalho, três dos métodos clássicos para comitês são usados a fim de comparações, e os mesmos serão descritos adiante.

3.3.1 *Bagging*

Num comitê criado por *Bagging*, cada classificador é treinado em um conjunto de m exemplos selecionados com reposição do conjunto de treinamento original de tamanho m [MM04]. Aquele conjunto é chamado de *réplica de inicialização* (*bootstrap replicate*) do conjunto de treinamento original. Cada réplica contém, em média, 63,2% do conjunto de treinamento original, com muitas instâncias aparecendo múltiplas vezes.

O *Bagging* [Rok10] é normalmente aplicado a algoritmos de aprendizado que são instáveis, i.e., uma pequena mudança no conjunto de treinamento leva a mudanças significativas no classificador construído.

A combinação do comitê é via fusão de seus classificadores, e a predição de novos exemplos é feita através do voto majoritário [MM04][Rok10].

3.3.2 AdaBoost (*Adaptive Boosting*)

Boosting é um método geral para melhorar o desempenho de um classificador fraco. O método trabalha ao treinar repetidamente um classificador fraco em várias distribuições dos dados de treinamento [Rok10]. Os classificadores são então combinados em um único classificador composto e forte.

Há diversas variações de Boosting na literatura [MM04]. Entretanto, quando se cita *Boosting*, ou AdaBoost, normalmente a referência é ao algoritmo AdaBoost.M1, descrito em [FS96]. A ideia principal deste método é dar um maior foco a padrões que são mais difíceis de classificar [Rok10].

O AdaBoost [MM04] mantém um conjunto de pesos sobre os exemplos de treinamento. Em cada iteração i, o classificador C_i é treinado para minimizar o erro ponderado sobre o conjunto de treinamento. O erro ponderado de C_i é computado e usado para atualizar a distribuição de erros nos exemplos de treinamento. Os pesos dos exemplos que foram erroneamente classificados são aumentados, enquanto que os pesos dos exemplos que foram corretamente classificados são diminuídos. O próximo classificador é treinado com os pesos já atualizados e o processo se repete.

Após o treinamento, a classificação de instâncias desconhecidas é feita através do voto ponderado de cada classificador. O peso de cada classificador é calculado de acordo com sua precisão no conjunto em que foi treinado [MM04]. Porém, caso haja poucas instâncias ou bastante ruído, o método pode retornar um fraco desempenho [MM04].

Além do AdaBoost.M1 — que lida apenas com classificação binária -– existe também o AdaBoost.M2, que consiste em uma alternativa para problemas com múltiplas classes [Rok10].

3.3.3 *Mixture of Experts*

Mixture of Experts (MoE) foi introduzido por [JJNH91] e pode ser visto como uma arquitetura estruturada em árvore, baseada no princípio *dividir para conquistar*. A arquitetura original tem três principais componentes: (1) especialistas, que são funções de regressão ou classificadores; (2) o portão (*gate*) que produz partições suaves — i.e., que não são disjuntas -– no espaço amostral e assinala diferentes classificadores para cada partição e (3) um modelo probabilístico para combinar os especialistas e o portão.

No MoE [YWG12] os especialistas e o portão cooperam para a solução de um problema de aprendizado supervisionado não-linear. Para isso o espaço amostral é dividido em um conjunto aninhado de regiões. O portão faz uma divisão suave de todo o espaço amostral, e os especialistas treinam nessas regiões. O portão é treinado simultaneamente a fim de selecionar estocasticamente o especialista com o melhor desempenho na resolução do problema [CXC99].

Durante o treinamento o portão e os especialistas são dissociados, de forma que a arquitetura do MoE passa a ser uma estrutura modular [YWG12]. O algoritmo *Expectation-Maximization* (EM) foi então introduzido [JJ94]; dessa forma o processo de aprendizado foi também dissociado de maneira que encaixa com a estrutura modular do MoE [CXC99].

O *Mixture of Experts*, apesar de ainda ser bastante utilizado em sua forma orginal, possui muitas variações e mais detalhes do que quando foi introduzido há mais de 20 anos [YWG12]. Por exemplo, os especialistas e o portão do modelo original foram modificados para funções de classificação ou regressão mais complexas, o algoritmo de

aprendizado foi trocado, e o modelo de mistura dos classificadores foi modificado para estimação de densidade e para a representação de dados de séries temporais [YWG12].

3.4 Outros Modelos

Os demais modelos descritos nesta seção serão utilizados na análise comparativa deste trabalho. Os modelos foram escolhidos pelo fato de construírem comitês de classificadores para a resolução de problemas e por alcançarem bons resultados.

Nas próximas subseções, os seis métodos escolhidos serão descritos. Os três primeiros possuem uma maior similaridade ao método proposto.

3.4.1 CSEA (*Clustering-and-Selection with Evolutionary Algorithms*)

CSEA [Alm11] é um método de duas fases que constrói automaticamente um SMC usando SC. Inicialmente, o CSEA encontra de forma otimizada o agrupamento para o conjunto de dados de treinamento. Em uma segunda etapa o método constrói uma Rede Neural Artificial otimizada para cada grupo encontrado na etapa anterior. O funcionamento do classificador construído com o CSEA inicia com a apresentação de um conjunto de dados ao agrupamento e identificação do grupo mais próximo. Essa proximidade é baseada na distância Euclidiana dos dados em relação ao agrupamento de dados. Com a identificação do grupo mais apropriado a representar os dados de entrada, acontece então a apresentação dos dados ao classificador associado ao grupo. Com a apresenteção dos dados ao classificador obtem-se o rótulo classificador para o sinal de entrada, ou seja, a resposta do classificador como um todo.

Na primeira fase, de agrupamento, os experimentos foram feitos com *k-means* e a rede SOM (*Self-Organizing Maps*). Na fase de classificação, foram utilizados Perceptrons Lineares e Multicamadas. Para a otimização dos parâmetros e desempenho das diferentes técnicas usadas em ambas as fases, foram usados PSO (*Particle Swarm Optimization*), com ajuste dinâmico de parâmetros, e Evolução Diferencial (ED), integrada a um AG.

Para os testes foram usadas dezoito bases de dados do repositório UCI [AN07]:

Cancer, *Card*, *Pima Indinas Diabetes*, *Gene*, *Glass*, *Heart*, *Heartc*, *Horse*, *Mushroom*, *Page-blocks*, *Pendigits*, *Segment*, *Satimage*, *Soybean*, *Thyroid*, *Vowel*, *Waveform* e *Yeast*. Na maioria das bases o CSEA superou outros métodos que constroem SMCs, entre clássicos e outros presentes na literatura. Entretanto, este método acaba sendo bastante complexo, devido ao número de diferentes técnicas que o compõe. Como consequência o CSEA exige um alto custo computacional e de tempo.

3.4.2 CSJADE (*Classifier Selection and JADE*)

CSJADE [LSL12a], similarmente ao CSEA, é um método que em duas fases constrói automaticamente um SMC usando SC. Em uma das fases, classificadores individuais são criados. Tais classificadores consistem em RNAs, e seus parâmetros são determinados através de uma busca realizada com Evolução Diferencial Adaptativa.

Na outra fase um agrupadamento é executado. A seleção dos classificadores para cada *cluster* também é executada. Tanto o agrupamento quanto a seleção ocorrem ao mesmo tempo através de uma busca via Evolução Diferencial Adaptativa.

Nos testes foram usadas sete bases de dados do repositório UCI [AN07]: *Cancer*, *Card*, *Diabetes*, *Glass*, *Heart*, *Heartc* e *Horse*. Em seis das sete bases o método conseguiu um melhor desempenho que os demais usados para comparação. O principal problema é o consumo de tempo.

3.4.3 SFDEGL (*Selection and Fusion via Differential Evolution with Global and Local neighborhoods*)

Este método [LSL12b] é a evolução do CSJADE, descrito anteriormente. Em vez de utilizar somente seleção de classificadores, os autores optaram por combinar seleção e fusão de classificadores. Outra modificação foi acrescentar na fase de agrupamento redes SOM (*Self-Organizing Maps*). Por fim veio o uso de Evolução Diferencial com vizinhança local e global.

Os testes foram feitos com quatro bases de dados do repositorio UCI [AN07]: *Cancer*,

Diabetes, *Heart* e *Heartc*.Houve melhoras em relação ao CSJADE, mas o consumo de tempo continua sendo um problema.

3.4.4 DECORATE (*Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples*)

DECORATE [MM04] foca na diversidade do SMC, dado esta propriedade ser importante para bons resultados. Este método constrói diretamente diversas hipóteses usando exemplos de treinamento adicionais, artificialmente construídos. É uma técnica de meta-aprendizado geral que usa algum classificador *forte* para construir um comitê diverso e efetivo de maneira simples e direta.

A cada iteração um novo classificador é criado e treinado com dados do conjunto de treinamento original e com dados artificialmente criados – estes são novamente criados a cada iteração, garantindo a diversidade. O novo classificador só não será inserido no comitê caso diminua o desempenho do mesmo. Novos classificadores serão iterativamente treinados e adicionados no comitê até que este alcance o tamanho desejado ou o número de iterações alcance um limite. A classificação de uma instância desconhecida x é feita através da seguinte equação:

$$\hat{P}_y(x) = \frac{\sum_{C_i \in C^*} \hat{P}_{C_{i,y}}(x)}{\mid C^* \mid} \tag{3.1}$$

Na equação (3.1), C_i é um classificador base no comitê C^*, $\hat{P}_{C_{i,y}}(x)$ é a probabilidade estimada da instância x pertencer à classe y de acordo com o classificador C_i, e $\hat{P}_y(x)$ é a probabilidade de x pertencer a y. A classe mais provável é então selecionada.

Os experimentos realizados usam comitês de árvore-de-decisão, entretanto é possível que outro classificador base seja utilizado também. Os resultados mostram que o DECORATE alcançou uma maior precisão que seu classificador base e demais outros utilizados em comparação, e em um caso melhorando seu desempenho em bases pequenas. As bases utilizadas foram retiradas do UCI [AN07]: *Anneal*, *Audio*, *Autos*, *Breast-w*, *Credit-a*, *Glass*, *Heart-c*, *Hepatitis*, *Colic*, *Iris*, *Labor*, *Lymph*, *Segment*, *Soybean* e *Splice*.

3.4.5 MultiBoosting

MultiBoosting [Web00] é uma técnica que combina AdaBoost com *Wagging* (variante do *Bagging*).

Através de observações pode-se concluir que [Web00] é possível obter benefícios da junção de AdaBoost com *Bagging*, mesmo com um comitê de poucos classificadores. Essa junção pode manter a redução de viés do AdaBoost. Ao mesmo tempo, a redução de variância também alcançada pelo AdaBoost é reforçada pelo *Bagging*.

Uma maneira de unir os métodos é criando dois subcomitês, de forma que um conterá membros formados pelo AdaBoost e outro terá membros formados pelo *Bagging*. Neste ponto surge o desafio de como combinar os votos dos dois subcomitês. O AdaBoost dá pesos aos votos de seu SMC enquanto que o *Bagging* não dá pesos, fazendo dos votos dos membros de cada SMC incomensuráveis. Em vez de continuar nesta abordagem e procurar uma solução para esta questão, o MultiBoosting explora o encapsulamento de um conjunto de subcomitês, formados, cada um, pelo AdaBoost. E em vez de usar *Bagging* – que reduziria o número de exemplos de treinamento disponíveis para cada subcomitê – foi usado o *Wagging*, pois há mais benefícios manter todos os exemplos no conjunto de treinamento.

Em resumo, o MultiBoosting [Web00] pode ser considerado como SMCs *Wagging* formados pelo AdaBoost.

Da mesma forma que o DECORATE, o MultiBoosting não utilizou RNAs como classificador base. Quanto aos seus resultados, na maior parte das bases de dados testada, o desempenho foi próximo ao alcançado pelo DECORATE. As bases de dados usadas foram extraídas do repositório UCI [AN07]: *Adult*, *Anneal*, *Audio*, *Autos*, *Balance-scale*, *Breast Cancer Slov.*, *Breast Cancer Wisc.*, *Cleveland*, *Credit Aust.*, *Discordant*, *Echocardiogram*, *German*, *Glass*, *Heart*, *Hepatitis*, *Horse-colic*, *House-votes-84*, *Hungarian*, *Hypo*, *Iris*, *Kr-vs-kp*, *Labor-neg*, *Lenses*, *Letter-recognition*, *Lymphography*, *New-thyroid*, *Phoneme*, *Pima diabetes*, *Promoters*, *Primary tumor*, *Soybean large*, *Segment*, *Sick*, *Splice junction*, *Tic-tac-toe* e *Waveform*.

3.4.6 WAVE (*Weight-Adjusted Voting for Ensembles of classifiers*)

O WAVE [KKMA11] é um método de classificação de SMC com pesos nos votos, o qual utiliza dois vetores de pesos, um para os classificadores e outro para as instâncias. O vetor de pesos das instâncias atribui valores mais altos a observações que são difíceis de classificar. O vetor de pesos dos classificadores dá maiores pesos àqueles que melhor classificam as instâncias difíceis de classificar. Os valores dos vetores são calculados em conjunto através de um procedimento iterativo. Isso significa que as instâncias com maiores pesos têm um papel mais significativo em determinar os pesos dos classificadores, e vice-versa. Os vetores convergem aos valores ótimos que podem ser diretamente calculados da matriz de desempenho dos classificadores. A predição final do SMC é obtida por votação usando o peso otimizado do vetor de classificadores.

Os testes foram realizados em 28 bases reais ou artificias, e as que seguem foram extraídas do repositório UCI [AN07]: *Breast Cancer Wisconsin*, *Liver disorders*, *Boston housing*, *Contraceptive method choice*, *Horse colic*, *Credit approval*, *Cylinder bands*, *Dermatology*, *German credit*, *Glass*, *Statlog (Heart)*, *Ionosphere*, *Iris*, *LED display domain*, *Pima indians dibetes*, *SPECTF heart*, *Statlog (Vehicle Silhouettes)*, *Congressional voting records* e *Vowel recognition*. Os resultados mostram que o WAVE supera consistentemente o *Bagging*, tende a superar o *Boosting* e possui desempenho semelhante ao método *Random Forest*.

3.5 Considerações Finais

Neste capítulo os Comitês de Classificadores foram apresentados. Através de uma breve introdução foi possível conhecer sobre sua motivação como também de sua composição. Foi abordado brevemente também uma das características mais importantes de um CC: a diversidade de seus membros. Na literatura é possível encontrar várias medidas de diversidade, porém ainda não há um estudo conclusivo que prove que uma medida venha a ser melhor que as demais [MM04]. Neste capítulo ainda foram apontadas cinco maneiras de se conseguir diversidade no comitê.

Foram descritas também duas abordagens gerais da combinação de classificadores: fusão e seleção. A primeira abordagem é a mais encontrada na literatura. A última é

menos utilizada, entretanto não menos eficiente. A abordagem da seleção foi a utilizada neste trabalho.

Por fim, três métodos clássicos e seis outros métodos foram descritos. Devido à eficiência e simplicidade dos métodos clássicos é importante verificar se métodos novos conseguem superá-los. Além disso, o MoE foi escolhido pela sua semelhança à seleção de classificadores. Os demais métodos possuem semelhanças ao método proposto neste trabalho e possuem motivações diferentes em relação aos clássicos. Também, em vários casos, os demais métodos superarem os clássicos. Portanto, é importante que na proposição de um novo método, o mesmo seja comparado a outros presentes na literatura.

4

Método Proposto

O método proposto neste trabalho possui duas versões. A primeira foi denominada BMG-GAVS (*Grouping Genetic Algorithm for Blockmodeling and Vote System*). A segunda versão foi denominada BMG2GA (*Grouping Genetic Algorithm for Blockmodeling and Ensemble Genetic Algorithm*). A principal diferença entre os mesmos se dá no ajuste dos parâmetros. Enquanto que no BMGGAVS o ajuste foi feito manualmente, no BMG2GA o ajuste é automático.

Além do ajuste dos parâmetros, outra diferença entre as versões do método se dá na seleção dos classificadores. No BMGGAVS a seleção é feita através de um sistema simples de voto. No BMG2GA a seleção é feita através de uma busca pela melhor combinação de classificadores únicos. Além disso, há diferenças no processamento que ocorre antes do agrupamento dos dados. Entretanto o agrupamento em si é feito de forma similar.

Para o detalhamento do método proposto, é importante esclarecer que antes da execução do mesmo, em suas duas versões, os dados são divididos aleatoriamente em conjuntos e subconjuntos, como mostrado na Figura 4.1. Apesar da divisão ser aleatória, há um cuidado sobre a proporcionalidade das classes. Dessa forma, os conjuntos e subconjuntos terão um número proporcional de representantes de cada classe. Os conjuntos são o de treinamento e teste final. O conjunto de treinamento é dividido em três subconjuntos: treinamento, validação e validação-extra. Este último subconjunto foi usado para testes durante o treinamento dos classificadores. O conjunto de teste final foi usado para calcular o erro do método proposto somente após toda a fase de treinamento.

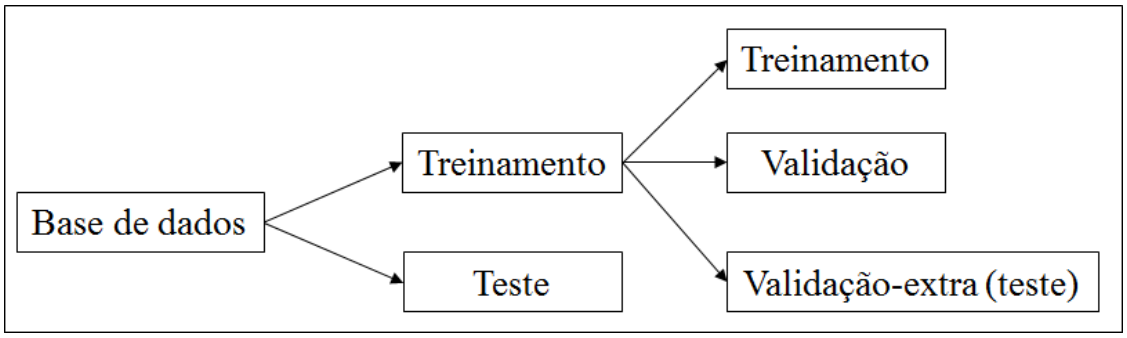

Figura 4.1 Divisão da base de dados em conjuntos e subconjuntos

O detalhamento de ambas as versões segue nos próximos tópicos.

4.1 BMGGAVS

O BMGGAVS é um método que, após um pré-processamento, pode ser descrito em duas fases. Na primeira fase o BM-GGA é usado para agrupar os dados do subconjunto de treinamento. Após o agrupamento os blocos são preenchidos com os dados dos subconjuntos de validação e validação-extra. Na segunda fase os classificadores são treinados e então, através de um sistema de votação simples cada *cluster* tem para si um classificador designado.

O algoritmo é descrito em pseudocódigo na Figura 4.2 e no fluxograma da Figura 4.3.

4.1.1 Pré-processamento

Após a definição dos conjuntos e subconjuntos a construção da matriz de adjacência tem início. A matriz de adjacência é a matriz construída a partir do grafo que representa as ligações entre os padrões. No BMGGAVS a maior distância entre os padrões é encontrada e definida como a distância padrão d. Após definida a distância padrão, um limite l é definido sobre d, sendo $l < d$. Esta relação pode ser melhor compreendida através da Figura 4.4. A partir de então todas as distâncias entre os padrões são verificadas, ou seja, para n padrões, são calculadas todas as distâncias euclidianas d_{ij}, com $i, j = 1, ..., n$.

```
1:  Execute o agrupamento com BM-GGA nos dados de treinamento;
2:  enquanto qualquer cluster < 5 nodos
3:      Adicione cada nodo deste cluster ao respectivo cluster mais
        próximo;
4:      Atualize o valor dos centroides;
5:  fim
6:  para cada cluster
7:      enquanto proporção não está completa
8:      Encontre os padrões de validação e validação-extra mais
        próximos;
9:      Insira-os no cluster;
10:     Atualize o valor do centroide;
11:     Encontre o padrão de teste final mais próximo e insira-o no
        cluster;
12:     fim
13: fim
14: enquanto qualquer padrão de validação ou validação-extra estiver
    sem cluster
15:     para cada padrão
16:         Encontre o cluster mais próximo;
17:         Insira este padrão no cluster;
18:         Atualize o centroide do cluster;
19:     fim
20: fim
21: para cada cluster
22:     Crie e execute uma rede MLP e um Perceptron;
23:     Inicie a votação (votos 1 a 6);
24:     se há empate
25:         Duplique o primeiro voto;
26:     fim
27:     Assinale a este cluster o classificador vencedor;
28: fim
```

Figura 4.2 Pseudocódigo do BMGGAVS

Caso $d_{ij} \leq l$, considera-se haver uma relação entre os padrões i e j. Devido ao fato de o limite ser o mesmo para todas as instâncias, na matriz de adjacência haverá a indicação

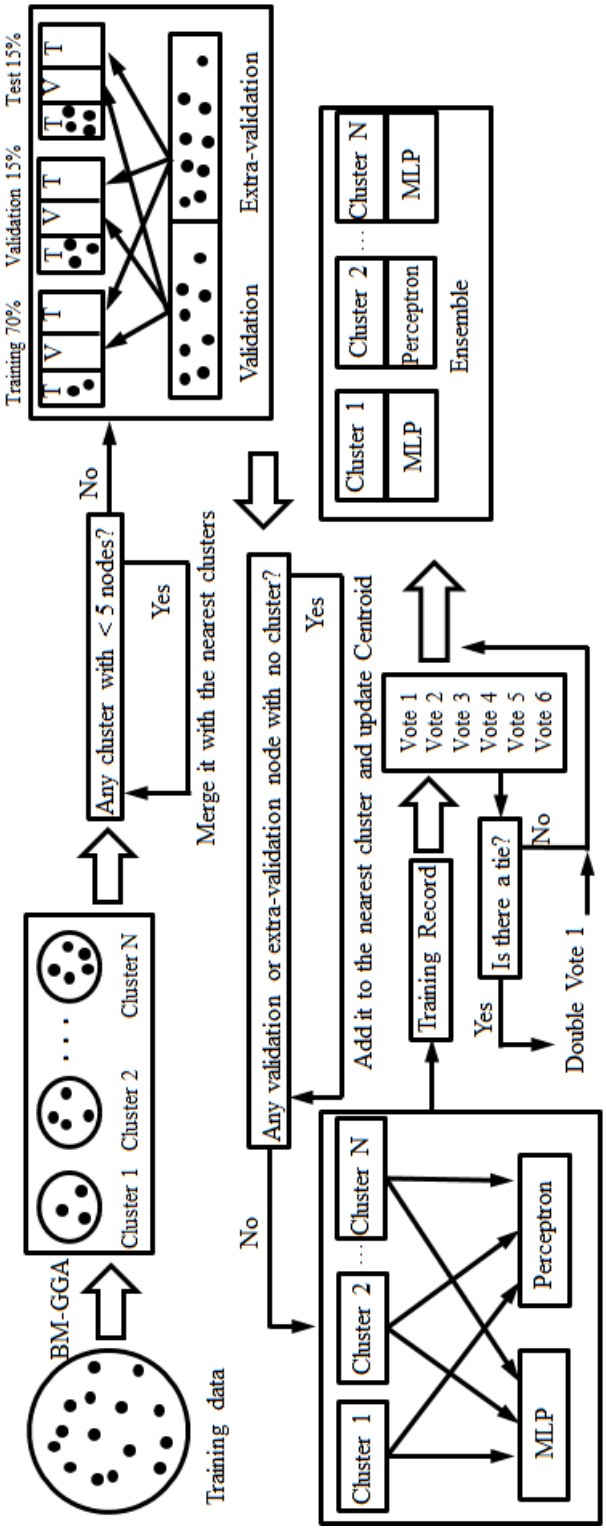

Figura 4.3 Fluxograma do BMGGAVS

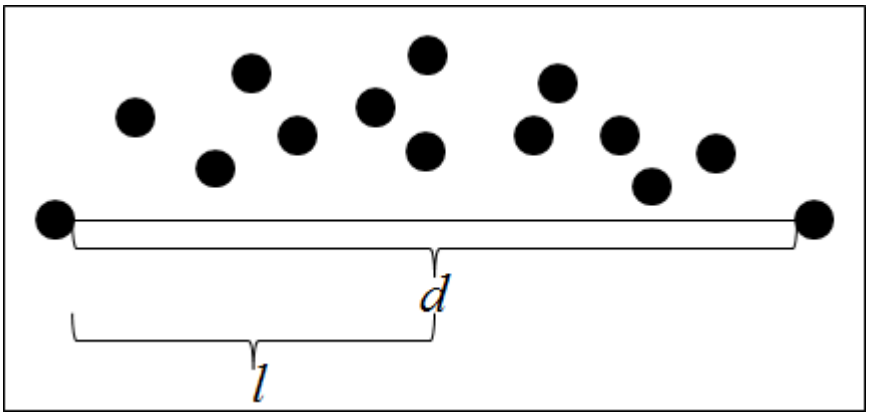

Figura 4.4 Distância padrão d e limite l definido sobre d

de relação entre os padrões i e j como também entre j e i. No caso de d_{ij} ser maior que l não haverá relação entre os padrões.

Montada a matriz de adjacência, é possível dar início ao BMGGAVS.

4.1.2 Primeira fase

O BM-GGA atua sobre o subconjunto de treinamento. Após o agrupamento há uma verificação sobre o tamanho dos *clusters*. O tamanho mínimo foi fixado em cinco padrões. Isto se deve ao fato de posteriormente os dados para validação e teste do *cluster* serem adicionados automaticamente, seguindo uma proporção. Caso o *cluster* tenha menos de cinco padrões, nenhum dado de validação e teste será adicionado posteriormente. Devido a isto, sempre que houver um *cluster* com menos de cinco membros, cada membro será realocado, individualmente, para o *cluster* mais próximo. Após a dissolução do *cluster* pequeno, os centroides dos *clusters* restantes são atualizados.

Terminada a verificação dos tamanhos dos *clusters*, novos padrões são adicionados aos blocos formados. Essa adição é feita proporcionalmente ao tamanho dos *clusters*. A proporção segue a taxa de 70% para treinamento, 15% para validação e 15% para teste. É necessário reforçar que os dados para teste serão preenchidos do subconjunto de validação-extra.

A adição dos novos padrões é feita da seguinte forma: para cada *cluster* são verificados os padrões de validação e validação-extra mais próximos (apenas um de cada). Quando

achados, os mesmos são inseridos no *cluster*. Este, por sua vez, tem o valor de seu centroide atualizado. Após a atualização do centroide o padrão mais próximo do conjunto de teste final é encontrado e também adicionado, em separado, ao *cluster*. Este padrão não exercerá qualquer influência sobre o *cluster*, mas será usado posteriormente para medir a precisão do classificador responsável por este bloco. O processo se repete até que todos os blocos sejam preenchidos.

O próximo passo é verificar se há padrões de validação ou validação-extra que não tenham sido inseridos em qualquer *cluster*. Caso positivo, para cada padrão é encontrado o *cluster* mais próximo, e o padrão é inserido nele. A cada inserção o *cluster* que recebe o novo membro tem seu centroide atualizado. O procedimento se repete até que todos os padrões sobressalentes sejam assinalados a algum *cluster*. Nesta etapa é possível que somente um *cluster* receba dados extras.

4.1.3 Segunda Fase

Na segunda fase os classificadores de cada tipo são construídos e treinados. No BMG-GAVS apenas dois tipos foram usados: Perceptron e MLP (de dez ou quinze nodos na camada escondida[1]). O número de classificadores de cada tipo é equivalente ao número de blocos, i.e., tendo o BM-GGA retornado k blocos, k classificadores de cada tipo são construídos, um para cada *cluster*. Cada classificador irá treinar somente no seu respectivo *cluster*, fazendo-o conhecedor apenas das características de um único bloco de dados.

Após o treinamento, uma votação é iniciada para a seleção dos classificadores. A votação é simples e rápida, pois leva em consideração apenas informações já disponíveis e sem pesos. O sistema de votação leva em conta seis informações: (1) tamanho do conjunto de treinamento ou teste do *cluster*; (2) erro do classificador após o treino; (3) desempenho do classificador após o treino; (4) desempenho de treinamento do classificador na melhor época; (5) desempenho de validação do classificador na melhor época e (6) desempenho de teste do classificador na melhor época.

Para cada informação um dos classificadores receberá um voto de acordo com um critério estabelecido (e.g., receberá o voto o classificador que possuir o maior valor). Os

[1] As quantidades de nodos da camada escondida foram escolhidas ao acaso.

critérios serão abordados no Capítulo 5.

O sistema de votação pode ser definido mais formalmente como segue. Seja t o tipo do classificador, c o classificador e k a quantidade de *clusters*; c_{ij} ($i = 1,...,t$ e $j = 1,...,k$) receberá o voto de acordo com o critério estabelecido para as informações 2 a 6.

A informação 1 foi excluída no parágrafo anterior pelo fato de ter somente um critério: sempre que o conjunto de treinamento ou teste for menor ou igual a oito, o classificador do tipo Perceptron receberá o voto. Do contrário o classificador do tipo MLP receberá o voto.

Dado que o número de votos é par, é possível que haja empate entre os classificadores. Neste caso o desempate é feito duplicando o primeiro voto. Como haverá apenas dois classificadores competindo em cada *cluster*, não há o risco de não haver desempate.

Após o treinamento e seleção dos classificadores a fase de teste é realizada. Entretanto desta vez são utilizados somente os dados do conjunto de teste final, os quais, até então, são desconhecidos pelos classificadores. Cada classificador será exposto aos dados do teste final de seu respectivo *cluster* e então será possível determinar o erro final do comitê.

O cálculo do erro final é a soma do número de padrões classificados erroneamente em cada *cluster* dividido pelo número total de padrões. Seja e_k o número de padrões de teste final do cluster k erroneamente classificados. Seja n o número total de padrões de teste final. O desempenho do comitê se dará por:

$$erro\,final = \frac{\Sigma_1^k e_k}{n} \tag{4.1}$$

4.2 BMG2GA

O BMG2GA, assim como o BMGGAVS, tem um pré-processamento e posteriormente duas fases de operação. O pré-processamento contém uma modificação feita em relação à primeira versão do método. No BMG2GA o limite de distância calculado para se definir a vizinhança de cada padrão passa a ser calculado automaticamente. Na primeira fase a modificação foi feita no acréscimo proporcional de padrões nos *clusters*.

```
1:  Execute o agrupamento com BM-GGA com os dados de treinamento;
2:  enquanto qualquer cluster < 5 nodos
3:      Adicione cada nodo deste cluster ao respectivo cluster mais
        próximo;
4:      Atualize o valor dos centroides;
5:  fim
6:  para cada cluster
7:      enquanto proporção não está completa
8:          Encontre os padrões de validação e validação-extra mais
            próximos;
9:          Insira-os no cluster;
10:         Atualize o valor dos centroides;
11:     fim
12: fim
13: enquanto qualquer padrão de validação ou validação-extra
    estiver sem cluster
14:     para cada padrão
15:         Encontre o cluster mais próximo;
16:         Insira este padrão no cluster;
17:         Atualize o centroide do cluster;
18:     fim
19: fim
20: para cada cluster
21:     Crie e execute um classificador de cada tipo;
22:     Guarde os erros de treinamento, validação e teste;
23:     Execute Ensemble GA;
24: fim
```

Figura 4.5 Pseudocódigo do BMG2GA

A segunda fase teve mais modificações. De início, o número de classificadores base teve seu número aumentado, de dois[2] para sete. Logo após, a modificação veio na seleção de classificadores. Em vez de um sistema de votos, a seleção é feita através de uma busca por AG. Esta busca tem por finalidade retornar a melhor combinação de classificadores únicos que consigam o menor erro global (desta vez somando-se os erros de treinamento, validação e teste).

A Figura 4.5 contém o algoritmo em pseudocódigo do BMG2GA.

[2]Apesar de haver três classificadores base, o método usou apenas dois de cada vez.

4.2.1 Pré-processamento

No BMGGAVS a definição do limite é feita manualmente. Em resumo, um limite sobre uma distância padrão é definida e diz-se que há *relacionamento* entre dois padrões caso sua distância euclidiana seja menor ou igual ao valor do limite.

Na evolução do método o princípio para definir o *relacionamento* entre dois padrões continuou o mesmo, contudo o limite passou a ser calculado automaticamente de acordo com as características da base de dados.

O raciocínio segue o exposto na Seção 1.2, ou seja, é esperado que normalmente padrões da mesma classe estejam mais próximos entre si devido às características que definem a classe. Ao mesmo tempo há uma busca para que os *clusters* não sejam muito pequenos e também não muito grandes.

Procurou-se então definir uma fórmula que mapeasse as particularidades de cada padrão de forma a definir de maneira mais verossímil sua vizinhança. Ou seja, o limite agora é definido para cada amostra, individualmente.

A primeira fórmula proposta foi a da média aritmética de cada padrão:

$$l_p = \frac{\Sigma_{j=1}^{n} d_{pj}}{n} \tag{4.2}$$

Na equação (4.2), l_p é o limite para o padrão p, d_{pj} é a distância entre os padrões p e j, e n é o número total de padrões.

É importante lembrar que o grafo que está sendo construído é não direcional. Isto significa que se um padrão i se liga a um padrão j, j também se liga a i, ainda que a distância entre i e j supere o limite estabelecido para j. Esta relação está exemplificada na Figura 4.6.

Contudo, a equação (4.2) não se mostrou suficiente para mapear as particularidades. Caso a média de distância seja relativamente alta, o limite também o será. Isto faz com que muitos padrões sejam considerados vizinhos do padrão em questão numa situação onde não seria verossímil tal relacionamento, dado que os padrões já estão separados

por uma relativa distância. Também, no caso de amostras de duas classes estarem muito próximas umas das outras, seria interessante que naquela região amostral houvesse um *cluster* maior. O tamanho maior do grupo daria mais exemplos ao classificador de forma que este pudesse aprender melhor as particularidades dos dados. Um tamanho menor, neste caso, poderia resultar em um aumento de erro, pois os *clusters* poderiam estar bastante desbalanceados. Na Figura 4.7 é mostrado um exemplo do que se considera um agrupamento automático ideal. Utilizando somente a equação (4.2) não é possível criar grupos menores quando as distâncias são maiores, nem grupos maiores quando as distâncias são menores.

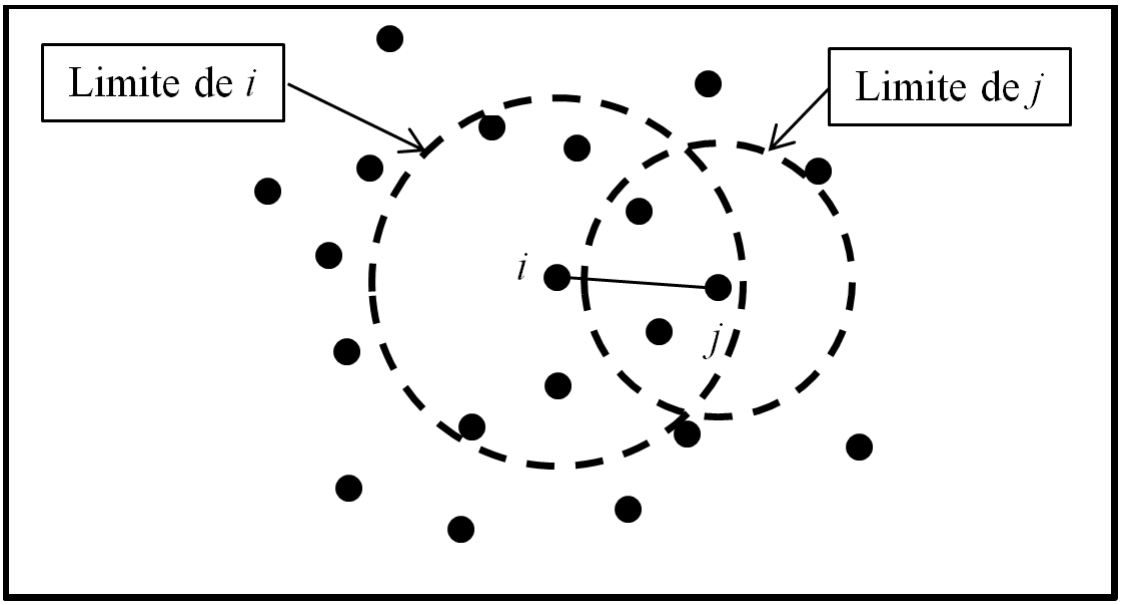

Figura 4.6 No grafo j se liga a i mesmo que i esteja fora do limite

A solução encontrada foi dividir 1 pela média, como na equação (4.3). Assim, quanto menor a média, maior o limite, e quanto maior a média das distâncias do padrão, menor o limite.

$$l'_p = \frac{1}{l_p} \tag{4.3}$$

Mesmo com as modificações realizadas o método não conseguiu levar em conta características dos dados que contribuíssem para uma melhora de seu desempenho. Desse modo, foi necessário uma investigação sobre a distribuição dos padrões em relação ao padrão observado. Dessa forma, caso o desvio-padrão das distâncias a partir de uma

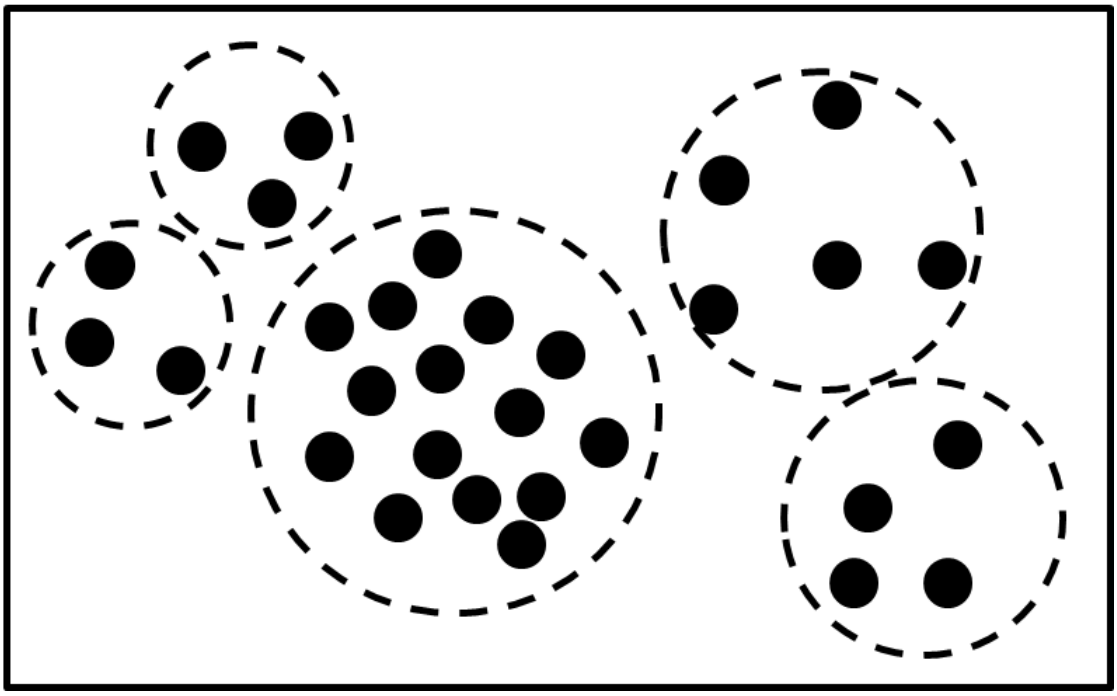

Figura 4.7 Exemplo de agrupamento automático ideal

amostra seja alto, as demais amostras estariam heterogeneamente espalhadas em relação a ela. Isto pode dificultar a aprendizagem de um classificador. Do contrário a distribuição dos demais padrões se mostraria mais uniforme. Isto refletiria uma padronização possivelmente mais fácil de ser detectada por um classificador.

Portanto, o objetivo é maximizar o desvio padrão das instâncias a partir de uma amostra, de modo que o número de vizinhos no grafo também seja maximizado. Da mesma forma, ao minimizar o desvio padrão das distâncias a partir de uma amostra, sua vizinhança deverá ser menor. O acréscimo do desvio-padrão se deu como segue:

$$dv_p = \sqrt{\frac{\Sigma_{j=1}^{n}(d_{pj} - l_p)^2}{n-1}} \tag{4.4}$$

$$l'_p = \frac{1}{l_p - dv_p} \tag{4.5}$$

Nas equações 4.4 e 4.5, dv_p é o desvio padrão das distâncias a partir da amostra p; n é o número de padrões (amostras), l_p é o limite da amostra p como calculado na equação (4.2) -- ou seja, é a média de distâncias de p a todas as outras amostras -- e l'_p é

o novo limite de distância.

Após o cálculo de l'_p, com $p = 1, ..., n$, o grafo e a matriz de adjacência são construídos.

4.2.2 Primeira fase

O agrupamento acontece como no BMGGAVS. O BM-GGA é executado e posteriormente os *clusters* com menos de cinco amostras são extinguidos.

Na complementação dos *clusters* com dados de validação, validação-extra e teste final há uma pequena modificação. No BMGGAVS os dados de teste final são inclusos após a atualização dos centroides. Isso pode fazer com que alguns padrões que seriam incluídos originalmente em um *cluster* sejam inseridos em outro. Portanto, antes de os centroides serem atualizados, os dados de teste final são inseridos nos *clusters*. Os dados de teste final continuam a não influenciar no cálculo de atualização dos centroides.

4.2.3 Segunda fase

O início da segunda fase do método teve como modificação a inserção de novos tipos de classificadores. Enquanto no BMGGAVS havia somente Perceptron e MLP — de dez ou quinze nodos na camada escondida -–, no BMG2GA há Perceptron, MLP5 (MLP com cinco nodos na camada escondida), MLP10, MLP15 e ELM, com cinco, dez e quinze nodos[3].

Além do incremento de novos tipos de classificador, o sistema de votação foi substituído por uma busca via Algoritmo Genético, chamada *Ensemble GA*. Para alcançar um bom resultado com o sistema de votos, foram necessários ajustes de parâmetros. Foi preciso definir para os votos 4 a 6, o critério de pior desempenho ao mesmo tempo em que nos demais votos se buscava o critério de melhor desempenho. Como não é possível prever o erro final com as mudanças feitas nos parâmetros, foi realizada uma busca por uma melhor combinação de classificadores de modo a reduzir os erros de treinamento, validação e teste. Espera-se então que um comitê que ache os menores erros consiga obter uma melhor generalização nos dados de teste final.

[3] O número de nodos nas camadas escondidas foi escolhido ao acaso.

O cromossomo no *Ensemble GA* possui k genes, sendo k o número de *clusters*. Os tipos de classificadores foram mapeados de acordo com a Tabela 4.1.

Tabela 4.1 Classificadores base e suas respectivas referências no *Ensemble GA*

Classificador	Referência
Perceptron	1
MLP5	2
MLP10	3
MLP15	4
ELM5	5
ELM10	6
ELM15	7

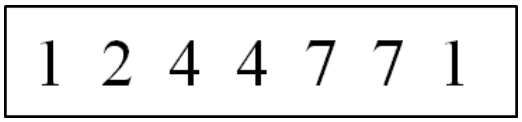

Figura 4.8 Exemplo de cromossomo do *Ensemble GA*

Logo, o cromossomo mapeia os classificadores selecionados para cada *cluster*. Na Figura 4.8, $k = 7$, portanto existem 7 *clusters*. É possível perceber que dois classificadores do tipo Perceptron foram escolhidos para o primeiro e último *clusters*; um classificador do tipo MLP5 foi selecionado para o segundo *cluster*; dois classificadores do tipo MLP15 foram selecionados para os *clusters* 3 e 4 e dois classificadores do tipo ELM15 foram escolhidos para os *clusters* 5 e 6. A função de aptidão é como segue:

$$f = 3 - \left(\frac{\Sigma_1^k e_{tr_k}}{n_{tr}} + \frac{\Sigma_1^k e_{v_k}}{n_v} + \frac{\Sigma_1^k e_{t_k}}{n_t}\right) \tag{4.6}$$

Na equação (4.6), e_{tr_k} é o número de padrões erroneamente classificados no treinamento do cluster k e n_{tr} é o número de padrões de treinamento; e_{v_k} é o número de padrões erroneamente classificados na validação do cluster k e n_v é o número de padrões de validação; e_{t_k} é o número de padrões erroneamente classificados no teste do cluster k e n_t é o número de padrões de teste (tirados do subconjunto de validação-extra).

A melhor combinação de classificadores é a escolhida como o comitê e então o método parte para o teste final. O erro do teste final é calculado com a equação (4.1).

5

Experimentos e Discussões

Neste capítulo os experimentos realizados são detalhados e discutidos. Os experimentos para cada versão do método são descritos em duas seções distintas.

Na primeira seção são mostrados os valores dos parâmetros para o BMGGAVS, como também as tabelas e gráficos dos resultados alcançados. Posteriormente há uma breve discussão sobre os resultados obtidos.

Na segunda seção há detalhes sobre os parâmetros do BMG2GA e suas respectivas tabelas de resultados e gráficos. Houve mais experimentos neste ponto devido à busca de uma melhor configuração que resultasse em um melhor resultado do comitê de classificadores. Ao final de cada conjunto de resultados há discussões sobre os mesmos.

Os resultados são mostrados em tabelas de comparações de média de erros, médias de tempo e em gráficos. Nas tabelas os melhores resultados são mostrados em **negrito** enquanto os segundos melhores resultados são mostrados sublinhados. Quando aparecem traços '–', significa que para aquela base de dados não foi fornecido qualquer valor para comparação. O desvio-padrão é apresentado entre parênteses e em itálico.

Os experimentos foram executados sobre bases de dados retiradas do UCI [AN07], repositório eletrônico. A seguir, as bases de dados são apresentadas.

5.1 Bases de dados

Ao todo foram doze bases de dados reais e diferentes. Nove delas foram usadas por ambas as versões do método, e as três adicionais foram usadas somente nos testes do BMG2GA.

A seguir as bases são listadas e brevemente descritas. Cada base de dados terá seu título e em seguida, entre parênteses, o nome abreviado usado nas tabelas e discussões.

1. **Câncer de mama *Winsconsin*** (*Cancer*): Esta base de dados foi obtida da *University of Wisconsin Hospitals*. O que se busca é aprender se o câncer encontrado é benigno ou maligno.

2. **Aprovação de crédito na Austrália** (*Card*): Esta base está relacionada a requisições de cartão de crédito. Todos os nomes dos atributos, como também seus valores, foram renomeados de forma a perder seu significado original; desta forma, pretende-se proteger a confidencialidade dos dados. O objetivo é liberar o crédito da forma mais segura ao credor, ou seja, com o menor risco de inadimplência.

3. **Diabetes de índios Pima** (*Diabetes*): Esta base contém os dados de descendentes de índios Pima. Todos os pacientes são do sexo feminino e têm pelo menos vinte e um anos. O que se busca é separar, baseado nos atributos, quais pessoas possuem diabetes e quais não possuem.

4. **Tipos de vidro** (*Glass*): Este problema foi motivado pela investigação criminológica, pois numa cena de crime o vidro deixado pode ser usado como uma evidência, caso seja corretamente identificado.

5. **Doença cardíaca** (*Heart*): Esta base de dados é a junção de quatro outras menores. Os dados foram coletados de quatro diferentes localizações: *Cleveland Clinic Foundation*; *Hungarian Institute of Cardiology*, em Budapeste; *V.A. Medical Center*, em Long Beach, Califórnia e *University Hospital*, em Zurique, Suíça. O formato das instâncias é padronizado nas quatro bases de dados.

6. **Doença cardíaca em Cleveland** (*Heartc*): Esta base de dados compreende apenas os dados obtidos em *Cleveland Clinic Foundation*, i.e., uma das quatro bases de dados que compõem a base *Heart*.

7. **Cólica de cavalo** (*Horse*): Nesta base de dados busca-se determinar o tipo de lesão que está causando dor ao cavalo.

8. **Segmentação de imagem** (*Segment*): Nesta base as instâncias foram retiradas aleatoriamente de um banco de dados de sete imagens, as quais foram manualmente segmentadas para criar uma classificação para cada pixel.

9. **Soja** (*Soybean*): Nesta base procura-se identificar, de acordo com os atributos, que tipo de dano a soja sofreu.

10. **Doença da Tireoide** (*Thyroid*): Esta base está relacionada a doenças na tireoide. De acordo com os atributos procura-se classificar cada padrão como normal ou apresentando hipertireoidismo ou hipotireoidismo.

11. **Reconhecimento de vogal** (*Vowel*): Com esta base de dados procura-se o reconhecimento de onze vogais estáveis do Inglês Britânico. O problema foi montado ao se registrar quinze locutores falando, cada um, onze vogais, seis vezes cada vogal.

12. **Levedura** (*Yeast*): A problemática para essa base é predizer os locais onde se localizarão proteínas em células.

Na Tabela 5.1 é possível encontrar as características das bases de dados, i.e., seu número de instâncias, classes e atributos.

Tabela 5.1 Número de instâncias, classes e atributos das bases de dados

	Número de instâncias	Número de atributos	Número de classes
Cancer	699	9	2
Card	690	15	2
Diabetes	768	8	2
Glass	214	9	6
Heart	920	35	2
Heartc	303	35	2
Horse	368	28	3
Segment	2310	19	7
Soybean	307	35	19
Thyroid	7200	21	3
Vowel	990	10	11
Yeast	1484	8	10

5.2 BMGGAVS

5.2.1 Parâmetros e critérios

Como descrito na Seção 4.1, antes do início do BMGGAVS ocorre a construção de uma matriz de adjacência. Quando a mesma é construída para problemas sociais os pesquisadores normalmente já possuem os grafos das relações sociais existentes [SLA13]. No caso de bases de dados não sociais não há essa informação.

A solução proposta foi encontrar a maior distância entre os padrões da base de dados e defini-la como distância padrão d:

$$d = max(d_{ij}) \tag{5.1}$$

Na equação (5.1) d_{ij} é a distância entre os padrões i e j, com $i, j = 1, ..., n$. Sobre essa distância um limite é imposto.

Contudo não se sabe qual limite de distância é suficiente ou recomendado para indicar uma *relação* entre os padrões. Deste modo, foram testados onze limites, começando em 50%, até 2%, decaindo (exceto entre o penúltimo e último limite) 5% por vez. Estes são: 50%, 45%, 40%, 35%, 30%, 25%, 20%, 15%, 10%, 5% e 2%. Para cada um dos limites, um grafo e sua respectiva matriz de adjacência foram construídos.

Supondo o limite de 50% sobre d e n padrões. Um padrão é escolhido, e este será ligado a todos os outros padrões que estiverem a um raio de 50% de d a partir dele. Outro padrão é escolhido e a mesma verificação é feita; assim todos os demais padrões próximos (i.e., cuja distância for menor ou igual ao limite estabelecido) serão assinalados como tendo *relação* com ele. A iteração se repete até que os n padrões tenham conhecimento de quais serão seus vizinhos no grafo. A Figura 5.1 contém o exemplo de um padrão i, cuja distância ao padrão j coincide de ser a distância d. O limite l é 50% de d, e através de linhas tracejadas o padrão i se liga à sua vizinhança.

Portanto, é possível notar que, dependendo da disposição espacial, alguns padrões terão poucos vizinhos enquanto outros terão muitos. É possível que dois ou mais padrões

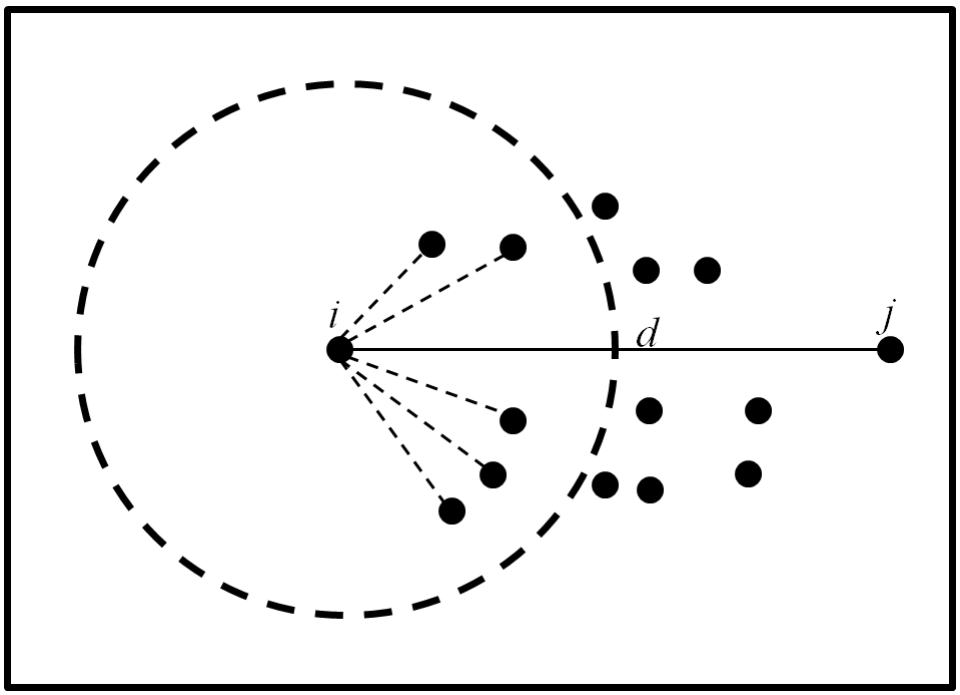

Figura 5.1 Exemplo de um padrão se ligando a todos os outros que estão a 50% de d

tenham exatamente os mesmos vizinhos, caso estejam suficientemente próximos um do outro.

Após a construção das matrizes de adjacência, cada matriz passa pelo processo de agrupamento, que é realizado pelo BM-GGA.

Os valores dos parâmetros neste ponto seguem o proposto em [JBR10]: a população no Algoritmo Genético é de cem indivíduos, o número de gerações é vinte, a taxa de recombinação, ou cruzamento, é de 100% e não há mutação. Além disso, a substituição da população é geracional, ou seja, a população de prole substitui por completo a população anterior.

Finalizada a primeira fase, os blocos devem estar formados com todos os seus devidos padrões. A segunda fase, de treinamento e teste, é iniciada. Neste ponto foram usados Perceptrons e MLPs. Os valores dos parâmetros do Perceptron usados foram os valores padrões fornecidos pelo *software* MATLAB. Dentre eles, a função de treinamento cíclica, o número máximo de épocas sendo 1000 e função de ativação degrau. Da mesma forma, os valores dos parâmetros do MLP usados são também os valores padrões fornecidos pelo MATLAB. Entre eles a função de treinamento *Levenberg-Marquardt*, o número máximo

de épocas sendo 1000, e função de ativação sigmoidal nos nodos da camada escondida. O número de nodos na camada escondida foi escolhido como sendo 10 ou 15.

5.2.2 Resultados e discussões

Devido ao fato do método proposto ser estocástico, uma melhor verificação de seus resultados se dará com cada experimento sendo repetido diversas vezes, e então, pela média e desvio-padrão dos resultados, é possível determinar seu desempenho. A Figura 5.2 contém um fluxograma do experimento realizado com o BMGGAVS.

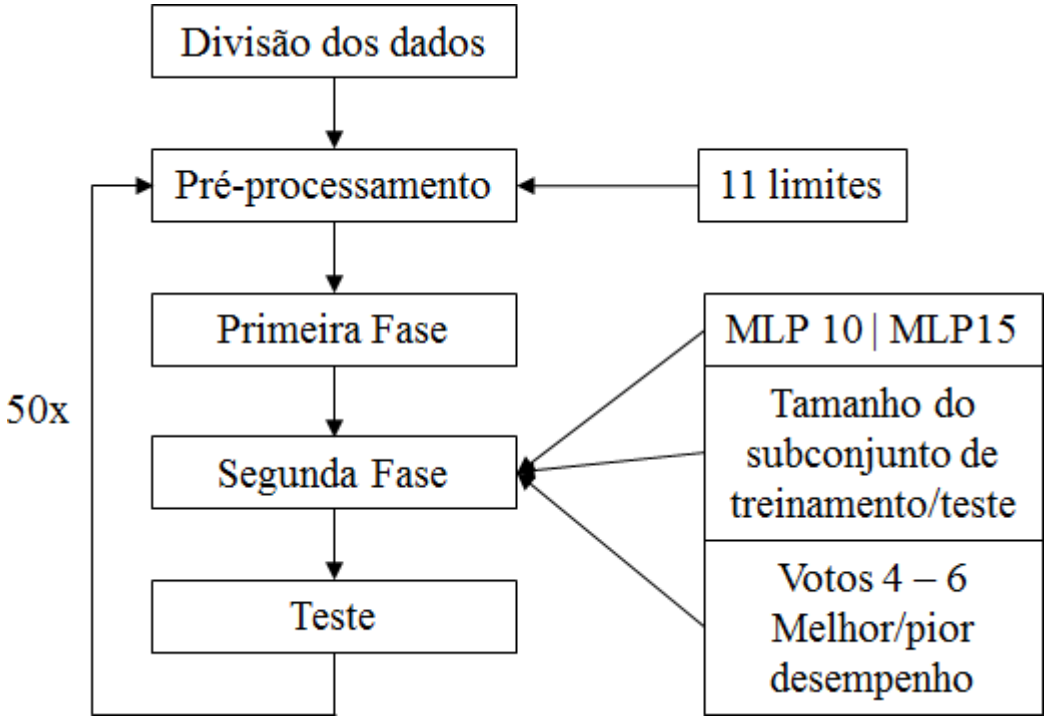

Figura 5.2 Fluxograma do experimento realizado com o BMGGAVS

Para cada limite estabelecido, a base de dados é dividida em conjuntos e subconjuntos. Cada divisão é independente das demais. Além disso, um grafo e, posteriormente, uma matriz de adjacência, foram gerados.

De posse da matriz de adjacência, o método é executado cinquenta vezes sobre os subconjuntos estabelecidos. Reforçamos que neste ponto há somente uma única divisão de dados para cada limite. A divisão é realizada antes do início do método e é utilizada em todas as iterações. Dessa forma, em cada uma das cinquenta repetições o método

proposto terá acesso aos mesmos subconjuntos.

Todas as possíveis combinações de parâmetros foram testadas, i.e., diferentes limites com diferentes tamanhos de camada escondida das MLPs (dez ou quinze nodos). Levou-se em consideração também o tamanho do subconjunto de treinamento ou subconjunto de teste. Tais combinações resultaram em quarenta e quatro resultados para cada base de dados.

Além das combinações acima, nos votos de quatro a seis, do sistema de votação, duas configurações(ou critérios) foram testadas: levando em conta o melhor desempenho e levando em conta o pior desempenho. Dessa forma, sobe para oitenta e oito o número de resultados obtidos para cada base de dados.

Os experimentos foram executados em computadores diferentes com configurações diferentes. O método foi implementado no MATLAB e algumas partes do código foram executadas em paralelo. Quanto às comparações, quando possível, testes estatísticos com $\alpha = 5\%$ foram feitos. Sendo os dados normais um teste *t-student* foi executado, do contrário um teste *rank-sum* de Wilcoxon [Wil45].

Os testes de hipóteses não foram realizados nas situações em que os demais autores não forneceram seus desvios-padrão, de forma a ser impossível recriar uma nova amostra com características semelhantes às suas. Nestes casos optou-se por comparações simples.

Nas Tabelas 5.2 e 5.3, são fornecidos os resultados respectivos para cada uma das configurações, divididas em dois grandes blocos: (1) usando classificadores do tipo Perceptron e MLP10 e (2) usando classificadores do tipo Perceptron e MLP15. Tais resultados foram obtidos após o treinamento e seleção, quando os dados do conjunto de teste foram expostos ao comitê formado.

As linhas nomeadas com TR e T correspondem aos subconjuntos de treinamento e teste, respectivamente. Em outras palavras, nas linhas TR o tamanho do subconjunto de treinamento foi levado em consideração no primeiro voto do sistema de votação. Nas linhas nomeadas com T, o primeiro voto foi dado levando em consideração o tamanho do subconjunto de teste.

Pela comparação das tabelas é possível perceber que, exceto na base *Vowel*, ao se levar em conta o pior desempenho nos votos 4 a 6, temos um melhor resultado. A

Tabela 5.2 Médias de erro de teste levando em consideração o melhor desempenho nos votos 4 – 6

	Limite		*Cancer*	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
MLP10	50%	TR	0,0266	0,2405	0,5235	0,2941	0,2028	0,2205	0,3361	0,1908	0,3884
		T	0,0250	0,2132	0,5228	0,2928	0,1870	0,2080	0,2951	0,1901	0,3727
	45%	TR	0,0098	0,2498	0,3576	0,5153	0,2739	0,2917	0,3594	0,0918	0,3814
		T	**0,0090**	0,2275	0,3471	0,5153	0,2609	0,2596	0,2972	0,0791	0,3806
	40%	TR	0,0667	0,2248	0,2973	0,5061	0,2181	0,2057	0,3600	0,0806	0,4503
		T	0,0674	0,2241	0,2949	0,5039	0,2029	0,2107	0,3324	0,0718	0,4439
	35%	TR	0,0567	0,2123	0,2927	0,3403	0,2216	0,3062	0,3131	0,0749	0,3549
		T	0,0552	0,2019	0,2726	0,3316	0,2137	0,2963	**0,2821**	**0,0670**	0,3488
	30%	TR	0,0557	0,2044	0,2762	0,3231	0,2311	0,2722	0,4638	0,1083	0,2922
		T	0,0520	0,1946	0,2653	0,3231	0,2430	0,2607	0,4536	0,1012	0,2859
	25%	TR	0,0447	0,1698	0,3420	0,4122	0,2285	0,2606	0,4094	0,1423	0,1977
		T	0,0404	**0,1663**	0,3367	0,4089	0,2123	0,2445	0,3777	0,1333	0,1933
	20%	TR	0,0447	0,2020	0,2938	0,3196	0,2569	0,3113	0,3914	0,1491	0,2079
		T	0,0410	0,1783	0,3015	0,3167	0,2209	0,3251	0,3486	0,1465	0,1900
	15%	TR	0,0403	0,2223	0,3018	0,3676	0,2188	0,2315	0,3697	0,0905	0,1684
		T	0,0388	0,2174	0,2944	0,3639	0,2147	0,2195	0,3321	0,0882	0,1595
	10%	TR	0,0738	0,2069	0,3268	0,3906	0,3026	0,3038	0,3984	0,1256	0,1819
		T	0,0677	0,2046	0,3303	0,3761	0,2936	0,3058	0,3791	0,1127	0,1790
	05%	TR	0,0633	0,1973	0,2912	0,3323	0,1998	0,2292	0,3742	0,1370	0,1753
		T	0,0633	0,1918	0,2992	0,3297	0,1842	0,2097	0,3598	0,1322	0,1654
	02%	TR	0,0504	0,1963	0,3702	0,3108	0,1824	0,3133	0,4095	0,1080	0,1762
		T	0,0413	0,1981	0,3616	0,3108	**0,1743**	0,3155	0,3732	0,0988	0,1749
MLP15	50%	TR	0,0473	0,1908	0,1842	0,3088	0,2040	0,2609	0,3981	0,0765	0,3580
		T	0,0469	0,1735	**0,1767**	0,3090	0,1895	0,2310	0,3451	0,0727	0,3570
	45%	TR	0,0439	0,1896	0,3062	0,4519	0,2346	0,2577	0,4349	0,1487	0,4989
		T	0,0412	0,1892	0,3046	0,4519	0,2319	0,2485	0,3817	0,1400	0,4969
	40%	TR	0,0449	0,2281	0,2714	0,3551	0,2298	0,2228	0,4301	0,2011	0,4958
		T	0,0444	0,2079	0,2606	0,3551	0,2163	0,2142	0,3936	0,1915	0,4905
	35%	TR	0,0324	0,2210	0,2824	0,2715	0,2405	0,2129	0,3865	0,1521	0,3375
		T	0,0286	0,2049	0,2673	**0,2681**	0,2346	**0,1847**	0,3357	0,1430	0,3298
	30%	TR	0,0288	0,2502	0,3272	0,4951	0,2444	0,2426	0,4496	0,1243	0,2738
		T	0,0250	0,2302	0,3244	0,4829	0,2268	0,2507	0,3943	0,1141	0,2653
	25%	TR	0,0469	0,2391	0,2981	0,3262	0,2797	0,2675	0,3626	0,0988	0,2238
		T	0,0415	0,2263	0,2724	0,3300	0,2647	0,2440	0,3219	0,0863	0,2171
	20%	TR	0,0677	0,1968	0,3147	0,3508	0,2073	0,2593	0,4070	0,1041	0,1842
		T	0,0647	0,1914	0,2941	0,3504	0,1980	0,2637	0,3782	0,0939	0,1865
	15%	TR	0,0403	0,2097	0,3318	0,3158	0,2619	0,2048	0,4404	0,1505	0,1587
		T	0,0352	0,1962	0,3198	0,3101	0,2542	0,1866	0,3862	0,1294	**0,1498**
	10%	TR	0,0292	0,1689	0,3058	0,3928	0,2134	0,2019	0,3648	0,1528	0,1677
		T	0,0228	0,1665	0,2823	0,3834	0,2107	0,1924	0,3220	0,1373	0,1644
	05%	TR	0,0748	0,2260	0,2886	0,4127	0,2750	0,3082	0,3739	0,1501	0,1894
		T	0,0748	0,2141	0,3018	0,4083	0,2666	0,2918	0,3172	0,1414	0,1825
	02%	TR	0,0385	0,1917	0,3087	0,4329	0,2235	0,2947	0,3598	0,1944	0,2277
		T	0,0399	0,1965	0,2816	0,4463	0,2287	0,2728	0,2989	0,1879	0,2273

Tabela 5.3 Médias de erro de teste levando em consideração o pior desempenho nos votos 4 – 6

	Limite		*Cancer*	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
MLP10	50%	TR	0,0235	0,2039	0,5071	0,2731	0,1815	0,2109	0,2405	0,1715	0,3214
		T	0,0506	0,1799	0,5103	0,2690	0,1784	0,1969	0,2331	0,1692	0,3142
	45%	TR	0,0091	0,2395	0,3262	0,5112	0,2677	0,2469	0,2155	0,0637	0,3519
		T	**0,0077**	0,2568	0,3241	0,5092	0,2534	0,2184	0,1997	0,0616	0,3519
	40%	TR	0,0741	0,2247	0,2490	0,4741	0,2042	0,1810	0,2619	0,0534	0,4058
		T	0,0736	0,2518	0,2418	0,4612	0,2519	0,1749	0,2473	0,0519	0,3986
	35%	TR	0,0477	0,1898	0,2456	0,2593	0,2227	0,2791	0,2206	0,0363	0,3073
		T	0,0583	0,1861	0,2192	0,2584	0,2177	0,2467	0,2184	**0,0299**	0,3095
	30%	TR	0,0467	0,1966	0,2509	0,3214	0,2372	0,2607	0,4281	0,0687	0,2583
		T	0,0575	0,1924	0,2281	0,3248	0,2979	0,2405	0,4129	0,0572	0,2643
	25%	TR	0,0422	0,1547	0,3273	0,3803	0,2000	0,2407	0,3177	0,1011	**0,1701**
		T	0,0394	0,1830	0,3296	0,3880	**0,1683**	0,2360	0,2985	0,1047	0,1916
	20%	TR	0,0348	0,1758	0,2937	0,3330	0,2202	0,3269	0,3065	0,1239	0,1731
		T	0,0299	0,1622	0,3160	0,3558	0,1886	0,3548	0,3011	0,1243	0,2897
	15%	TR	0,0380	0,2206	0,2973	0,3970	0,1931	0,2684	0,2895	0,0946	0,2086
		T	0,0401	0,2362	0,3011	0,4451	0,2113	0,2177	0,2857	0,0953	0,2617
	10%	TR	0,0656	0,1954	0,3221	0,3280	0,2821	0,3311	0,2952	0,0798	0,1953
		T	0,0657	0,1854	0,3697	0,3294	0,2703	0,3496	0,2759	0,0772	0,2476
	05%	TR	0,0635	0,1862	0,3182	0,2592	0,1781	0,2459	0,3184	0,0927	0,1908
		T	0,0635	0,2286	0,3232	0,2351	0,2278	0,2424	0,3109	0,0915	0,2413
	02%	TR	0,0355	0,1667	0,3453	0,2509	**0,1683**	0,3536	0,2782	0,0677	0,2525
		T	0,0340	0,2109	0,3501	0,2996	0,1836	0,4021	0,2738	0,0663	0,2693
MLP15	50%	TR	0,0426	0,1628	0,1714	0,3159	0,1740	0,2234	0,2828	0,0446	0,3213
		T	0,0369	0,1507	**0,1652**	0,3150	0,1692	0,2108	0,2715	0,0444	0,3193
	45%	TR	0,0399	0,1912	0,2997	0,4464	0,1970	0,2207	0,3381	0,0868	0,4638
		T	0,0449	0,2235	0,3137	0,4230	0,2103	0,2342	0,3316	0,0866	0,4638
	40%	TR	0,0428	0,1969	0,2550	0,3531	0,2080	0,2286	0,3591	0,1535	0,4154
		T	0,0959	0,1875	0,2247	0,2872	0,2549	0,2594	0,3565	0,1518	0,3997
	35%	TR	0,0234	0,1870	0,2768	**0,2173**	0,2361	0,1674	0,2763	0,1084	0,2950
		T	0,0223	0,1734	0,2711	0,2307	0,2431	**0,1417**	0,2594	0,1055	0,2939
	30%	TR	0,0205	0,2101	0,3244	0,4354	0,2157	0,2688	0,3272	0,0855	0,2558
		T	0,0145	0,2029	0,3186	0,4271	0,2599	0,2866	0,2959	0,0773	0,2557
	25%	TR	0,0312	0,2190	0,2716	0,2846	0,2624	0,2900	0,2675	0,0564	0,2120
		T	0,0292	0,2143	0,2126	0,3392	0,2580	0,2865	0,2525	0,0572	0,2375
	20%	TR	0,0606	0,1692	0,2797	0,3012	0,1876	0,2627	0,3499	0,0652	0,1981
		T	0,0907	0,1930	0,2498	0,2467	0,1840	0,2782	0,3151	0,0632	0,2192
	15%	TR	0,0372	0,1999	0,3297	0,2946	0,2413	0,2008	0,3226	0,0979	0,1799
		T	0,0428	0,1852	0,3182	0,2942	0,2456	0,2467	0,3118	0,0890	0,2266
	10%	TR	0,0119	**0,1365**	0,2826	0,3709	0,2044	0,1761	0,2852	0,0999	0,2212
		T	0,0209	0,1827	0,2614	0,4135	0,2090	0,1780	0,2736	0,0929	0,2737
	05%	TR	0,0787	0,2170	0,2964	0,3798	0,2657	0,2997	0,2235	0,1070	0,2274
		T	0,0787	0,2040	0,3422	0,3855	0,2634	0,2729	**0,1970**	0,0964	0,2518
	02%	TR	0,0358	0,1672	0,3128	0,4906	0,2133	0,2644	0,2533	0,1519	0,2696
		T	0,0352	0,2108	0,2701	0,4702	0,2247	0,2494	0,2253	0,1515	0,3060

diferença entre os resultados normalmente não é grande, pois em apenas duas bases há diferenças maiores que 5%. O que corrobora com isso é o fato de que, estatisticamente, em três bases (*Cancer*, *Diabetes* e *Heart*), os melhores resultados são iguais. Com isso, é possível realizar a primeira afirmação: a segunda configuração é [pouco] melhor que a primeira. Somente na base *Horse* o método produziu resultados muito diferentes quando se comparam as duas configurações.

Para as bases *Card*, *Glass*, *Heartc*, *Horse* e *Soybean* o segundo melhor resultado, da segunda configuração (Tabela 5.3), ainda consegue superar o melhor resultado da primeira configuração (Tabela 5.2). De forma parecida — mas invertendo as configurações -– acontece com a base *Vowel*. O segundo melhor resultado na primeira configuração (Tabela 5.2) é menor que o melhor resultado da segunda configuração (Tabela 5.3).

É possível então seguir para a segunda afirmação, complementar à primeira: a segunda configuração é suficientemente melhor que a primeira configuração.

Com as afirmações é possível observar um fato inusitado. Espera-se que os melhores classificadores no treinamento sejam os melhores no teste. Entretanto, neste trabalho, em cinco das nove bases, os classificadores que tiveram o pior desempenho no treinamento foram os que conseguiram generalizar melhor no teste. Isto corrobora com o fato de que ainda que os melhores classificadores sejam selecionados no treinamento, os mesmos não necessariamente serão os melhores para dados não vistos [RG05].

A explicação encontrada para este fenômeno neste trabalho é que, possivelmente, os classificadores decoraram alguns padrões de treinamento, manifestando o que é conhecido em Aprendizagem de Máquina como *overfitting*.

Em posse dos resultados do BMGGAVS, é necessário compará-los aos demais resultados, obtidos por outros métodos. Contudo, primeiramente, é preciso deixar claro quais serão os resultados do BMGGAVS usados para comparação. Os escolhidos foram os melhores dentre os oitenta e oito de cada base. Na Tabela 5.4 os valores são mostrados.

Tabela 5.4 Melhores resultados do BMGGAVS

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
BMGGAVS	0,0077	0,1365	0,1652	0,2173	0,1683	0,1417	0,1970	0,0299	0,1498

A primeira comparação será feita com três métodos de comitê clássicos: AdaBoost, *Bagging* e *Mixture of Experts*.

Os três métodos foram executados em MATLAB. Para a execução dos mesmos os dados foram separados da mesma forma em que o foram para o método proposto. O número de classificadores usados em todos os métodos clássicos foi de quinhentos e doze. Este número foi escolhido ao acaso.

Além do número de classificadores, para o MoE, foram definidos como taxa de aprendizado o valor de 0,1 (10%), o número de iterações como dez e os classificadores foram configurados como competitivos.

Os métodos clássicos foram executados por cem vezes, sendo que os dados foram separados em cada iteração, i.e., em cada iteração os subconjuntos continham padrões diferentes.

A Tabela 5.5 e a Figura 5.3 mostram os resultados obtidos e comparados ao BMGGAVS.

Tabela 5.5 Comparação de médias de erro de teste com métodos clássicos

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
AdaBoost	0,0410	0,1471	0,2429	0,5147	0,1795	0,2007	0,3649	0,6013	0,5960
	(0,0191)	*(0,0327)*	*(0,0344)*	*(0,0523)*	*(0,0276)*	*(0,0567)*	*(0,0320)*	*(0,0159)*	*(0,0352)*
Bagging	0,0307	**0,1290**	0,2394	**0,2134**	0,1804	0,1780	0,2975	0,0546	**0,0319**
	(0,0151)	*(0,0292)*	*(0,0341)*	*(0,0660)*	*(0,0293)*	*(0,0579)*	*(0,0431)*	*(0,0186)*	*(0,0175)*
MoE	0,0420	0,1566	0,3478	0,2188	0,2133	0,1902	0,2545	**0,0080**	0,1248
	(0,0195)	*(0,0341)*	*(≈ 0)*	*(0)*	*(0,0306)*	*(0,0478)*	*(≈ 0)*	*(0,0031)*	*(0,0104)*
BMGGAVS	**0.0077**	0.1365	**0.1652**	0.2173	**0,1683**	**0,1417**	**0,1970**	0,0299	0,1498
	(0,0046)	*(0,0176)*	*(0,0425)*	*(0,0834)*	*(0,0214)*	*(0,0227)*	*(0,0078)*	*(0,0105)*	*(0,0275)*

É possível perceber que o BMGGAVS, em comparação aos métodos clássicos, consegue ser o melhor, ou o segundo melhor, em todas as bases de dados, exceto na *Vowel*. O método proposto supera todos os demais em cinco das nove bases. Na Figura 5.3 fica evidente que o BMGGAVS consegue uma taxa de erro bem abaixo dos demais métodos nas bases *Diabetes*, *Heartc* e *Horse*.

Além disso, nas bases *Card* e *Glass*, estatisticamente, o BMGGAVS tem resultado

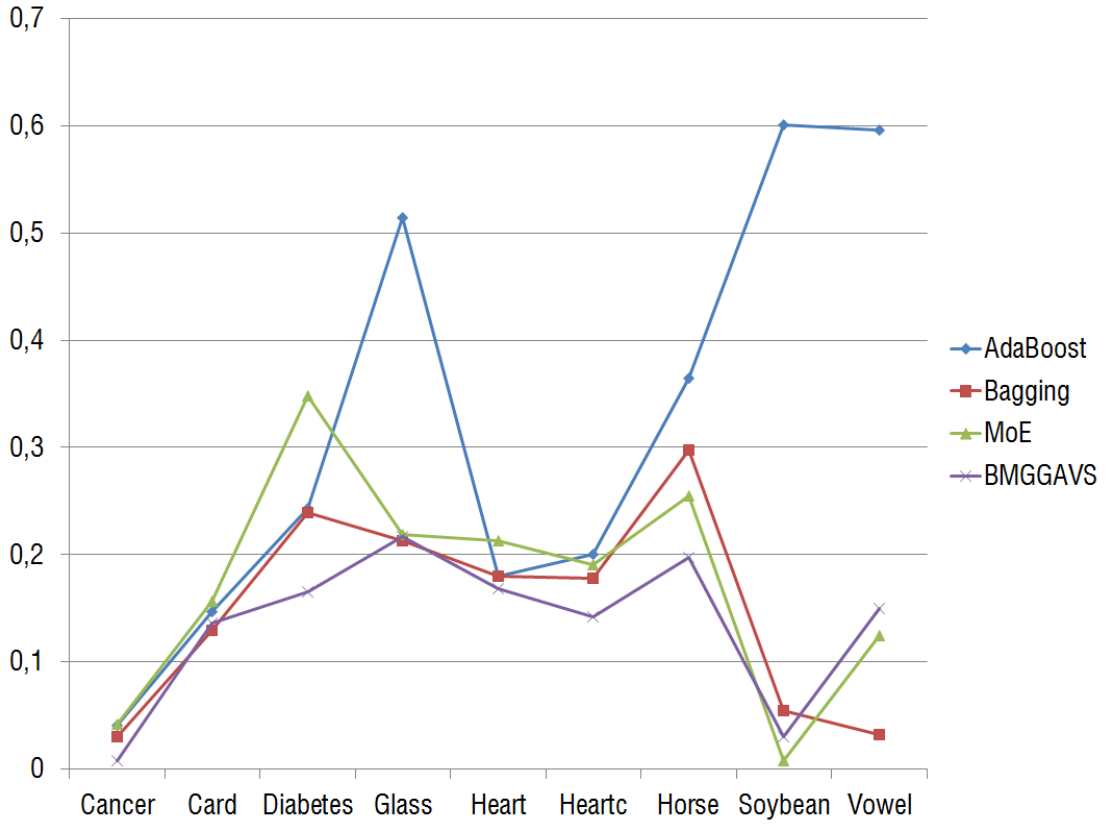

Figura 5.3 Comparação de médias de erro de teste com métodos clássicos

igual ao melhor dos métodos clássicos. Desta forma o mesmo se torna o melhor método em sete de nove bases.

Nas duas bases restantes (*Soybean* e *Vowel*) o BMGGAVS é o segundo melhor em um deles. Portanto, o método se destaca em oito das nove bases em relação aos métodos clássicos.

Assim sendo, é possível concluir que o método proposto consegue superar os métodos clássicos. Contudo, ainda é necessário verificar como o mesmo se comporta em relação a outros métodos presentes na literatura.

Nas Tabelas 5.6 e 5.7 e Figuras 5.4 e 5.5, o BMGGAVS é comparado aos métodos relacionados e similares, respectivamente.

Em comparação aos métodos relacionados, o BMGGAVS consegue ser o melhor em sete das nove bases de dados. É o pior somente na base *Card*. Contudo, nesta última

Tabela 5.6 Comparação de médias de erro de teste com métodos relacionados

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
DECORATE	0,3179	**0,1261**	0,2448	0,2766	0,1815	0,2249	0,2442	0,0542	–
MultiBoosting	0,3150	0,1360	0,2440	0,2490	0,1970	–	**0,1690**	0,0700	–
WAVE	0,0376	0,1374	0,2199	0,2430	0,1729	–	0,2911	–	0,2922
BMGGAVS	**0,0077**	0,1365	**0,1652**	**0,2173**	**0,1683**	**0,1417**	0,1970	**0,0299**	**0,1498**

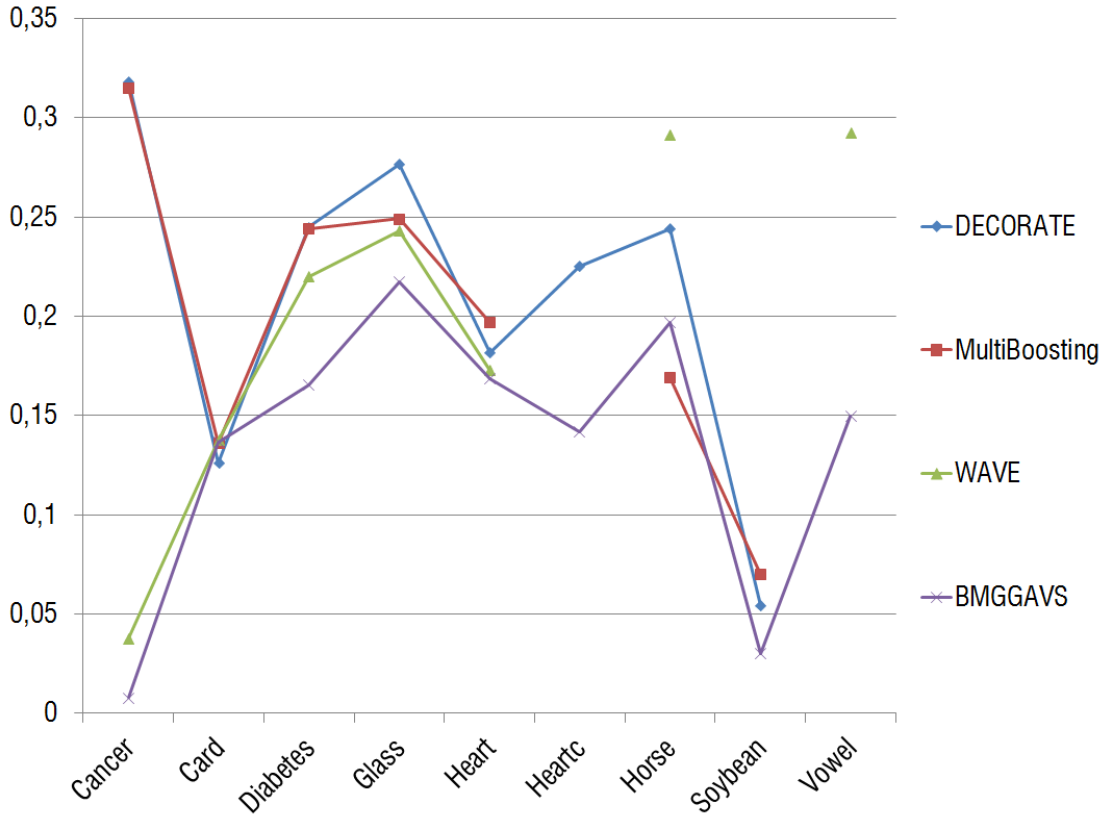

Figura 5.4 Comparação de médias de erro de teste com métodos relacionados

base, seu resultado chega a ser muito próximo ao do segundo melhor método. Como a diferença está somente na quarta casa decimal é possível dizer que há equivalência entre os resultados. Isto eleva o BMGGAVS ao posto de segundo melhor método na base *Card*.

Na base *Horse*, mesmo não sendo o melhor, o BMGGAVS, junto com o melhor método, conseguiram erros pelo menos 4% menores que os demais. Entre si a diferença é de menos de 3%. Em *Cancer* a diferença do BMGGAVS ao segundo melhor também teve uma diferença de cerca de 3%, mas em relação aos demais métodos o BMGGAVS

Tabela 5.7 Comparação de médias de erro de teste com métodos similares

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
CSEA	0,0199	0,1579	0,2378	0,3675	**0,1609**	**0,1075**	0,3108	0,2124	0,4098
CSJADE	0,0301	**0,1232**	0,2265	0,3400	0,1721	0,1736	0,3376	–	–
	(0,0116)	*(0,0279)*	*(0,0196)*	*(0,0540)*	*(0,0221)*	*(0,0343)*	*(0,0302)*		
SFDEGL	0,0335	–	0,2309	–	0,1652	0,1670	–	–	–
	(0,0100)		*(0,0197)*		*(0,0241)*	*(0,0242)*			
BMGGAVS	**0,0077**	0,1365	**0,1652**	**0,2173**	0,1683	0,1417	**0,1970**	**0,0299**	**0,1498**
	(0,0046)	*(0,0176)*	*(0,0425)*	*(0,0834)*	*(0,0214)*	*(0,0227)*	*(0,0078)*	*(0,0105)*	*(0,0275)*

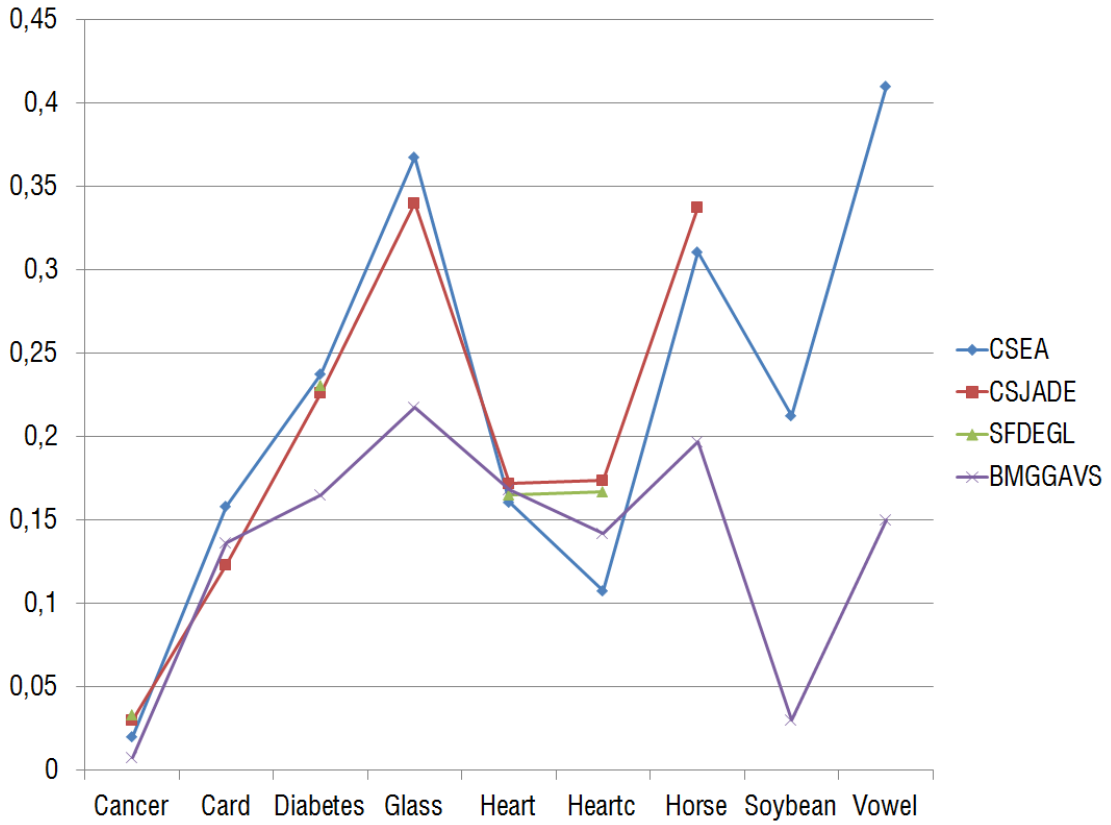

Figura 5.5 Comparação de médias de erro de teste com métodos similares

conseguiu um resultado mais de quarenta vezes menor.

Além da base *Cancer*, através da Figura 5.4 é possível ver que nas bases *Diabetes*, *Heartc* e *Vowel*, o BMGGAVS consegue taxas de erro significativamente menores. Nas demais bases, entretanto, as diferenças não foram tão grandes.

Em comparação aos métodos similares, da Tabela 5.7 é possível perceber que em todas as bases de dados o BMGGAVS supera ao menos um método. Ainda, na maioria das comparações, i.e., em seis de nove, o BMGGAVS consegue o menor erro. Estatisticamente apenas os resultados do CSJADE e SFDEGL nas bases *Card* e *Heart*, respectivamente, são iguais. Isto faz do BMGGAVS o melhor método em *Card* e o segundo melhor em *Heart*. Comparando com o CSEA, apenas nas bases *Heart* e *Heartc* o BMGGAVS não conseguiu melhores resultados. Contudo, em *Soybean* e *Vowel*, como é possível perceber na Figura 5.5, os resultados alcançados pelo BMGGAVS são bastante superiores aos do CSEA.

Portanto, é possível concluir que o BMGGAVS é um bom método, ao passo em que na maioria das bases ele consegue o menor ou o segundo menor erro. Destaca-se ainda pelo fato de várias vezes sua taxa de erro ficou bem abaixo dos demais métodos em comparação.

Finalizada as comparações entre as médias de erro, é interessante verificar também a média de tempo. Na Tabela 5.8 é possível ver as médias de tempo em minutos do BMGGAVS, dos métodos clássicos, como também de dois métodos similares. Esta comparação também é mostrada na Figura 5.6, entretanto os métodos CSJADE e SFDEGL foram retirados pelo fato de enviesarem bastante o gráfico, de modo que não seria possível ver as diferenças entre o BMGGAVS e os demais métodos. A comparação de tempo com outros métodos não foi feita devido ao fato de que os demais autores não forneceram tal informação.

Como esperado, os métodos clássicos conseguiram os menores custos de tempo, devido à sua simplicidade. Um fato notável é o BMGGAVS ter conseguido superar inclusive os métodos clássicos em duas bases (*Glass* e *Soybean*). Ainda, foi o segundo melhor em duas outras bases (*Heartc* e *Vowel*).

Em comparação aos métodos similares, o BMGGAVS foi sempre o melhor. No mínimo, foi sete vezes mais rápido que o CSJADE. Quanto ao SFDEGL, somente na base *Heart* houve uma proximidade nos resultados; nas demais bases, novamente o BMGGAVS consegue resultados de, no mínimo, sete vezes menores.

Além disso, é possível observar melhor pela Figura 5.6 que, exceto nas bases *Card* e

Tabela 5.8 Comparação de médias de tempo em minutos

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
AdaBoost	**0,2284**	**0,2705**	**0,2349**	0,2141	**0,2632**	**0,2379**	**0,2557**	0,3500	**0,2825**
	(0,0071)	*(0,0031)*	*(0,0062)*	*(0,0101)*	*(0,0012)*	*(≈ 0)*	*(0,0022)*	*(0,0039)*	*(0,0029)*
Bagging	0,2411	0,5839	0,5770	0,2785	0,7338	0,3850	0,6048	0.5929	0,6377
	(0,0102)	*(0,0235)*	*(0,0595)*	*(0,0347)*	*(0,0383)*	*(0,0296)*	*(0,0258)*	*(0,0735)*	*(0,0110)*
MoE	0,5970	0,9266	0,6227	0,4261	1,3592	0,4197	0,6281	5,5258	2,9167
	(0,0331)	*(0,1009)*	*(0,0256)*	*(0,0179)*	*(0,2772)*	*(0,1476)*	*(0,1399)*	*(0,6817)*	*(0,0072)*
CSJADE	23,79	28,21	33,21	32,69	28,08	30,48	37,48	–	–
	(2,59)	*(3,16)*	*(2,92)*	*(3,62)*	*(2,23)*	*(4,06)*	*(2,91)*		
SFDEGL	5,0589	–	5,0598	–	7,7036	3,8941	–	–	–
BMGGAVS	0,6075	3,8357	0,7103	**0,1809**	6,5947	0,2730	1,0549	**0,3450**	0,5116
	(0,0525)	*(0,1782)*	*(0,0718)*	*(0,0181)*	*(0,3849)*	*(0,0215)*	*(0,0256)*	*(0,0233)*	*(0,0839)*

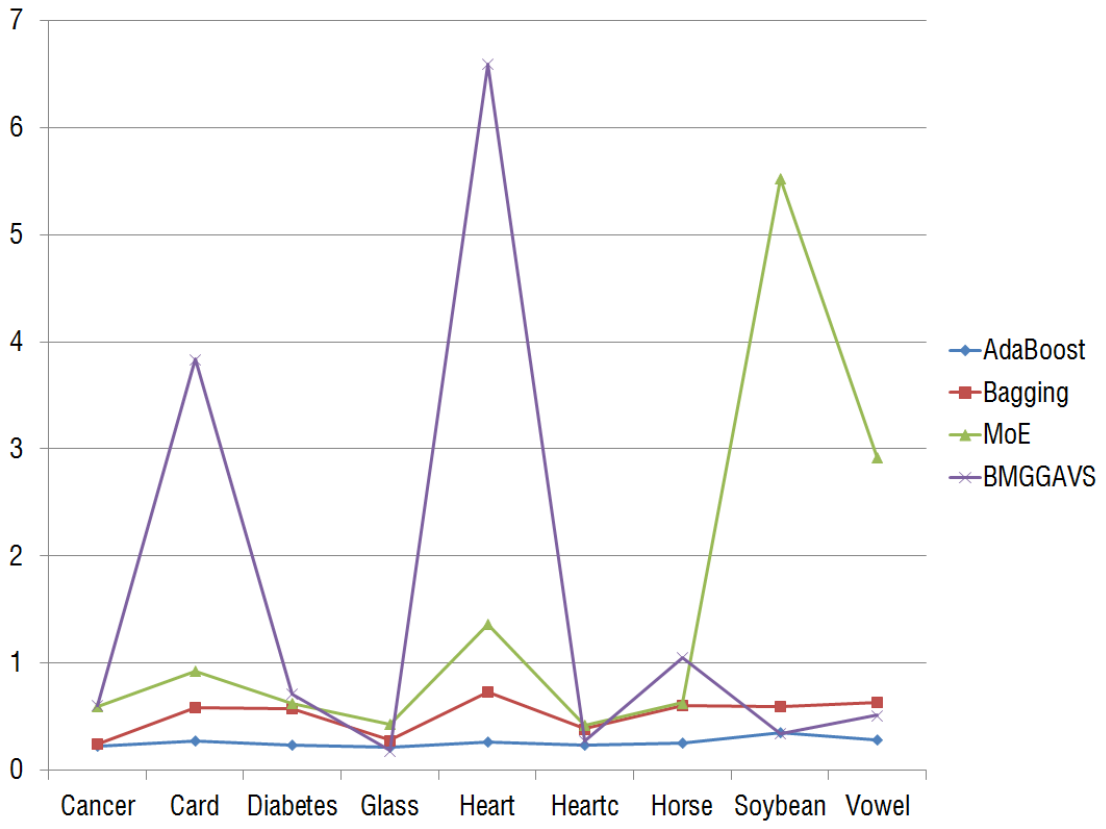

Figura 5.6 Comparação de médias de tempo em minutos

Heart[1], o tempo gasto pelo BMGGAVS normalmente é próximo ao dos métodos clássicos.

[1]Nestas bases o maior consumo de tempo deve-se ao fato de as bases possuírem mais instâncias,

Portanto, é possível concluir que, além de prover bons resultados, com baixos erros, o BMGGAVS não sofre com o consumo de tempo, evidenciando mais uma vantagem.

5.3 BMG2GA

5.3.1 Parâmetros e critérios

No BMGGAVS o relacionamento entre dois padrões é determinado manualmente. Em contrapartida, o BMG2GA constrói automaticamente o grafo que mostra todas as instâncias e seus respectivos vizinhos. Diferentemente de se testar onze limites gerais, o limite agora é calculado para cada padrão individualmente.

Os detalhes do cálculo do limite podem ser vistos na Seção 4.2.1, onde são apresentadas as equações 4.2, 4.3, 4.4 e 4.5.

A próxima modificação do BMG2GA em relação ao BMGGAVS são os novos tipos de classificadores. Enquanto a primeira versão do método havia apenas dois tipos de classificadores (MLP10 ou MLP15 e Perceptron), a segunda versão chega a ter sete tipos de classificadores: MLP5, MLP10, MLP15, Perceptron, ELM5, ELM10 e ELM15. Assim como no BMGGAVS, os valores dos parâmetros das MLPs e do Perceptron são os valores padrões, fornecidos pelo MATLAB – como descrito no último parágrafo da Seção 5.2.1. Quanto às ELMs, todas receberam a função sigmoide como função de ativação.

O sistema de votação presente no BMGGAVS foi substituído por um Algoritmo Genético, e chamado *Ensemble GA*. O número de indivíduos é cem, e a escolha dos pais para a recombinação (cuja taxa é de 100%) é através do método da roleta. Após a recombinação a prole produzida passa pelo processo de mutação a uma taxa de 0,1%. Por fim, a substituição da população é geracional, i.e., a população de prole substitui por completo a população anterior.

A função de aptidão é a exposta na equação (4.6). Essa função sofre alteração quando ELMs são usadas no comitê de classificadores, pois não há o uso do subconjunto de validação. Portanto, com ELMs presentes na seleção de classificadores, a função de

atributos ou classes.

aptidão passa a ser:

$$f = 1 - \frac{\Sigma_1^k e_{t_k}}{n_t} \tag{5.2}$$

Na equação (5.2) e_{t_k} é o número de padrões erroneamente classificados no teste do cluster k e n_t é o número de padrões de teste. É possível perceber que a mudança sofrida em relação à equação (4.6) é apenas a retirada dos erros de treinamento e validação. Agora se leva em consideração apenas o erro de teste.

Este segundo algoritmo genético retorna a melhor combinação de classificadores, um para cada *cluster*. Encontrada a combinação, os classificadores serão expostos ao conjunto de teste final.

5.3.2 Resultados e discussões

O experimento inclui novas bases de dados, como também o aumento do número de repetições, de cinquenta para cem. Outra diferença entre os experimentos, como pode ser visto na Figura 5.7, reside no fato de que a cada repetição o conjunto de dados é novamente dividido em conjuntos e subconjuntos. No BMGGAVS essa divisão era realizada somente uma vez. Assim a cada nova iteração, os mesmos conjuntos e subconjuntos eram usados. Desta vez a cada nova iteração os conjuntos e subconjuntos são diferentes.

Os demais detalhes seguem o que foi estabelecido na experimentação do BMGGAVS. Por exemplo, o uso do teste t de *student* com $\alpha = 5\%$ quando os dados são normais e o uso do teste *rank-sum* de Wilcoxon, quando os dados não foram normais. Além disso, é importante lembrar que os experimentos foram realizados em computadores diferentes, com configurações diferentes. As variações do BMG2GA serão detalhadas posteriormente.

Nas Tabelas 5.9, 5.10 e 5.11 e Figuras 5.8, 5.9 e 5.10 são mostrados os primeiros resultados, onde o BMG2GA, em sua variação 1, é comparado aos métodos clássicos, relacionados e similares, respectivamente. É possível notar que não se fazem presentes as outras três bases de dados referidas na Seção 5.1. As novas bases foram usadas apenas

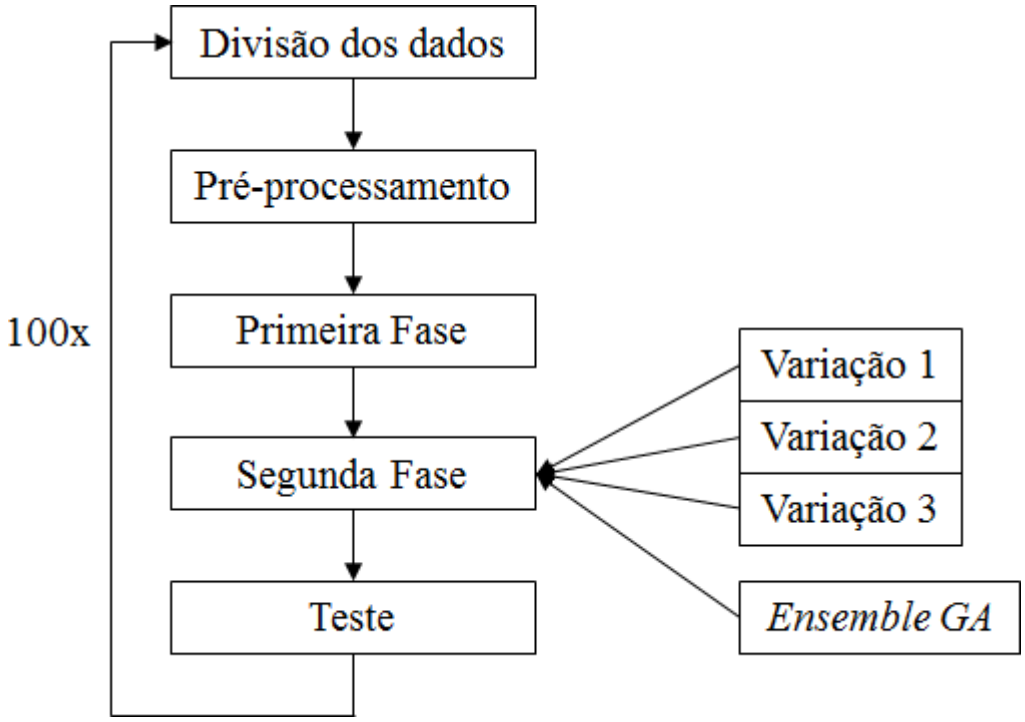

Figura 5.7 Fluxograma do experimento realizado com o BMG2GA

nos experimentos das variações 2 e 3 do BMG2GA, devido ao fato de essas variações terem conseguido melhores resultados.

Tabela 5.9 Comparação de médias de erro de teste do BMG2GA com métodos clássicos

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
AdaBoost	0,0410	0,1471	0,2429	0,5147	**0,1795**	0,2007	0,3649	0,6013	0,5960
	(0,0191)	*(0,0327)*	*(0,0344)*	*(0,0523)*	*(0,0276)*	*(0,0567)*	*(0,0320)*	*(0,0159)*	*(0,0352)*
Bagging	**0,0307**	**0,1290**	**0,2394**	**0,2134**	0,1804	**0,1780**	0,2975	0,0546	**0,0319**
	(0,0151)	*(0,0292)*	*(0,0341)*	*(0,0660)*	*(0,0293)*	*(0,0579)*	*(0,0431)*	*(0,0186)*	*(0,0175)*
MoE	0,0420	0,1566	0,3478	0,2188	0,2133	0,1902	**0,2545**	**0,0080**	0,1248
	(0,0195)	*(0,0341)*	*(≈ 0)*	*(0)*	*(0,0306)*	*(0,0478)*	*(≈ 0)*	*(0,0031)*	*(0,0104)*
BMG2GA	0,0328	0,2906	0,2649	0,3496	0,2770	0,2839	0,3907	0,1377	0,2686
	(0,0190)	*(0,0447)*	*(0,1241)*	*(0,1025)*	*(0,0444)*	*(0,0813)*	*(0,0709)*	*(0,0357)*	*(0,0405)*

O BMG2GA é um método que não utiliza ajuste manual de parâmetros para cada nova situação. Como consequência óbvia, há uma taxa de erro maior. Entretanto, a taxa de erro alcançada pelo BMG2GA foi bem superior ao que era esperado, ao ponto de conseguir piores resultados que os métodos clássicos em quase todas as bases de dados testadas. Somente na base *Cancer* é que o BMG2GA conseguiu um resultado

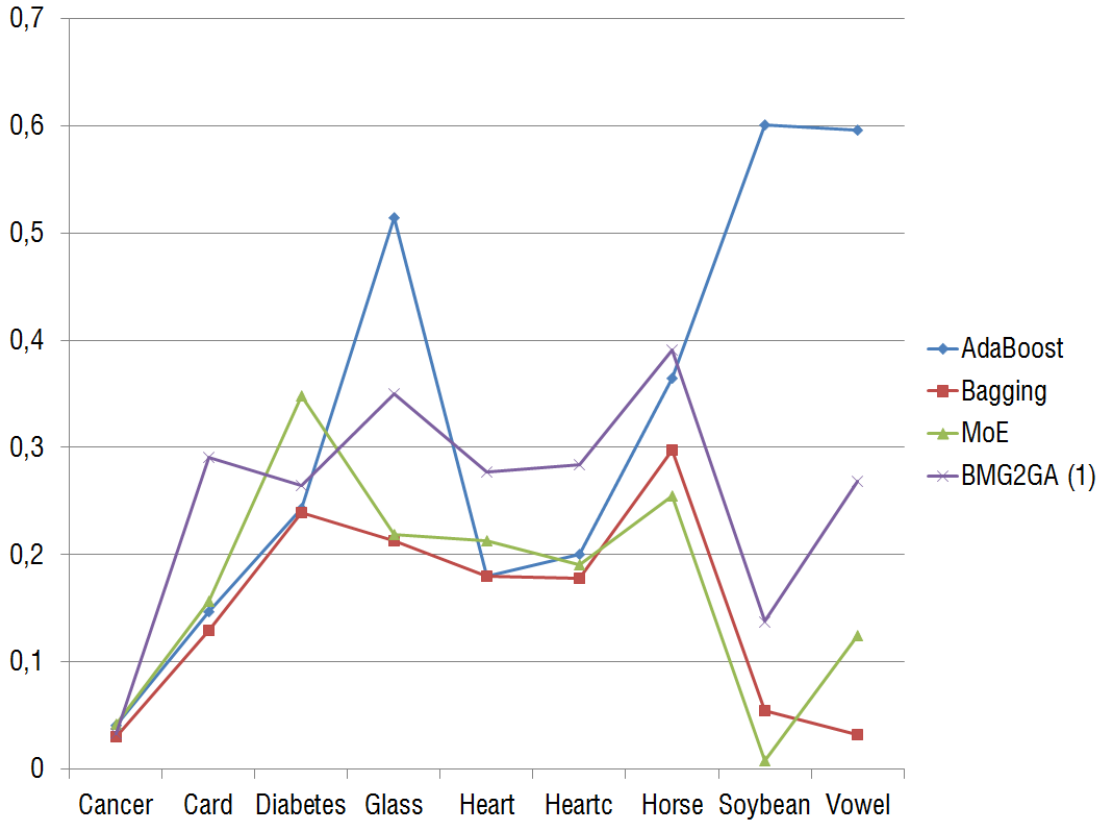

Figura 5.8 Comparação de médias de erro de teste do BMG2GA, variação 1, com métodos clássicos

estatisticamente igual ao melhor método. O melhor resultado, após esse, ocorreu somente na base *Diabetes*, onde estatisticamente o BMG2GA fica em segundo lugar, empatado com o AdaBoost.

Ainda que os resultados, quando comparados aos métodos clássicos, tenham sido ruins, o BMG2GA não deve ser considerado péssimo. O motivo é que o método conseguiu superar ao menos um de seus concorrentes em quatro das nove bases, e em nenhum caso chegou a ter 50% ou mais de erro. Ainda, como é possível observar na Figura 5.8, por mais que sua taxa de erro tenha sido, normalmente, maior que a dos demais métodos, o BMG2GA não se distanciou muito.

Na comparação com métodos relacionados, é possível ver com mais clareza que o BMG2GA, apesar de ter tido um fraco desempenho, não foi de todo ruim. Chegou a ser o melhor em duas bases (*Cancer* e *Vowel*). Ainda, em uma quarta base de dados (*Diabetes*), mesmo não tendo sido o melhor, obteve um resultado próximo ao obtido pelos

Tabela 5.10 Comparação de médias de erro de teste do BMG2GA com métodos relacionados

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
DECORATE	0,3179	**0,1261**	0,2448	0,2766	0,1815	**0,2249**	0,2442	**0,0542**	–
MultiBoosting	0,3150	0,1360	0,2440	0,2490	0,1970	–	**0,1690**	0,0700	–
WAVE	0,0376	0,1374	**0,2199**	**0,2430**	**0,1729**	–	0,2911	–	0,2922
BMG2GA	**0,0328**	0,2906	0,2649	0,3496	0,2770	0,2839	0,3907	0,1377	**0,2686**

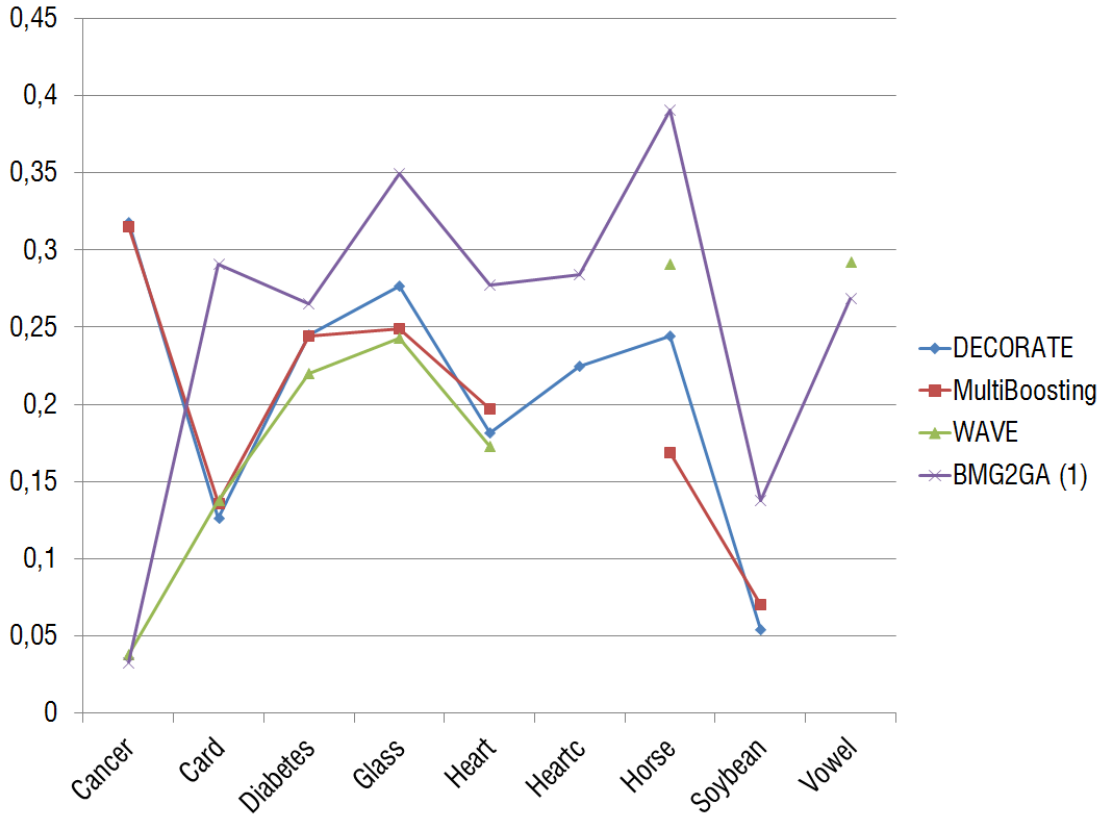

Figura 5.9 Comparação de médias de erro de teste do BMG2GA, variação 1, com métodos relacionados

seus concorrentes. Nas outras cinco bases restantes seu desempenho, como pode ser visto na Figura 5.9, infelizmente, ficou bem aquém em relação aos dos outros métodos.

O desempenho do BMG2GA melhora quando comparado aos métodos similares. A comparação com os métodos similares é importante pelo fato de que tanto a justificativa quanto a proposta para solução são bastante parecidas. Já a comparação com métodos clássicos tem sua importância devido ao fato destes serem normalmente mais simples e rápidos, ao mesmo tempo em que retornam bons resultados. Portanto, um bom de-

Tabela 5.11 Comparação de médias de erro de teste do BMG2GA com métodos similares

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
CSEA	**0,0199**	0,1579	0,2378	0,3675	**0,1609**	**0,1075**	**0,3108**	0,2124	0,4098
CSJADE	0,0301	**0,1232**	**0,2265**	**0,3400**	0,1721	0,1736	0,3376	–	–
	(0,0116)	*(0,0279)*	*(0,0196)*	*(0,0540)*	*(0,0221)*	*(0,0343)*	*(0,0302)*		
SFDEGL	0,0335	–	0,2309	–	0,1652	0,1670	–	–	–
	(0,0100)		*(0,0197)*		*(0,0241)*	*(0,0242)*			
BMG2GA	0,0328	0,2906	0,2649	0,3496	0,2770	0,2839	0,3907	**0,1377**	**0,2686**
	(0,0190)	*(0,0447)*	*(0,1241)*	*(0,1025)*	*(0,0444)*	*(0,0813)*	*(0,0709)*	*(0,0357)*	*(0,0405)*

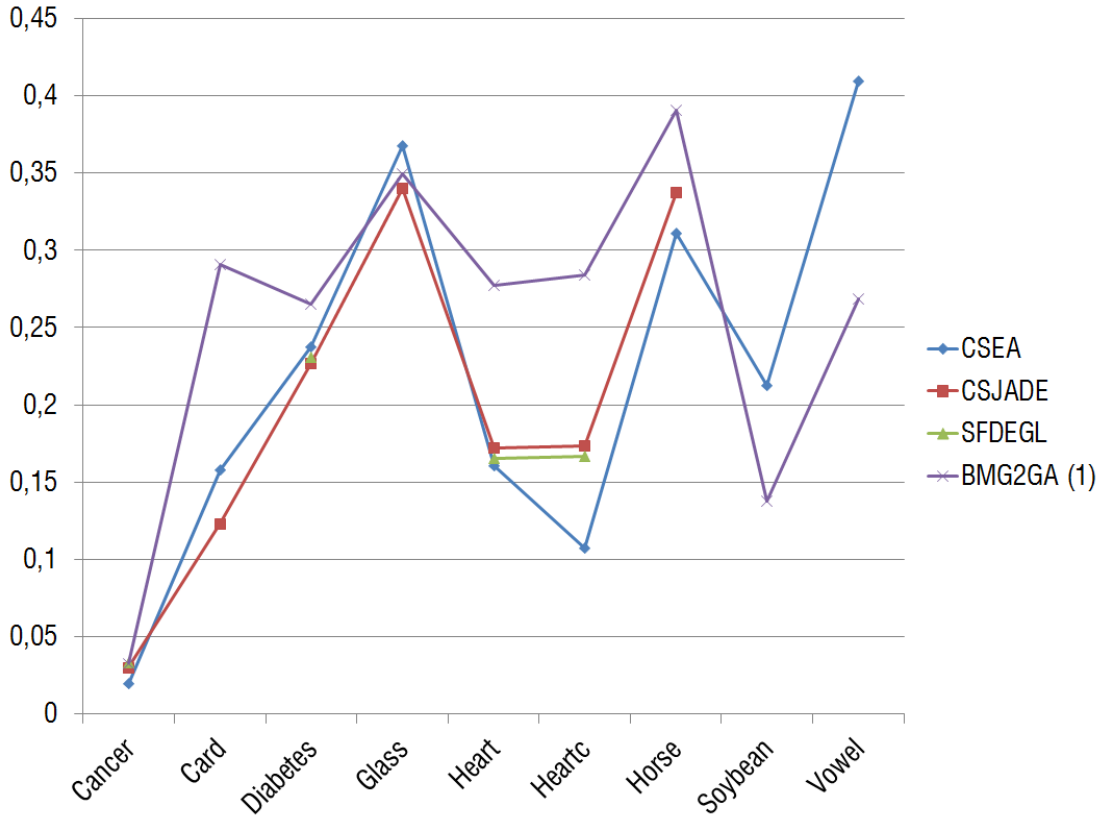

Figura 5.10 Comparação de médias de erro de teste do BMG2GA, variação 1, com métodos similares

sempenho em relação aos métodos similares, é um relativo sucesso, pois, dentre outros, o objetivo deste trabalho é promover uma alternativa mais viável em relação aos seus competidores diretos.

Os resultados mostram que em duas bases (*Soybean* e *Vowel*) o BMG2GA consegue

ser o melhor. Através da Figura 5.10 é possível perceber melhor o quão mais baixo é sua taxa de erro, em relação aos demais. Na base *Glass*, o BMG2GA, estatisticamente, é igual ao CSJADE, sendo, portanto, o melhor método nesta base também. Ainda, na base *Cancer*, estatisticamente, o BMG2GA volta a ser equivalente ao CSJADE, logo, vindo a ser também o segundo melhor método nesta base. Por fim, na base *Diabetes*, mesmo tendo sido o pior, seu resultado não foi distante dos demais.

Em conclusão, ainda que *a priori* os resultados do BMG2GA tenham sido ruins, é possível que se encontre resultados melhores caso o método venha a ser mais bem trabalhado.

Como citado anteriormente, outros experimentos foram realizados com variações do BMG2GA. Ao todo foram implementadas três variações. A primeira (cujos resultados foram mostrados nas Tabelas 5.9, 5.10 e 5.11 e Figuras 5.8, 5.9 e 5.10) tem por característica a presença de todos os tipos de classificadores. É importante lembrar que, devido à presença de ELMs, a função de aptidão no *Ensemble GA* é a mostrada na equação (5.2). Logo, o uso do erro do subconjunto de teste, somente, pode ter prejudicado na busca da melhor combinação de classificadores.

A segunda variação exclui somente as ELMs; todos os demais tipos de classificadores estão presentes. Desta forma, a escolha da combinação de classificadores passa a ser feita através da função de aptidão, mostrada na equação (4.6). Por fim, a terceira variação exclui os classificadores do tipo MLP15. Não foi experimentada outra variação, com menos tipos de classificadores, porque os resultados das variações 2 e 3 foram muito próximos, evidenciando que a partir de então não deve haver um ganho tão efetivo de desempenho.

As duas últimas variações obtiveram resultados melhores em relação à primeira variação. Portanto, somente com as mesmas é que as novas bases de dados foram usadas nos experimentos. As Tabelas 5.12, 5.13 e 5.14 e as Figuras 5.11, 5.12 e 5.13 apresentam os resultados obtidos com as duas últimas variações. Os números entre parênteses após o BMG2GA indicam qual variação, respectivamente, está sendo listada.

A comparação com métodos clássicos, na Tabela 5.12, é semelhante à da primeira configuração do BMG2GA. É perceptível um fraco desempenho. Contudo, o BMG2GA, em sua terceira configuração, consegue ser o melhor na base *Cancer* e o segundo melhor

Tabela 5.12 Comparação de médias de erro de teste das variações 2 e 3 do BMG2GA com métodos clássicos

	AdaBoost	Bagging	MoE	BMG2GA (2)	BMG2GA (3)
Cancer	0,0410	0,0307	0,0420	0,0330	**0,0295**
	(0,0191)	*(0,0151)*	*(0,0195)*	*(0,0176)*	*(0,0136)*
Card	0,1471	**0,1290**	0,1566	0,2105	0,1944
	(0,0327)	*(0,0292)*	*(0,0341)*	*(0,0446)*	*(0,0343)*
Diabetes	0,2429	**0,2394**	0,3478	0,2565	0,2588
	(0,0344)	*(0,0341)*	*(≈ 0)*	*(0,1141)*	*(0,1199)*
Glass	0,5147	**0,2134**	0,2188	0,3326	0,3330
	(0,0523)	*(0,0660)*	*(0)*	*(0,1058)*	*(0,1194)*
Heart	**0,1795**	0,1804	0,2133	0,2152	0,2115
	(0,0276)	*(0,0293)*	*(0,0306)*	*(0,0362)*	*(0,0348)*
Heartc	0,2007	**0,1780**	0,1902	0,2173	0,2204
	(0,0567)	*(0,0579)*	*(0,0478)*	*(0,0611)*	*(0,0543)*
Horse	0,3649	0,2975	**0,2545**	0,3285	0,3067
	(0,0320)	*(0,0431)*	*(≈ 0)*	*(0,0661)*	*(0,0585)*
Segment	0,0907	**0,0202**	0,1429	0,0701	0,0697
	(0,0153)	*(0,0074)*	*(≈ 0)*	*(0,0119)*	*(0,0115)*
Soybean	0,6013	0,0546	**0,0080**	0,1064	0,0983
	(0,0159)	*(0,0186)*	*(0,0031)*	*(0,0262)*	*(0,0272)*
Thyroid	0,0090	**0,0036**	0,0494	0,0358	0,0366
	(0,0027)	*(0,0015)*	*(≈ 0)*	*(0,0079)*	*(0,0090)*
Vowel	0,5960	**0,0319**	0,1248	0,1442	0,1478
	(0,0352)	*(0,0175)*	*(0,0104)*	*(0,0298)*	*(0,0266)*
Yeast	0,5847	0,3873	**0,1381**	0,3418	0,3313
	(0,0169)	*(0,0317)*	*(≈ 0)*	*(0,1481)*	*(0,1623)*

nas bases *Segment* e *Yeast*. No restante das bases, como é perceptível na Figura 5.11, mesmo não conseguindo se destacar, o BMG2GA, normalmente, obtém resultados próximos aos dos demais. Somente em quatro das doze bases testadas é que ambas as configurações não conseguem superar nenhum dos métodos clássicos.

Na Tabela 5.13 e Figura 5.12, as duas últimas configurações do BMG2GA são comparadas aos métodos relacionados. Por três vezes (*Cancer*, *Heartc* e *Vowel*) as duas configurações conseguiram os dois melhores resultados. Nas demais bases o método proposto não obteve um bom desempenho em relação aos outros. Poucas vezes seus resultados ficaram próximos aos dos demais métodos.

Por fim, as variações feitas no BMG2GA são comparadas aos métodos similares, na

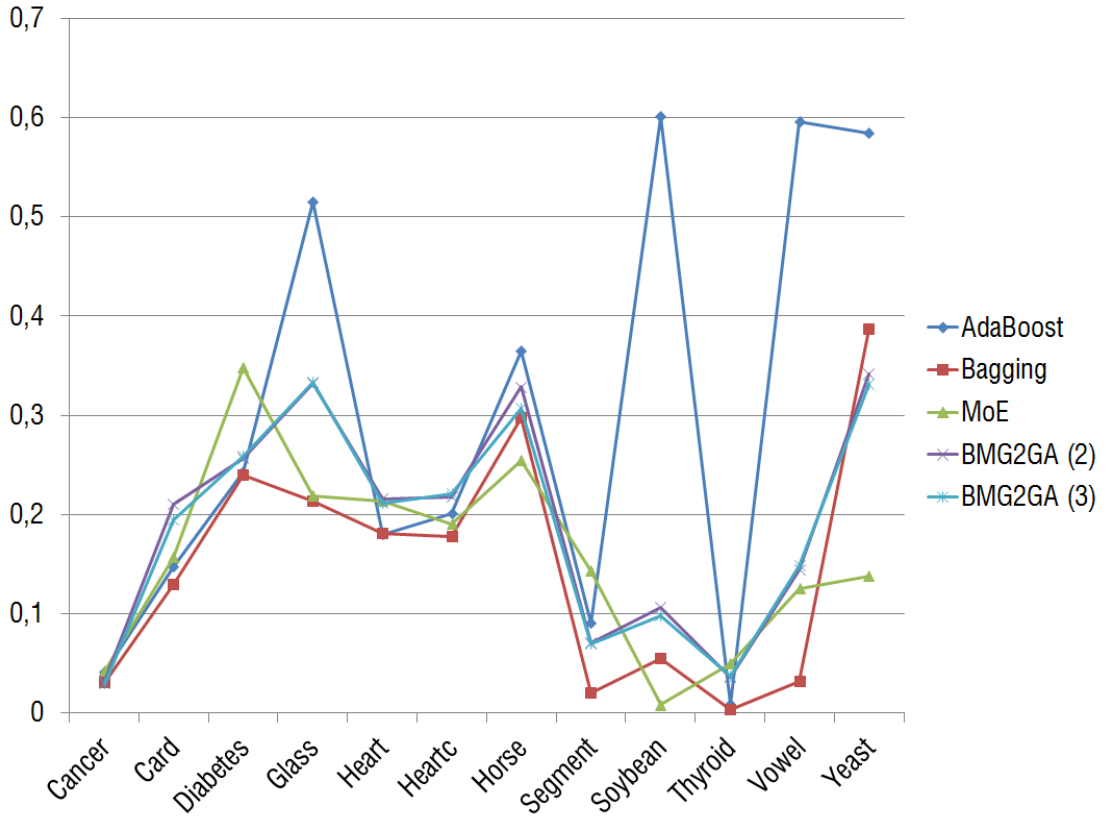

Figura 5.11 Comparação de médias de erro de teste das variações 2 e 3 do BMG2GA com métodos clássicos

Tabela 5.14 e Figura 5.13. Pela primeira vez, desde então, a terceira configuração do BMG2GA não conseguiu o melhor resultado na base *Cancer*, sendo superado pelo CSEA. De forma parecida às outras comparações, na base *Diabetes*, ainda que os resultados do BMG2GA tenham sido os piores, eles ainda ficaram bem próximos aos resultados de seus concorrentes.

Na base *Glass*, pela primeira vez, o BMG2GA conseguiu não só o melhor resultado, mas o segundo melhor também. Nas bases *Soybean*, *Vowel* e *Yeast* o mesmo volta a acontecer, i.e., os dois melhores resultados foram atingidos pelas variações do BMG2GA.

Além das bases já citadas, a terceira configuração do BMG2GA conseguiu se destacar como o melhor na base *Horse*. Nas demais bases, ainda que a segunda ou terceira configuração do BMG2GA tenha conseguido o segundo melhor resultado, isto não foi levado em conta pelo fato de ambos estarem concorrendo com apenas um terceiro método. Logo, se o outro método, neste caso, conseguiu o melhor resultado, obrigatoriamente o

Tabela 5.13 Comparação de médias de erro de teste das variações 2 e 3 do BMG2GA com métodos relacionados

	DECORATE	MultiBoosting	WAVE	BMG2GA (2)	BMG2GA (3)
Cancer	0,3179	0,3150	0,0376	0,0330	**0,0295**
Card	**0,1261**	0,1360	0,1374	0,2105	0,1944
Diabetes	0,2448	0,2440	**0,2199**	0,2565	0,2588
Glass	0,2766	0,2490	**0,2430**	0,3326	0,3330
Heart	0,1815	0,1970	**0,1729**	0,2152	0,2115
Heartc	0,2249	–	–	**0,2173**	0,2204
Horse	0,2442	**0,1690**	0,2911	0,3285	0,3067
Segment	–	–	–	0,0701	**0,0697**
Soybean	**0,0542**	0,0700	–	0,1064	0,0983
Thyroid	0,0140	**0,0030**	–	0,0358	0,0366
Vowel	–	–	0,2922	**0,1442**	0,1478
Yeast	–	–	–	0,3418	**0,3313**

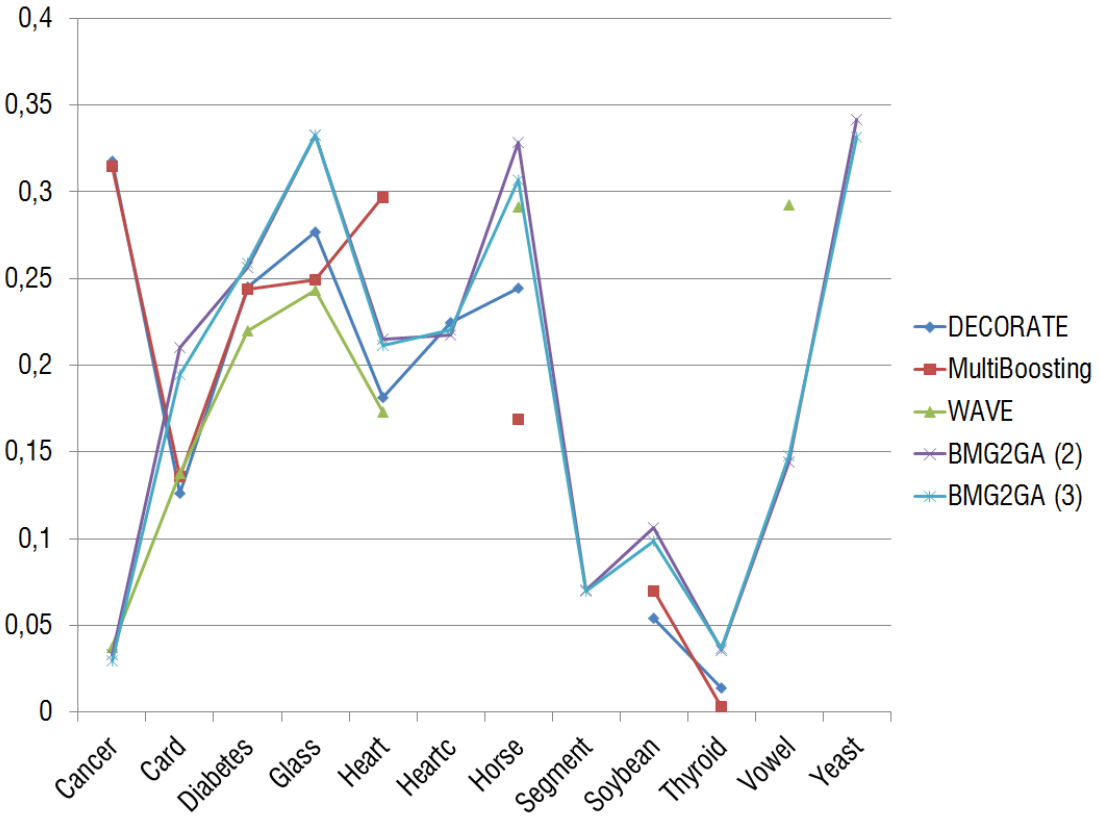

Figura 5.12 Comparação de médias de erro de teste das variações 2 e 3 do BMG2GA com métodos relacionados

Tabela 5.14 Comparação de médias de erro de teste das variações 2 e 3 do BMG2GA com métodos similares

	CSEA	CSJADE	SFDEGL	BMG2GA (2)	BMG2GA (3)
Cancer	**0,0199**	0,0301 *(0,0116)*	0,0335 *(0,0100)*	0,0330 *(0,0176)*	0,0295 *(0,0136)*
Card	0,1579	**0,1232** *(0,0292)*	–	0,2105 *(0,0446)*	0,1944 *(0,0343)*
Diabetes	0,2378	**0,2265** *(0,0196)*	0,2309 *(0,0197)*	0,2565 *(0,1141)*	0,2588 *(0,1199)*
Glass	0,3675	0,3400 *(0,0540)*	–	**0,3326** *(0,1058)*	0,3330 *(0,1194)*
Heart	**0,1609**	0,1721 *(0,0221)*	0,1652 *(0,0241)*	0,2152 *(0,0362)*	0,2115 *(0,0348)*
Heartc	**0,1075**	0,1736 *(0,0343)*	0,1670 *(0,0242)*	0,2173 *(0,0611)*	0,2204 *(0,0543)*
Horse	0,3108	0,3376 *(0,0302)*	–	0,3285 *(0,0661)*	**0,3067** *(0,0585)*
Segment	**0,0578**	–	–	0,0701 *(0,0119)*	0,0697 *(0,0115)*
Soybean	0,2124	–	–	0,1064 *(0,0262)*	**0,0983** *(0,0272)*
Thyroid	**0,0309**	–	–	0,0358 *(0,0079)*	0,0366 *(0,0090)*
Vowel	0,4098	–	–	**0,1442** *(0,0298)*	0,1478 *(0,0266)*
Yeast	0,4356	–	–	0,3418 *(0,1481)*	**0,3313** *(0,1623)*

BMG2GA será tido como segundo e terceiro.

Após as comparações com outros métodos, pode ser interessante a comparação do BMGGAVS com a sua evolução e variações. Por isso é fornecida a Tabela 5.15 e a Figura 5.14, onde os resultados obtidos nos experimentos com o método proposto são confrontados.

Como foi dito anteriormente, é possível notar que o BMG2GA, em sua primeira variação, conseguiu os piores resultados. Isso se deu pelo fato do mesmo considerar apenas o erro do subconjunto de teste ao buscar a melhor combinação de classificadores. Outro detalhe que pode ser visto com mais facilidade nesta tabela é o fato da terceira variação do BMG2GA se aproximar mais dos resultados obtidos pelo BMGGAVS. A

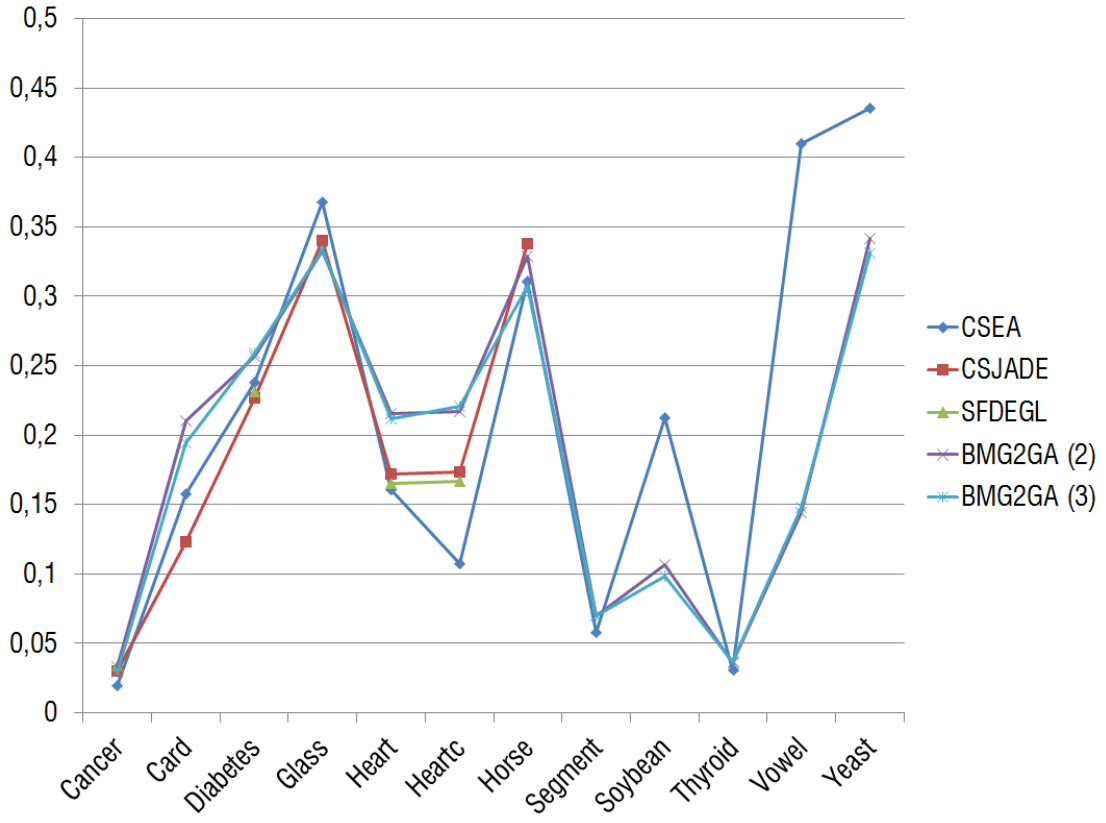

Figura 5.13 Comparação de médias de erro de teste das variações 2 e 3 do BMG2GA com métodos similares

única exceção é na base *Vowel*, onde a segunda e terceira configurações conseguiram superar o BMGGAVS.

Além disso, somente em três bases (*Diabetes*, *Glass* e *Heartc*) é que a segunda variação do BMG2GA superou a terceira, fazendo desta última, a configuração que mais se aproxima dos bons resultados obtidos pelo BMGGAVS. Apesar disso, na maioria dos casos, ambas as configurações chegam a ser estatisticamente iguais; este fato é mais facilmente percebido através da Figura 5.14. Entretanto, a terceira configuração tem a vantagem de ser mais simples (por envolver um tipo de classificador a menos) e mais rápida.

A questão do tempo pode ser vista na Tabela 5.16 e na Figura 5.15, onde as médias de tempo em minutos de todos os métodos usados em comparação neste trabalho são mostradas. Semelhantemente à Figura 5.6, na Figura 5.15 somente as médias do método proposto e métodos clássicos é que são mostradas. A diferença da Tabela 5.16 para a

Tabela 5.15 Comparação de médias de erro de teste entre BMGGAVS, BMG2GA e suas variações

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
BMGGAVS	**0.0077**	**0.1365**	**0.1652**	**0.2173**	**0.1683**	**0.1417**	**0.1970**	**0.0299**	0.1498
	(0,0046)	*(0,0176)*	*(0,0425)*	*(0,0834)*	*(0,0214)*	*(0,0227)*	*(0,0078)*	*(0,0105)*	*(0,0275)*
BMG2GA (1)	0,0328	0,2906	0,2649	0,3496	0,2770	0,2839	0,3907	0,1377	0,2686
	(0,0190)	*(0,0447)*	*(0,1241)*	*(0,1025)*	*(0,0444)*	*(0,0813)*	*(0,0709)*	*(0,0357)*	*(0,0405)*
BMG2GA (2)	0,0330	0,2105	0,2565	0,3326	0,2152	0,2173	0,3285	0,1064	**0,1442**
	(0,0176)	*(0,0446)*	*(0,1141)*	*(0,1058)*	*(0,0362)*	*(0,0611)*	*(0,0661)*	*(0,0262)*	*(0,0298)*
BMG2GA (3)	0,0295	0,1944	0,2588	0,3330	0,2115	0,2204	0,3067	0,0983	0,1478
	(0,0136)	*(0,0343)*	*(0,1199)*	*(0,1194)*	*(0,0348)*	*(0,0543)*	*(0,0585)*	*(0,0272)*	*(0,0266)*

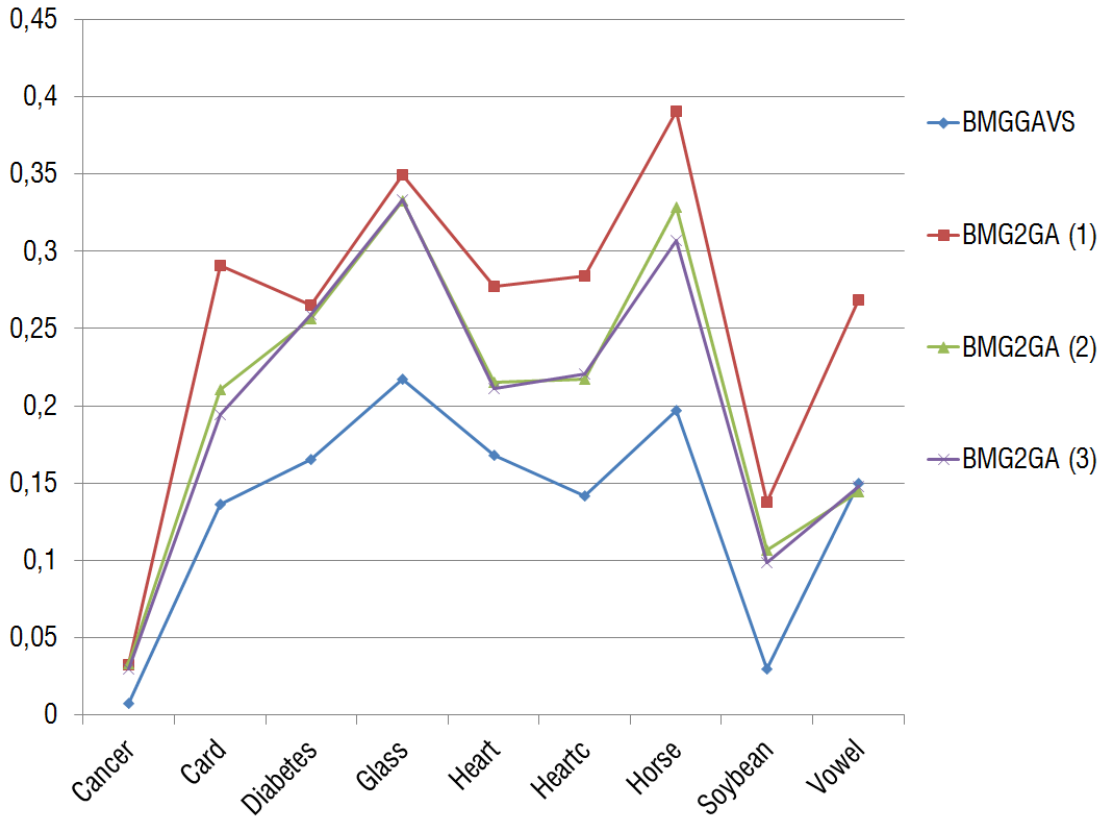

Figura 5.14 Comparação de médias de erro de teste entre BMGGAVS, BMG2GA e suas variações

Tabela 5.8 é o acréscimo das médias de tempo do BMG2GA. Logo, a discussão a seguir focará sobre esses novos resultados.

Normalmente o BMG2GA leva mais tempo que o BMGGAVS para executar por completo. O que contribui bastante para isso é o fato do sistema simples de voto ter

sido substituído por uma busca via Algoritmo Genético. Ainda assim, em duas situações (*Diabetes* e *Heart*), o BMG2GA conseguiu uma média de tempo menor. Em outras duas situações (*Cancer* e *Card*), as médias de tempo foram bem próximas.

Outro fator que pode ter influenciado no tempo de execução é a forma como o grafo foi construído. No BMG2GA a vizinhança de cada padrão é determinada de acordo com as suas propriedades. Desta forma, várias instâncias que viessem a se ligar no BMGGAVS, num determinado limite, podem não vir a se ligar no BMG2GA em hipótese alguma. Além disso, a cada nova repetição, o BMG2GA lida com padrões diferentes, pois os conjuntos e subconjuntos são produzidos aleatoriamente. Logo, o tempo de execução do BMG2GA depende dos dados. Ou seja, o BMG2GA pode vir a executar mais rápido quando os conjuntos estiverem mais *fáceis* e demorar mais se os conjuntos estiverem mais *difíceis*.

Tabela 5.16 Comparação de médias de tempo em minutos entre todos os métodos

	Cancer	*Card*	*Diabetes*	*Glass*	*Heart*	*Heartc*	*Horse*	*Soybean*	*Vowel*
AdaBoost	**0,2284**	**0,2705**	**0,2349**	0,2141	**0,2632**	**0,2379**	**0,2557**	0,3500	**0,2825**
	(0,0071)	*(0,0031)*	*(0,0062)*	*(0,0101)*	*(0,0012)*	*(≈ 0)*	*(0,0022)*	*(0,0039)*	*(0,0029)*
Bagging	0,2411	0,5839	0,5770	0,2785	0,7338	0,3850	0,6048	0.5929	0,6377
	(0,0102)	*(0,0235)*	*(0,0595)*	*(0,0347)*	*(0,0383)*	*(0,0296)*	*(0,0258)*	*(0,0735)*	*(0,0110)*
MoE	0,5970	0,9266	0,6227	0,4261	1,3592	0,4197	0,6281	5,5258	2,9167
	(0,0331)	*(0,1009)*	*(0,0256)*	*(0,0179)*	*(0,2772)*	*(0,1476)*	*(0,1399)*	*(0,6817)*	*(0,0072)*
CSJADE	23,79	28,21	33,21	32,69	28,08	30,48	37,48	–	–
	(2,59)	*(3,16)*	*(2,92)*	*(3,62)*	*(2,23)*	*(4,06)*	*(2,91)*		
SFDEGL	5,0589	–	5,0598	–	7,7036	3,8941	–	–	–
BMGGAVS	0,6075	3,8357	0,7103	**0,1809**	6,5947	0,2730	1,0549	**0,3450**	0,5116
	(0,0525)	*(0,1782)*	*(0,0718)*	*(0,0181)*	*(0,3849)*	*(0,0215)*	*(0,0256)*	*(0,0233)*	*(0,0839)*
BMG2GA	0,6462	3,8669	0,6691	0,2699	5,0283	1,3972	2,7131	5,2832	9,1880
(1)	*(0,0442)*	*(0,2056)*	*(0,0617)*	*(0,0631)*	*(0,1991)*	*(0,1225)*	*(0,4274)*	*(0,4434)*	*(0,4970)*
BMG2GA	0,7370	4,6331	0,8305	0,4070	6,7599	1,8817	2,8206	5,5042	13,1321
(2)	*(0,0340)*	*(0,1694)*	*(0,0683)*	*(0,0352)*	*(0,5241)*	*(0,2076)*	*(0,1438)*	*(0,2320)*	*(0,2495)*
BMG2GA	0,6761	4,3428	0,7713	0,3080	5,5758	1,0709	2,6023	4,1725	9,6341
(3)	*(0,0472)*	*(0,5799)*	*(0,0647)*	*(0,0575)*	*(0,4698)*	*(0,2018)*	*(0,3438)*	*(0,1779)*	*(0,4135)*

Outra característica que pode ser notada na Tabela 5.16 é o fato da segunda configuração do BMG2GA sempre demorar mais que as outras configurações. Ainda é possível notar que a primeira configuração consegue normalmente ter um tempo de execução menor, em relação às outras. A explicação para o tempo menor da primeira configuração

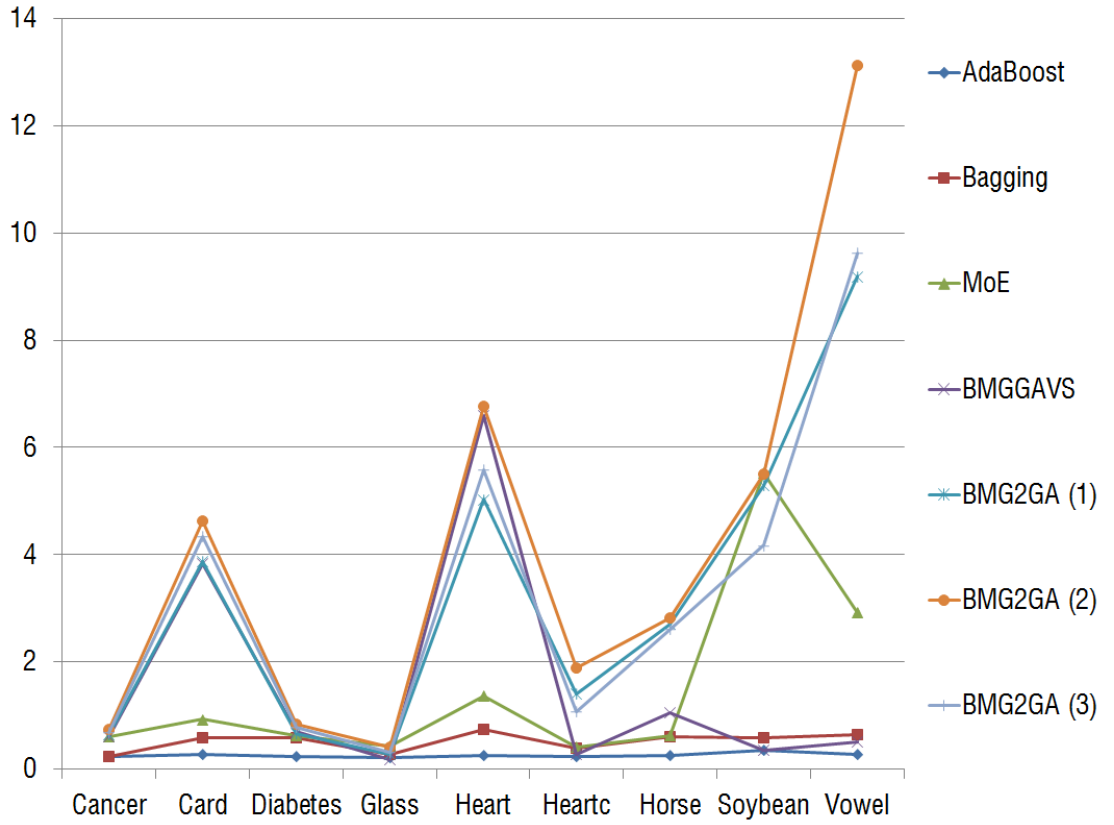

Figura 5.15 Comparação de médias de tempo em minutos entre métodos clássicos e o método proposto

é que as ELMs podem ter exercido alguma influência que fizesse o Algoritmo Genético convergir mais rapidamente a uma solução. A explicação para a segunda configuração ter levado um tempo maior está atrelada à explicação anterior, i.e., não tendo um classificador que pressionasse a convergência tanto quanto as ELMs, a busca por uma melhor solução demorava mais. Desta forma, ao ser retirado um dos tipos de classificadores (caso da terceira configuração) a convergência acontecia mais rapidamente, por haver menos combinações possíveis.

Para finalizar é possível concluir que, de forma análoga ao BMGGAVS, o BMG2GA não sofre bastante com o custo de tempo. Entretanto é necessário lembrar que o BMG2GA é dependente dos dados (executa mais rápido se os dados forem mais *fáceis*). Ele também sofre com problemas de escalabilidade, ou seja, quanto maior a base de dados e/ou quanto maior o número de atributos dos dados, mais tempo o método levará para finalizar uma execução.

5.4 Considerações finais

Os resultados do BMGGAVS foram satisfatórios, entretanto as Tabelas 5.2 e 5.3 mostram que há um ajuste manual para a obtenção dos melhores resultados. O grande número de ajustes pode ser visto como uma desvantagem. Apesar disso, tanto a baixa taxa de erro como um baixo custo de tempo fazem com que seja justificável a continuação da pesquisa.

Da continuação da pesquisa surgiu o BMG2GA. Seu principal foco foi a generalização do método. A primeira generalização foi a construção automática do grafo que sofrerá o agrupamento feito pelo BM-GGA. A outra generalização reside na seleção dos classificadores.

O BMG2GA foi dividido em três configurações, para se analisar em que situação o método modificado apresentaria os melhores resultados.

Infelizmente nenhuma das configurações conseguiu superar os métodos clássicos, o que mostra que ainda é necessário um esforço para melhorá-lo. Quanto aos demais métodos, somente as duas últimas configurações do BMG2GA conseguiram resultados próximos. Ainda, analogamente ao BMGGAVS, a segunda configuração do BMG2GA, que utiliza uma MLP com quinze nodos na camada escondida, sofreu *overfitting*.

Ademais, o aumento do erro pode ter sofrido influência por causa de uma modificação no pré-processamento. Enquanto que no BMGGAVS havia apenas uma divisão dos dados, fornecendo os mesmos subconjuntos em todas as iterações, no BMG2GA a divisão era feita a cada nova iteração. Como conseguência óbvia, o BMGGAVS tinha um número bem menor e fixo de possibilidades de agrupamento; logo foi mais *fácil* de encontrar resultados bons.

Em conclusão, as primeiras tentativas de modificação do BMGGAVS não tiveram muito sucesso, deixando o BMG2GA ainda a desejar. Contudo, como o método não ficou tão aquém dos métodos relacionados e similares, e o BMGGAVS provou conseguir bons resultados, ainda há motivação para que se continue uma pesquisa.

6

Conclusão

Reconhecer padrões é algo rotineiro aos seres humanos. Algumas vezes é uma atividade relativamente fácil. Por exemplo, o reconhecimento de faces é feito praticamente a todo instante sem que as pessoas se deem conta disso. Alguns tipos de padrões são processados pelas pessoas como parte de seu trabalho. Este é o caso de médicos que diagnosticam doenças de acordo com os sintomas característicos que um paciente apresenta. Entretanto, às vezes, a quantidade de informação chega a ser muito grande, ou ainda, as informações coletadas são difíceis de serem analisadas. Portanto é desejável que se forneça ao especialista humano uma ajuda que possa ser confiável, seja para ajudá-lo na tomada de alguma decisão, seja para substituí-lo.

Um esforço para que tal ajuda venha a existir vem sendo empregado na área de Aprendizagem de Máquina. Pesquisas que envolvem Inteligência Artificial e Computacional produzem algoritmos capazes de reconhecer padrões, utilizando classificadores. Entretanto, muitos destes algoritmos, ao utilizarem apenas um classificador, não são suficientes para vários problemas do mundo real, que requerem um alto nível de precisão.

Em face desta dificuldade, foi proposto o uso de comitês de classificadores. Desta forma, não apenas um, mas vários classificadores trabalham em conjunto para o reconhecimento e classificação de padrões. Dado o sucesso da proposta, foram iniciadas pesquisas cujo objetivo é conseguir explorar, cada vez mais, a potencialidade dos comitês. Para isso novas abordagens foram estudadas. Foram estudadas também maneiras para combinar tais abordagens com outras técnicas [WGC14][Rok09].

Para contribuir com a área de reconhecimento de padrões este trabalho estudou o comportamento de comitês de classificadores por uma abordagem chamada de Seleção de Classificadores. A inovação se faz presente no uso do BM-GGA para agrupar os padrões, e no uso de um sistema para a seleção dos classificadores.

A contribuição deste trabalho pode ser dividida em duas partes. A primeira parte contou com a proposição de um método, o qual foi chamado de BMGGAVS. Nele um sistema simples de votação é o responsável pela seleção de classificadores.

Levando em conta os melhores resultados obtidos foi possível concluir que, dada a combinação correta dos valores dos parâmetros, o método proposto tem um enorme potencial. O BMGGAVS alcançou um melhor desempenho que todos os demais métodos usados nas comparações, além de não ter um alto custo de tempo. Contudo, esse bom resultado esbarra no ajuste manual que é necessário para o mesmo. O ajuste, provavelmente, demandaria muito tempo para o encontro de bons resultados em outras bases de dados.

Desta primeira parte da contribuição foi publicado o artigo *Clustering and selection using grouping genetic algorithms for blockmodeling to construct neural network ensembles*, no evento *IEEE International Conference on Tools with Artificial Intelligence*, no ano de 2013 [SLA13].

A segunda parte da contribuição consistiu na evolução do método, onde se buscou deixar o mesmo completamente automatizado. Para isso, a atenção se voltou diretamente para os padrões de treinamento e suas propriedades em relação aos demais padrões. Além disso, em vez de escolher um indutor base fixo e variar um segundo, uma maior gama de indutores foi escolhida e todos foram utilizados. No sistema de votação havia quatro votos (i.e., os votos 1, 4, 5 e 6) que comprovadamente tinham influência significativa no resultado final, e nos testes os critérios eram modificados manualmente. Por causa disso houve uma substituição do sistema de votação por uma busca via Algoritmo Genético para a seleção dos classificadores.

Infelizmente os resultados do BMG2GA, evolução do BMGGAVS, não foram tão bons quanto os do último. Em nenhuma das configurações testadas o BMG2GA conseguiu um desempenho melhor que os métodos clássicos. Ainda assim, em relação aos métodos similares, o BMG2GA conseguiu alguns bons resultados, mostrando que de certa forma

possui um potencial. Logo, é desejável que se continue a pesquisa para averiguar como o método irá se comportar após novas modificações, sendo estas, baseadas nos resultados obtidos até então.

6.1 Trabalhos Futuros

Durante o desenvolvimento deste trabalho foram identificadas ideias de melhorias com pontencial para futuras pesquisas:

- Ainda que a vizinhança de cada padrão seja mapeada considerando as propriedades do mesmo em relação aos outros (equação (4.5)), foi observado que uma melhoria poderia ser implementada. De forma mais objetiva há o exemplo da base *Glass*, onde em boa parte das iterações, cada padrão tomava como vizinhos todos os demais, finalizando por haver um único *cluster*. Isso aconteceu porque o divisor da fórmula alcançava um valor muito baixo. Sendo o dividendo o valor 1, o valor final da distância acabava por ser maior que a maior distância entre dois padrões.

  É possível perceber que a falta de agrupamento é algo que deveria ser evitado. O que era buscado era a divisão dos padrões em conjuntos menores. Em outras bases houve situações parecidas e até contrárias. Ou seja, houve situações onde cada padrão tinha um número muito alto de vizinhos, como também houve situações em que cada padrão tinha quase nenhum vizinho.

  Portanto, como trabalho futuro: (1) a criação de uma nova fórmula, ou somente a alteração da existente (equação (4.5)); o objetivo é evitar a dependência da variação de distância que existe entre os padrões de bases de dados diferentes; e (2) o uso do algoritmo PCA (*Principal Component Analyses*) [AW10] ou outro algoritmo semelhante que possa selecionar um número menor de atributos que favoreça o agrupamento.

- Após o agrupamento do conjunto de treinamento, os *clusters* construídos são completados com dados de validação e teste. Um trabalho futuro, neste ponto, poderia analisar que impacto haveria o agrupamento sobre todos os dados, i.e., sobre os três subconjuntos ao mesmo tempo. Poderia também ser proposta uma nova alternativa para preencher os *clusters*. Outra possibilidade seria juntar ou

separar os *clusters* dinamicamente, de acordo com algum conhecimento prévio adquirido sobre os dados.

É importante lembrar também que houve uma modificação entre os dois métodos construídos neste trabalho; mais especificamente quando padrões do conjunto de teste final são atribuídos aos *clusters* à medida que os mesmos são preenchidos. Enquanto no BMGGAVS tais padrões são atribuídos após a atualização dos centroides dos *clusters*, no BMG2GA a atribuição acontece antes. Infelizmente não foi possível determinar se tal mudança surtiu algum efeito no resultado final. Como trabalho futuro pode haver uma análise neste sentido. Além disso, pode ser analisado o efeito da inserção dos padrões de teste final às suas respectivas regiões, ou *clusters*, somente após os mesmos estarem totalmente preenchidos.

- Tanto no BMGGAVS quanto no BMG2GA, após o preenchimento dos *clusters*, o treinamento dos classificadores é iniciado. Neste trabalho, um classificador de cada tipo é criado para cada *cluster*. Logo, havendo *k clusters*, haverá *k* classificadores de cada tipo, e cada um será treinado com os dados do seu respectivo *cluster*. Posteriormente, cada classificador concorrerá somente à classificação do *cluster* ao qual está assinalado. É possível perceber, então, que os classificadores estão "presos" aos seus *clusters*. Portanto, um trabalho futuro deverá permitir que todos os classificadores criados tenham a chance de serem selecionados para qualquer *cluster*.

- Os classificadores do tipo MLP e ELM tiveram cinco, dez ou quinze nodos em suas camadas intermediárias. Um trabalho futuro poderia determinar o número de nodos dinamicamente. Ou então utilizar mais nodos, estaticamente, e verificar se há uma influência para a obtenção de resultados melhores.

- Atualmente um Algoritmo Genético é utilizado para a seleção. Este AG leva em consideração os erros de treinamento, validação e teste. A combinação de classificadores que conseguir o menor erro será escolhida. Entretanto, quando foram utilizadas ELMs como classificadores, a função de aptidão teve de ser modificada, prejudicando o método. Como trabalho futuro, é desejável a proposta de uma nova função de aptidão que consiga fazer uma melhor seleção de classificadores; o objeto é alcançar melhores resultados. Ainda, pode haver a substituição do Algoritmo Genético por qualquer outra técnica, sacrificando, o mínimo possível, a rapidez e simplicidade. Uma alternativa para substituir o AG, seria o emprego em conjunto de seleção e fusão dos classificadores.

- A última observação ocorreu nos resultados únicos do BMG2GA. Por várias vezes o método conseguia um erro de 0% em alguma iteração. Ou seja, o método conseguia acertar a classificação de todas as instâncias desconhecidas. Como a cada nova repetição os dados disponíveis eram diferentes, acreditamos que algum algoritmo que possa separar os dados baseado em alguma heurística possa favorecer os resultados obtidos. Neste trabalho a construção dos conjuntos é feita aleatoriamente, havendo apenas o cuidado de deixar as classes balanceadas nos conjuntos. Um trabalho futuro poderia estender esses cuidados.

Por fim, sendo implementadas as propostas acima, é esperado que se encontre uma maneira completamente automática e eficiente de agrupar os dados e selecionar classificadores para a construção de um comitê, que, no mínimo, servirá de base para mais pesquisas. À medida que os testes forem executados, várias nuances serão percebidas e catalogadas, de forma que as mesmas poderão ajudar positivamente em futuras pesquisas.

Referências

[ABL78] Arabie, Phipps, Scott A. Boorman e Paul R. Levitt: *Constructing Blockmodels: How and Why*. Journal of Mathematical Psychology, 17:21–63, 1978.

[Alm07] Almeida, Leandro Maciel: *Uma metodologia de busca por redes neurais artificiais quase-ótimas*. Dissertação de Mestrado, UFPE, 2007.

[Alm11] Almeida, Leandro Maciel: *Construção de Sistemas de Múltiples Classificadores por meio da Hibridização e Otimização de Técnicas de Agrupamento e Classificação de Dados*. Tese de Doutorado, UFPE, 2011.

[AN07] Asuncion, A. e D.J. Newman: *UCI Machine Learning Repository*, 2007. http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html.

[AW10] Abdi, Hervé e Lynne J. Williams: *Principal component analysis*. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4):433–459, 2010, ISSN 1939-0068. http://dx.doi.org/10.1002/wics.101.

[BDSS13] Brusco, Michael, Patrick Doreian, Douglas Steinley e Cinthia B. Satornino: *Multiobjective Blockmodeling for Social Network Analysis*. Psychometrika, 78(3):498–525, 2013.

[BLC00] Braga, Antônio de Pádua, Teresa Bernarda Ludermir e André Ponce de Leon F. de Carvalho: *Redes Neurais Artificiais: Teoria e Aplicações*. LTC, Rio de Janeiro, RJ, Brasil, 2000, ISBN 9788521615644.

[BM93] Babu, G. Phanendra e M. Narasimha Murty: *A near-optimal initial seed value selection in K-means algorithm using a genetic algorithm*. Pattern Recognition Letters, 14(10):763–769, 1993.

[BMFD04] Batagelj, Vladimir, Andrej Mrvar, Anuška Ferligoj e Patrick Doreian: *Generalized Blockmodeling with Pajek*. Metodološki zvezki, 1(2):455–467, 2004.

[CLH11] Chan, Jeffrey, Samantha Lam e Conor Hayes: *Increasing the scalability of the fitting of generalised block models for social networks*. Em *Proceedings*

of the Twenty-Second international joint conference on Artificial Intelligence - Volume Two, IJCAI'11, páginas 1218–1224. AAAI Press, 2011, ISBN 978-1-57735-514-4.

[CXC99] Chen, K., L. Xu e H. Chi: *Improved learning algorithms for mixture of experts in multiclass classification*. Neural Networks, 12:1229–1252, 1999.

[CY05] Chandra, Arjun e Xin Yao: *Ensemble Learning Using Multi-objective Evolutionary Algorithms*. Journal of Mathematical Modelling and Algorithms, 5(4):417–445, 2005.

[Cyb88] Cybenko, G.: *Continuous valued neural networks with two hidden layers are sufficient*. Relatório Técnico, Department of Computer Science, Tufts University, 1988.

[Cyb89] Cybenko, G.: *Approximation by superpositions of a sigmoid function*. Mathematics of Control, Signals and Systems, 2:303–314, 1989.

[CZZ09] Chang, Dong Xia, Xian Da Zhang e Chang Wen Zheng: *A genetic algorithm with gene rearrangement for K-means clustering*. Pattern Recognition, 42(7):1210–1222, 2009.

[DBF04] Doreian, Patrick, Vladimir Batagelj e Anuska Ferligoj: *Generalized Blockmodeling*. Cambridge University Press, New York, NY, USA, 2004, ISBN 0521840856.

[ELL01] Everitt, Brian S., Sabine Landau e Morven Leese: *Cluster Analysis*. Arnold, Londres, 2001, ISBN 0340761199.

[Fal92] Falkenauer, A.: *The grouping genetic algorithms widening the scope of the GAs*. JORBEL Belgian J. Oper. Res., Stat. Comput. Sci., 33(1–2):79–102, 1992.

[FS96] Freund, Yoav e Robert E. Schapire: *Experiments with a New Boosting Algorithm*. Em Saitta, Lorenza (editor): *ICML*, páginas 148–156. Morgan Kaufmann, 1996, ISBN 1-55860-419-7.

[Her50] Herfindahl, O. C.: *Concentration in the U.S. steel industry*. Tese de Doutorado, Columbia University, New York, 1950.

[HHS94] Ho, Tin Kam, Jonathan J. Hull e Sargur N. Srihari: *Decision Combination in Multiple Classifier Systems*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(1):66–75, 1994.

[Hir64] Hirschman, A. O.: *The paternity of an index*. The American Economic Review, 54:761–762, 1964.

[HS95] Huang, Y. S. e C. Y. Suen: *A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(1):90–94, 1995.

[Hu01] Hu, Xiaohua: *Using Rough Sets Theory and Database Operations to Construct a Good Ensemble of Classifiers for Data Mining Applications*. Em *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, páginas 233–240, Washington, DC, USA, 2001. IEEE Computer Society, ISBN 0-7695-1119-8. http://dl.acm.org/citation.cfm?id=645496.657724.

[HZS06] Huang, Guang Bin, Qin Yu Zhu e Chee Kheong Siew: *Extreme learning machine: Theory and applications*. Neurocomputing, 70(1–3):489–501, 2006.

[JBR10] James, Tabitha, Evelyn Brown e Cliff T. Ragsdale: *Grouping genetic algorithm for the blockmodel problem*. Trans. Evol. Comp, 14(1):103–111, fevereiro 2010, ISSN 1089-778X.

[JJ94] Jordan, Michael I. e Robert A. Jacobs: *Hierarchical Mixtures of Experts and the EM Algorithm*. Neural Comput., 6(2):181–214, março 1994, ISSN 0899-7667. http://dx.doi.org/10.1162/neco.1994.6.2.181.

[JJNH91] Jacobs, Robert A., Michael I. Jordan, Steven J. Nowlan e Geoffrey E. Hinton: *Adaptive Mixtures of Local Experts*. Neural Comput., 3(1):79–87, março 1991, ISSN 0899-7667. http://dx.doi.org/10.1162/neco.1991.3.1.79.

[KHDM98] Kittler, Josef, Mohamad Hatef, Robert P.W. Duin e Jiri Matas: *On Combining Classifiers*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3):226–239, 1998.

[KKMA11] Kim, Hyunjoong, Hyeuk Kim, Hojin Moon e Hongshik Ahn: *A weight-adjusted voting algorithm for ensembles of classifiers*. Journal of the Korean Statistical Society, 40(4):437–449, dezembro 2011.

[Kun93] Kuncheva, Ludmila I.: *'Change-glasses' approach in pattern recognition*. Pattern Recognition Letters, 14(8):619–623, 1993.

[Kun00] Kuncheva, Ludmila I.: *Clustering-and-selection model for classifier combination*. Em Howlett, Robert J. e Lakhmi C. Jain (editores): *KES*, páginas 185–188. IEEE, 2000, ISBN 0-7803-6400-7.

[Kun04] Kuncheva, Ludmila I.: *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004, ISBN 0471210781.

[KW03] Kuncheva, Ludmila I. e Christopher J. Whitaker: *Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy*. Mach. Learn., 51(2):181–207, maio 2003, ISSN 0885-6125. http://dx.doi.org/10.1023/A:1022859003006.

[Lev44] Levenberg, Kenneth: *A method for the solution of certain non-linear problems in least squares*. Quarterly Journal of Applied Mathmatics, 2(2):164–168, 1944.

[LLFM98] Lo, Shih Chung B., Jyh Shyan J. Lin, Matthew T. Freedman e Seong K. Mun: *Application of Artificial Neural Networks to Medical Image Pattern Recognition: Detection of Clustered Microcalcifications on Mammograms and Lung Cancer on Chest Radiographs*. Journal of VLSI signal processing systems for signal, image and video technology, 18(3):263–274, abril 1998.

[LSL12a] Lima, Tiago P. F., Adenilton J. Silva e Teresa Bernarda Ludermir: *Clustering and selection of neural networks using adaptive differential evolution*. Em *IJCNN*, páginas 747–753. IEEE, 2012, ISBN 978-1-4673-1488-6.

[LSL12b] Lima, Tiago P. F., Adenilton J. Silva e Teresa Bernarda Ludermir: *Selection and Fusion of Neural Networks via Differential Evolution*. Em *IBERAMIA*, volume 7637, páginas 149–158. Springer, 2012, ISBN 978-3-642-34653-8.

[LY01] Liu, Rujie e Baozong Yuan: *Multiple classifiers combination by clustering and selection*. Information Fusion, 2(3):163–168, 2001.

[Mar63] Marquardt, Donald W.: *An algorithm for least-squares estimation of nonlinear parameters*. SIAM Journal on Applied Mathematics, 11(2):431–441, 1963. http://dx.doi.org/10.1137/0111030.

[ML06] Minku, A L. e Teresa B. Ludermir: *EFuNNs ensembles construction using a clustering method and a coevolutionary genetic algorithm*. Em *CEC2006*, páginas 5548–5555, 2006.

[MM04] Melville, Prem e Raymond J. Mooney: *Creating diversity in ensembles using artificial data*. Information Fusion, 6:99–111, 2004.

[MP69] Minsky, M. e S. Papert: *Perceptrons: an introduction to computational geometry*. MIT Press, Massachusetts, 1969.

[NG04] Newman, M. E. J. e M. Girvan: *Finding and evaluating community structure in networks*. Phys. Rev. E, 69(2):026113, fevereiro 2004. http://link.aps.org/doi/10.1103/PhysRevE.69.026113.

[Pol06] Polikar, R.: *Ensemble based systems in decision making*. IEEE Circuits and Systems Magazine, 6(3):21–45, 2006, ISSN 1531-636X.

[Pon11] Ponti, M. P.: *Combining Classifiers: From the Creation of Ensembles to the Decision Fusion*. 24th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), páginas 1–10, 2011.

[RG05] Ruta, Dymitr e Bogdan Gabrys: *Classifier selection for majority voting*. Information Fusion, 6(1):63–81, 2005.

[Rok09] Rokach, Lior: *Taxonomy for characterizing ensemble methods in classification tasks A review and annotated bibliography*. Computational Statistics & Data Analysis, 53(12):4046—-4072, Outubro 2009.

[Rok10] Rokach, Lior: *Ensemble-based Classifiers*. Artif. Intell. Rev., 33(1-2):1–39, fevereiro 2010, ISSN 0269-2821. http://dx.doi.org/10.1007/s10462-009-9124-7.

[Ros58] Rosenblatt, F.: *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychol. Rev., 65:386–408, 1958.

[Ros62] Rosenblatt, F.: *Principies of Neurodynamics: Perceptrons and the theory of brain mechanisms*. Spartan Books, New York, 1962.

[SLA13] Silva, Evandro José da Rocha e, Teresa B. Ludermir e Leandro Maciel Almeida: *Clustering and Selection Using Grouping Genetic Algorithms for Blockmodeling to Construct Neural Network Ensembles*. Em *ICTAI*, páginas 420–425. IEEE, 2013.

[SSF10] Silva, I. N., D. H. Spatti e R. A. Flauzino: *Redes Neurais Artificiais: para engenharia e ciências aplicadas*. Artliber, São Paulo, SP, Brasil, 2010, ISBN 8588098539.

[SYK97] Sarkar, Manish, B. Yegnanarayana e Deepak Khemani: *A clustering algorithm using an evolutionary programming-based approach*. Pattern Recognition Letters, 18(10):975–986, 1997.

[WBB76] White, Harrison C., Scott A. Boorman e Ronald L. Breiger: *Social Structure from Multiple Networks*. American Journal of Sociology, 81:730–780, 1976.

[Web00] Webb, Geoffrey I.: *MultiBoosting: A Technique for Combining Boosting and Wagging*. Mach. Learn., 40(2):159–196, 2000, ISSN 0885-6125.

[WGC14] Woźniak, Michał, Manuel Graña e Emilio Corchado: *A survey of multiple classifier systems as hybrid systems*. Information Fusion, 16:3–17, 2014.

[Wil45] Wilcoxon, Frank: *Individual Comparisons by Ranking Methods*. Biometrics Bulletin, 1(6):80–83, 1945.

[XKS92] Xu, Lei, Adam Krzyzak e Ching Y. Suen: *Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition*. IEEE Transactions on Systems, Man, and Cybernetics, 22(3):418–435, 1992.

[YWG12] Yuksel, Seniha Esen, Joseph N. Wilson e Paul D. Gader: *Twenty Years of Mixture of Experts*. IEEE Transactions on Neural Networks and Learning Systems, 23(8):1177–1193, 2012.