UNIVERSIDADE FEDERAL DO PARANÁ

JULIANO FABIANO DA MOTA

META-HEURÍSTICAS BASEADAS EM POPULAÇÃO PARA O TREINAMENTO DE REDES NEURAIS DE BASE RADIAL NO CONTEXTO DE INTELIGÊNCIA COMPUTACIONAL: TEORIA E IMPLEMENTAÇÕES

CURITIBA
2012

JULIANO FABIANO DA MOTA

META-HEURÍSTICAS BASEADAS EM POPULAÇÃO PARA O TREINAMENTO DE REDES NEURAIS DE BASE RADIAL NO CONTEXTO DE INTELIGÊNCIA COMPUTACIONAL: TEORIA E IMPLEMENTAÇÕES

Tese apresentada ao Curso de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração Programação Matemática, do Departamento de Matemática, Setor de Ciências Exatas e do Departamento de Construção Civil, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para a obtenção do título de Doutor em Ciências.

Orientador: Prof. Dr. Paulo Henrique Siqueira

Co-Orientadora: Prof[a]. Dr[a]. Luzia Vidal de Souza

CURITIBA

2012

TERMO DE APROVAÇÃO

JULIANO FABIANO DA MOTA

META-HEURÍSTICAS BASEADAS EM POPULAÇÃO PARA O TREINAMENTO DE REDES NEURAIS DE BASE RADIAL NO CONTEXTO DE INTELIGÊNCIA COMPUTACIONAL: TEORIA E IMPLEMENTAÇÕES

Tese aprovada como requisito parcial para a obtenção do grau de Doutor em Ciências, área de concentração Programação Matemática no Programa de Pós-Graduação em Métodos Numéricos em Engenharia, do Departamento de Matemática, Setor de Ciências Exatas e do Departamento de Construção Civil, Setor de Tecnologia, Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador: Prof. Dr. Paulo Henrique Siqueira
Programa de Pós-Graduação em Métodos
Numéricos em Engenharia, UFPR

Co-Orientadora: Profª. Drª. Luzia Vidal de Souza
Programa de Pós-Graduação em Métodos
Numéricos em Engenharia, UFPR

Profª. Drª. Deise Maria Bertholdi Costa
Programa de Pós-Graduação em Métodos
Numéricos em Engenharia, UFPR

Profª. Drª. Sonia Isoldi Marty Gama Muller
Programa de Pós-Graduação em
Engenharia de Produção, UFPR

Profª. Drª. Angela Olandoski Barboza
Universidade Tecnológica Federal
do Paraná, UTFPR

Curitiba, 07 de dezembro de 2012.

Aos meus pais, João e Enilda.

AGRADECIMENTOS

A Deus.

Aos meus pais, irmã e à minha esposa, Franciely, por todo amor, incentivo e por torcerem tanto por mim.

Aos meus amigos Adriano, Gislaine, Solange e Tatiane, pelo companheirismo, pelo apoio dado nos momentos difíceis e pelas risadas, principalmente das “piadinhas genéricas” que sempre surgiam.

À D. Aparecida, por ter me recebido tão bem em sua casa durante os quatro anos de doutorado.

Ao meu orientador, Professor Paulo, pelos ensinamentos, amizade, motivação e dedicação ao meu trabalho, sempre apresentando valiosas sugestões.

À minha co-orientadora, Professora Luzia, pelo aprendizado proporcionado durante nossos seminários e pelas valiosas contribuições dadas ao trabalho.

Ao Professor Anselmo Chaves Neto, pelo apoio e incentivo e por sempre ter acreditado na “Turma de Campo Mourão”.

À Universidade Federal do Paraná, pela oportunidade de cursar o doutorado.

Aos professores do Programa de Pós-Graduação em Métodos Numéricos em Engenharia, pelos ensinamentos transmitidos durante as disciplinas.

À Maristela Bandil, uma das pessoas mais simpáticas e gentis que já conheci, pela alegria e eficiência com as quais realiza seu trabalho. E como se não bastasse tudo isso, faz um café maravilhoso!

À Universidade Estadual do Paraná, Campus Campo Mourão, por me proporcionar condições necessárias para concluir este curso.

Finalmente, à Fundação Araucária, pelo apoio financeiro.

Não basta ensinar ao homem uma especialidade. Porque se tornará assim uma máquina utilizável, mas não uma personalidade. É necessário que adquira um sentimento, um senso prático daquilo que vale a pena ser empreendido, daquilo que é belo, do que é moralmente correto. A não ser assim, ele se assemelhará, com seus conhecimentos profissionais, mais a um cão ensinado do que a uma criatura harmoniosamente desenvolvida. Deve aprender a compreender as motivações dos homens, suas quimeras e suas angústias para determinar com exatidão seu lugar exato em relação a seus próximos e à comunidade.

Albert Einstein em “Como Vejo o Mundo”.

RESUMO

Um dos problemas da modelagem de uma RBFNN - *Radial Basis Neural Network*, Rede Neural de Base Radial, consiste em determinar os pesos da camada de saída, geralmente representados por uma matriz retangular. Uma abordagem que tem ganho alguma notoriedade recentemente na resolução desse problema é a criação de modelos híbridos baseados na combinação de Meta-heurísticas, que são modelos gerais para solução de problemas de otimização, como alternativa ao método tradicional de realizar a pseudo-inversão da matriz com os valores de ativação da camada intermediária. Nesta pesquisa, duas destas Meta-heurísticas, Algoritmos Genéticos e Nuvem de Partículas (*Particle Swarm Optmization*) são implementadas a fim de comparar seus desempenhos com o método tradicional e também é proposta a mudança da representação dos indivíduos de uma população, em Algoritmos Genéticos, com a consequente adaptação operadores para algoritmos genéticos contínuos em que os indivíduos são matrizes, como é o caso do problema de calcular a matriz de pesos de uma RBFNN. Além disso, essas técnicas também são comparadas com a FDLF - Função Discriminante Linear de Fisher na classificação de padrões. Para fins de validação da hipótese levantada, foi realizado um experimento com seis bancos de dados e os resultados mostraram que as abordagens mais eficientes foram o treinamento tradicional das RBFNN e a FDLF, já a modificação proposta se mostrou tão consistente quanto o Algoritmo Genético tradicional no que diz respeito à eficiência ao encontrar soluções.

Palavras-chave: Redes Neurais de Base Radial, Algoritmos Genéticos, Nuvem de Partículas, Sistemas Híbridos.

ABSTRACT

One of the issues of modeling a RBFNN - Radial Basis Neural Network is to determine the weights matrix of the output layer, wich is generally represented by a rectangular matrix. One of the existing approaches that has gained some notoriety recently in solving this problem is to create hybrid models based on a combination of Metaheuristcs, which are general models for solving optimization problems, as an alternative to the traditional method of performing the pseduoinversion the hidden layer activation values matrix. In this research two of these Metaheuristcs, Genetic Algorithms (GA) and Particle Swarm Optmization (PSO) are implemented in order to compare their performance with the traditional method and also a new kind representation of individuals in a population for genetic algorithms is proposed. This caused the adaptation of the continuous genetic operators in genetic algorithms with matricial individuals, specially for calculating the weights matrix of a RBFNN. Also a comparison of all these techniques with the LDFF - Linear Discriminant Fisher Function For is provided. Trying to validating the research hypothesis an experiment was designed using six diffrerent databases and results shows that the traditional training method of RBFNN and the Fisher LDF was more effective than the others techniques. The new proposed GA individuals representation was equally consistent with the traditional GA in finding solutions.

Key-words: Radial Basis Function Neural Networks, Genetic Algorithms, Particle Swarm Optimization, Hybrid Systems.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Há tempos os cientistas tentam desenvolver métodos para prever valores em problemas de classificação para uma melhor tomada de decisão em cenários de incerteza. Uma gama de métodos matemáticos e estatísticos já foi desenvolvida e, por este motivo, busca-se melhorar cada vez mais os métodos já existentes bem como o desenvolvimento de novos métodos que reduzam o custo computacional, ofereçam melhores resultados no que diz respeito ao desempenho do modelo ou ambos.

Todos os métodos existentes já oferecem muitas opções quanto à velocidade e qualidade da previsão ou classificação. A tarefa mais difícil, que todo pesquisador anseia, é desenvolver um método que consiga elevada acurácia o mais rápido possível, dada a necessidade de rapidez na era *on-line* em que a sociedade se encontra. Por "elevada acurácia" entende-se o menor erro possível, num problema de previsão, e o maior percentual possível de classificações corretas, para o caso de um problema de classificação.

A idéia de criar um grupo de modelos matemáticos que simule os neurônios humanos é considerada, por muitos cientistas, brilhante e extremamente útil para os responsáveis pela tomada de decisão nas mais diversas áreas do conhecimento. Tais modelos são as Redes Neurais Artificiais. Segundo Haykin (2001), uma rede neural é uma máquina de processamento paralelo que pode transformar conhecimento experimental em informações utilizáveis na prática, por exemplo, no auxílio à tomada de decisão.

O subgrupo de modelos de Redes Neurais Artificiais mais difundido na literatura é aquele das baseadas no tipo de arquitetura *feed-forward* (alimentada adiante) ao qual pertencem, por exemplo, o *Perceptron* e as Redes Neurais de Base Radial, sendo a primeira, mais conhecida e difundida entre os cientistas da área e a última um dos focos deste trabalho.

Uma Meta-heurística é um Método Heurístico para resolver, de forma genérica, problemas de Pesquisa Operacional, normalmente da área de Otimização Combinatória, Reconhecimento de Padrões ou Previsão de Séries Temporais. Entretanto, também é possível utilizar

uma Meta-heurística para Problemas de Otimização de Funções.

Geralmente, as Meta-heurísticas são inspiradas em outras ciências, como é o caso das próprias Redes Neurais Artificiais, dos Algoritmos Genéticos, *Simulated Annealing*, Colônia de Formigas, Nuvem de Partículas, entre outros, formando assim um sistema com regras matemáticas inspiradas em sistemas, a *priori*, não matemáticos .

Como ocorre com outras técnicas matemáticas e estatísticas, é possível a formação de Sistemas Híbridos baseados em Meta-heurísticas, combinando as estratégias de duas ou mais dessas técnicas. Ademais, este campo de estudo tem ganho alguma notoriedade recentemente, como é possível observar pela quantidade de trabalhos publicados sobre este tema (Seção 2.6). Existem também métodos estatísticos para a classificação de padrões, sendo que o mais conhecido é a FDLF - Função Discriminante Linear de Fisher, descrita no Capítulo 2.

Um dos principais problemas a ser resolvido na modelagem de uma Rede Neural de Base Radial é o cálculo da matriz de pesos da camada intemediária, que é a única camada presente numa rede desse tipo além da camada de saída.

1.1 OBJETIVOS

A pesquisa de Mota et al. (2012) compara dois métodos de treinamento de uma RBFNN, um é considerado tradicional, que se baseia na pseudo-inversão de uma matriz retangular e o outro utiliza Algoritmos Genéticos, convertendo esta matriz em indivíduos de uma população genética, buscando encontrar a matriz ótima (ou quase-ótima) através de seleção natural e operadores genéticos, mais especificamente os propostos por Michalewicz, Logan e Swaminathan (1994).

Nesse contexto, os principais objetivos deste trabalho são:

- propor uma nova maneira de representação dos indivíduos em Algoritmos Genéticos, para o caso em que as soluções do problema são matrizes ao invés de vetores;
- propor uma extensão dos operadores de Michalewicz, Logan e Swaminathan (1994) para esse novo modo de representação representação;

- comparar o desempenho de seis abordagens para a classificação de padrões em seis bancos de dados disponíveis na *web* que atenderam os critérios de aplicação dos métodos, sendo:
  - a Função Discriminante Linear de Fisher;
  - o treinamento de uma RBFNN tradicional (pseudo-inversão);
  - duas abordagens de treinamento de uma RBFNN baseadas em Algoritmos Genéticos, sendo que a primeira é um Algoritmo Genético Tradicional utilizando os operadores de Michalewicz, Logan e Swaminathan (1994) e a segunda é a adaptação proposta neste trabalho;
  - duas abordagens de treinamento de uma RBFNN utilizando a Metaheurística "Nuvem de Partículas", sendo uma abordagem com cada uma de suas duas variantes mais conhecidas: *gbest* e *lbest*.

1.2 CONTRIBUIÇÕES

As contribuições deste trabalho são:

- Uma verificação empírica da eficiência de duas técnicas de Inteligência Computacional para o treinamento de uma RBFNN e uma técnica estatística para fins de classificação de padrões;
- Uma nova maneira de representar e "enxergar" os indivíduos de uma população genética e, consequentemente, a extensão dos operadores propostos por Michalewicz, Logan e Swaminathan (1994);

1.3 ESTRUTURA DO TRABALHO

Para que os objetivos elencados possam ser alcançados e a hipótese principal possa ser verificada, preceitos básicos sobre Inteligência Computacional, Redes Neurais de Base Radial, Função Discriminante Linear de Fisher, Algoritmos Genéticos e Nuvem de Partículas são apresentados e uma verificação experimental é realizada. Todas estas etapas do trabalho estão divididas em capítulos, da seguinte maneira:

- No Capítulo 2, são apresentados, de maneira breve, os principais sistemas da Inteligência Computacional (IC), afim de indicar sua classificação como área de conhecimento, explicitar as técnicas que a compõem e discutir como estas técnicas podem ser utilizadas de maneira isoladas ou com a criação de sistemas híbridos. Também estão contemplados os conceitos relativos às Redes Neurais de Base Radial, destacando suas características e estratégias de modelagem e treinamento e conceitos essenciais sobre a Função Discriminante Linear de Fisher (FDLF), ambas no contexto de classificação de padrões e os conceitos relacionados às duas Meta-heurísticas de Busca Baseadas em População citadas anteriormente: Algoritmos Genéticos e Nuvem de Partículas;
- O Capítulo 3, trata das estratégias utilizadas para criar uma Meta-heurística Híbrida baseada em Redes Neurais de Base Radial e Algoritmos Genéticos e Nuvem de Partículas, são apresentados detalhes sobre o experimento realizado
- O Capítulo 4 traz a análise estatística dos resultados obtidos e apresenta uma discussão acerca desses resultados;
- Finalmente, o Capítulo 5 traz as conclusões da pesquisa e sugestões para trabalhos futuros.

2 REVISÃO DE LITERATURA

Para Engelbrecht (2007), “um grande impulso no desenvolvimento de novos algoritmos é a crescente complexidade dos problemas que surgem a cada dia”. Assim como enxerga-se problemas mais complexos a cada dia, numa velocidade quase equivalente os cientistas são capazes de desenvolver métodos para resolver tais problemas. Esse desenvolvimento, por vezes, ocorre com o desenvolvimento de algoritmos, uma vez que o advento do computador tem como um de seus objetivos auxiliar em tarefas complexas para que antes eram realizadas manualmente.

Uma das vertentes na busca pela automatização de tarefas complexas é a criação de métodos e sistemas baseados na inteligência natural e biológica que Engelbrecht (2007) denomina como “Sistemas Inteligentes”. Basicamente existem quatro grandes grupos de sistemas inteligentes: Redes Neurais Artificiais, Computação Evolucionária, Inteligência do Enxame e Sistemas Difusos. Vale salientar que juntamente com Lógica; Sistemas Especialistas; Raciocínio Baseado em Casos e Dedutivo; e Máquinas de Sistemas de Aprendizagem Simbólica, os Sistemas Inteligentes pertencem ao campo de estudo denominado Inteligência Artificial (IA).

Os quatro sistemas mais conhecidos baseados em Inteligência Computacional (IC) são: As Redes Neurais Artificiais (RNAs), a Computação Evolucionária (CE), a Inteligência do Enxame (IE) e os Sistemas Difusos (SD). Tais sistemas podem ser utilizados como única abordagem a um problema, dependendo de suas características, ou podem ainda serem combinados formando sistemas híbridos.

A Figura 1 ilustra esta ideia de relacionamento desses quatro modelos por meio de setas, podendo haver relações entre seus mecanismos internos. Também é possível visualizar a indicação da interação dos quatro grupos de sistemas com os Métodos Estatísticos que, essencialmente, são utilizados para tratamento dos dados e validação de testes, por isso a diferenciação com linhas pontilhadas, embora haja métodos estatísticos tanto para a

Classificação de Padrões quanto para a previsão de Séries Temporais.

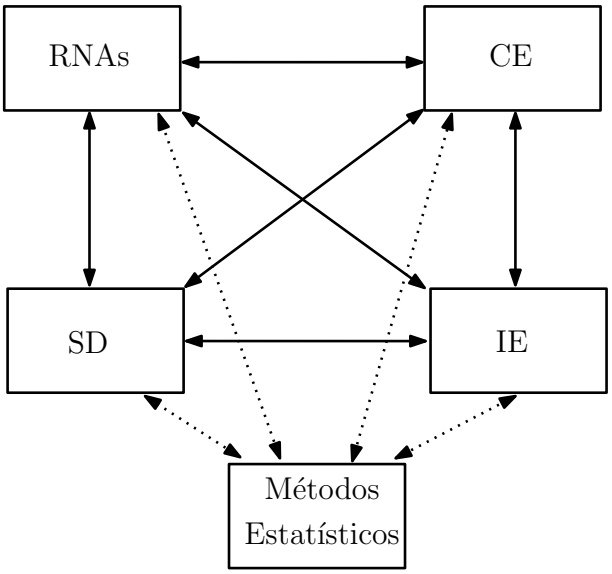


FIGURA 1: Sistemas Baseados em IC

Os quatro grupos de sistemas aqui mencionados são inspirados na Biologia. As RNAs são baseadas no sistema neurológico humano, os modelos de CE, por sua vez são inspirados da Teoria da Evolução, a IE é baseada em organismos que vivem em colônias ou enxames, como o próprio nome sugere e os SD têm suas origens em como os organismos interagem com o ambiente em que vivem.

2.1 REDES NEURAIS ARTIFICIAIS

Segundo Engelbrecht (2007), o cérebro humano é um computador complexo, paralelo e não linear que tem a habilidade de aprender, memorizar e generalizar a partir de exemplos. É capaz de realizar tarefas como o reconhecimento de padrões, o controle motor e a percepção em muito menos tempo do que qualquer computador.

Para Silva, Spatti e Flauzino (2010), os cientistas possuem o sonho antigo de construir um mecanismo autônomo, que seja dotado de inteligência. Possivelmente a principal motivação para este anseio seja a complexidade do funcionamento do cérebro humano, sua capacidade de processamento, adaptação, entre outras habilidades.

As Redes Neurais Artificiais (RNAs) são, portanto, um grupo de modelos matemáticos/computacionais inspirados no funcionamento do sistema neurológico humano, como é possível ver

em Silva, Spatti e Flauzino (2010) e Engelbrecht (2007). Para Haykin (2001), uma Rede Neural (RN) se assemelha ao cérebro em dois aspectos: possui um processo de aprendizagem a partir do ambiente em que está inserida e o conhecimento adquirido fica armazenado nas conexões entre os neurônios.

Para Silva, Spatti e Flauzino (2010), Haykin (2001) e Fausett (1994), as principais características benéficas das RNAs são:

- capacidade de absorver a não linearidade nos dados de entrada;
- capacidade de mapeamento de entrada-saída;
- adaptabilidade à modificações do meio ambiente;
- resposta à evidências;
- contextualização natural da informação por meio da interação entre neurônios;
- tolerância a falhas, para o caso de implementação em *hardware*;
- possibilidade de implementação em larga escala;
- uniformidade de representação e;
- analogia neurológica.

A estrutura de um neurônio biológico está apresentada na Figura 2 e a de um neurônio artificial na Figura 3. A unidade básica de nosso cérebro apresenta uma região onde informações são processadas (corpo celular), algumas entradas (dendritos) e uma saída (axônio). Os impulsos elétricos recebidos nos dendritos são processados e o resultado deste processamento é colocado no axônio.

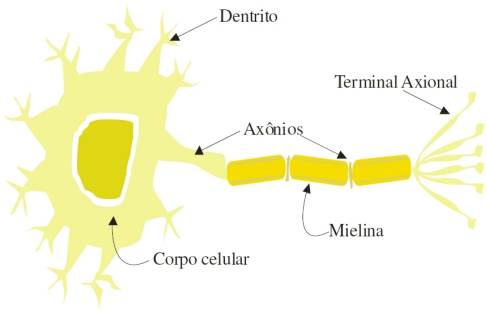


FIGURA 2: Representação do Neurônio Biológico

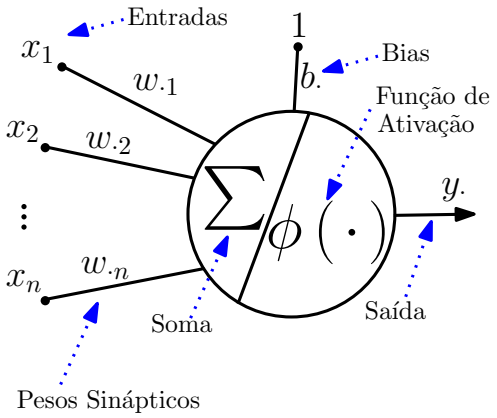


FIGURA 3: Representação do Neurônio Artificial

Na Figura 3, os valores denotados por x_i são informações provenientes do ambiente externo denominadas nós de entrada. Se compararmos com o neurônio biológico da Figura 2, a conexão entre os vetores x_i e os pesos $w_{i\cdot}$, $i = 1, \ldots, n$, é realizada pela sinapse que termina por conectar o ambiente externo ao neurônio, por meio de seus dentritos, os quais são responsáveis por receber os sinais de entrada dos neurônios. As sinapses são uma região de contato muito próximo entre os dentritos e outras células, inclusive outros neurônios.

Os dentritos conduzem os sinais de entrada para o corpo celular, que no neurônio artificial é representado pela soma e função de ativação, fazendo o papel do núcleo de processamento do neurônio. O valor representado por $y_{\cdot}$ é a saída (resultado), que acusa o nível de ativação de um neurônio artificial. No neurônio biológico, essa função é desempenhada pelo axônio. A ligação entre o terminal axional de um neurônio e os dentritos de outro neurônio é feita também pela sinapse.

No neurônio artificial as sinapses recebem multiplicadores (pesos) que podem ser inibitórios (negativos) ou excitatórios (positivos). No esquema representado na Figura 3, os pesos $w_{\cdot}$ e $b_{\cdot}$ representam esses pesos, sendo que os pesos $b_{\cdot}$ são adicionados à rede apenas para aumentar seus graus de liberdade, causando um relaxamento no modelo matemático, o valor de suas entradas é sempre “1”, por este ser o elemento neutro da multiplicação.

Com relação à aplicabilidade das RNAs, vale salientar que é possível modelar problemas de áreas como Ciências Exatas, Sociais, Humanas, entre outras áreas via Redes Neurais. As principais tarefas associadas a uma RNA são a classificação, o agrupamento de padrões e a previsão de Séries Temporais.

Entretanto, como é possível ver em Siqueira, Scheer e Steiner (2005), o roteamento de veículos e a designação também são problemas passíveis de resolução por uma abordagem via RNAs. Mais exemplos de aplicações podem ser encontrados em Silva, Spatti e Flauzino (2010), Engelbrecht (2007), Haykin (2001) e Fausett (1994).

Os cinco grupos de RNAs mais citados na literatura são:

- RNAs de camada única como, por exemplo, a Rede de Hopfield;
- de camada múltipla como o *Perceptron* de Múltiplas Camadas (MLP);
- redes de Elman e a de Jordan que trabalham com a ideia de recorrência;
- Mapas Auto-organizáveis, sendo a mais conhecida a rede de Kohonen e;
- redes alimentadas adiante combinadas com mapas auto-organizáveis como, por exemplo, as Redes Neurais de Base Radial (RBFNN).

As Redes Neurais de Base Radial ou *Radial Basis Function Neural Networks* (RBFNNs), que são o foco dessa pesquisa, suas características, métodos para definição de seus parâmetros e seus métodos tradicionais de treinamento estão descritos a seguir.

2.1.1 **Redes Neurais de Base Radial**

Uma Rede Neural de Base Radial (RBFNN) é do tipo alimentada adiante e possui apenas duas camadas, sendo uma a intermediária e a última a camada de saída. Na camada intermediária as funções de ativação dos neurônios são ditas *funções de base radial*. A seguir, serão apresentadas as principais características das RBFNNs, mas antes vejamos alguns conceitos preliminares.

Funções de Base Radial e a Interpolação Exata

Uma RBF - *Radial Base Function* (Função de Base Radial) é definida por Haykin (2001) como qualquer função que satisfaça a Equação 2.1. Dizemos que uma função é de base

radial quando seus valores funcionais são iguais aos valores funcionais das normas de seus argumentos.

$$f(x) = f(\|x\|) \tag{2.1}$$

Em outras palavras, uma função é de base radial quando seu valor funcional depende apenas da distância de seu argumento à origem. Algumas dessas funções, comumente utilizadas, são apresentadas nas equações 2.2–2.4.

$$f(x) = e^{-\beta x^2}, \text{para } \beta > 0 \text{ (gaussiana)} \tag{2.2}$$

$$f(x) = \sqrt{x^2 + \beta^2}, \text{para } \beta > 0 \text{ (multiquadrática)} \tag{2.3}$$

$$f(x) = x^k, \text{para } k = 1, 3, \ldots \text{ (spline poliarmônica)} \tag{2.4}$$

De acordo com Powell (1988), a abordagem que busca aproximar funções por meio de combinações lineares de funções de base radial consiste em introduzir um conjunto com $n = 1, \cdots, N$ funções base centradas num ponto x^n, uma para cada observação amostral, tomando a forma $\phi(\|x - x^n\|)$ em que $\phi(\ .\ )$ é uma RBF. Então, cada função base depende apenas da distância $(\|x - x^n\|)$ e a aproximação toma a forma definida pela Equação 2.5,

$$h(x^n) = \sum_{n=1}^{N} w_n \cdot \phi(\|x - x^n\|), \tag{2.5}$$

em que w_n são os pesos da combinação linear. Considerando esta abordagem para resolver o problema de interpolação exata na Equação 2.6, em que $t(x^n)$ são os valores alvo e $h(x^n)$ são os resultados do modelo de interpolação temos,

$$h(x^n) = t(x^n). \tag{2.6}$$

Combinando as Equações 2.5 e 2.6, temos

$$\sum_{n=1}^{N} w_n \cdot \phi(\|x - x^n\|) = t, \tag{2.7}$$

que podemos representar matricialmente por,

$$\phi \cdot w = t, \tag{2.8}$$

em que $t = (t^1, t^2, \ldots, t^n)$, $w = (w_1, w_2, \ldots, w_n)$ e ϕ é uma matriz quadrada. A solução formal para 2.8 é dada na Equação 2.9 e para funções multiquadráticas, multiquadráticas inversas e gaussianas, respectivamente, a matriz ϕ é invertível.

$$w = \phi^{-1} \cdot t \tag{2.9}$$

Redes Neurais de Base Radial e sua Arquitetura

O processo de aprendizagem desta rede tem suas bases na teoria da Programação Não Linear. A arquitetura de uma RBFNN é composta por apenas duas camadas além dos nós de entrada. Tem-se uma camada escondida, que possui funções de base radial como funções de ativação, e uma camada de saída, que possui funções lineares como ativação. A Figura 4 ilustra este tipo de rede.

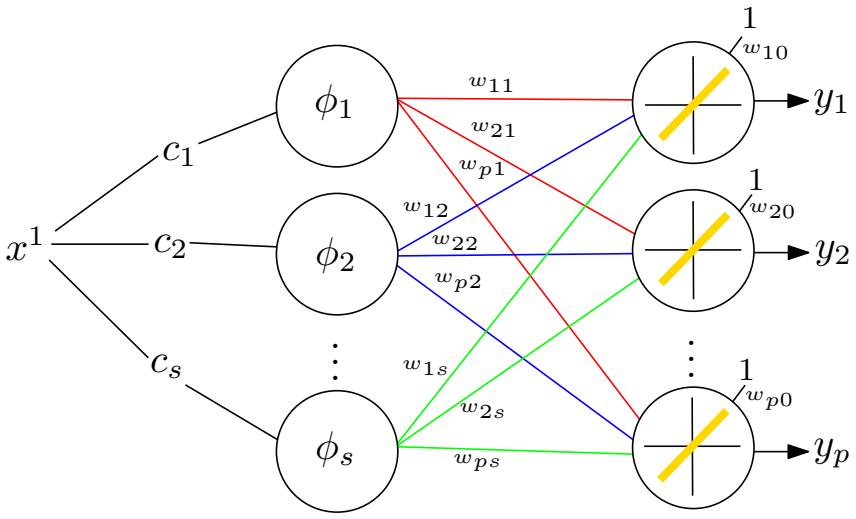


FIGURA 4: Representação da arquitetura de uma RBFNN

Cada neurônio da camada escondida possui um vetor associado, chamado de centro do neurônio, o qual define o centro do campo receptivo daquele neurônio. Geralmente tais vetores são armazenados numa matriz C, chamada de matriz de centros dos neurônios. Estes vetores exercem alguma influência sobre o desempenho da rede. A seguir, alguns métodos serão brevemente apresentados.

Parâmetros de uma RBFNN

Nesta seção serão apresentadas algumas das estratégias mais utilizadas para a determinação dos parâmetros de uma RBFNN que são, essencialmente três: os Centros do campo receptivo dos neurônios, o grau de abertura do raio da Função de Base Radial e a Matriz de Pesos Sinápticos entre a camada intermediária e a de saída.

Seleção dos Centros dos Neurônios

A tarefa de selecionar os centros (ou vetores centróides) dos neurônios é, essencialmente, um problema de agrupamento de padrões. Há algumas estratégias clássicas descritas em Haykin (2001) e uma breve discussão sobre estas estratégias é apresentada a seguir.

Centros Fixados Aleatoriamente

Esta é a maneira mais simples e menos custosa computacionalmente de selecionar os centros. Embora seja simples, é considerada por Haykin (2001) como a abordagem “mais sensata”, pois a cada experimento uma parte diferente da matriz de observações é utilizada e, por este motivo, a matriz de centros passa a conter uma boa representação do espaço de entrada. O método consiste em escolher uma das seguintes opções:

(a) tomar valores aleatórios de um intervalo dado, sendo que a escolha deste intervalo,

certamente, influencia o desempenho do algoritmo;

(b) escolher aleatoriamente, dentre os valores observados, os centros, podendo obter boas representações do conjunto de treinamento no que diz respeito aos pontos “medianos”, dependendo das características do problema e, consequentemente, da distribuição dos dados.

A limitação deste método reside na necessidade de um conjunto de treinamento grande para que um desempenho satisfatório seja alcançado. O padrão mais usado é aquele que utiliza 60% para treinamento, 20% para validação e 20% para teste. Uma mudança nesse padrão deve ser feita de maneira cuidadosa, pois o modelo pode perder capacidade de generalização.

Seleção Auto-organizada de Centros

Descrito em Haykin (2001) e também em Silva, Spatti e Flauzino (2010), este método de seleção de centros consiste em aplicar a ideia básica do algoritmo SOM - *Self Organizing Map* (Mapa Auto-organizável), proposto por Kohonen (1998) e descrito na Figura 5.

1. Gerar valores randômicos (distintos), ou escolher aleatoriamente, dentre os valores observados, os centros iniciais;
2. Na *n*-ésima iteração, tomar uma amostra do conjunto de treinamento;
3. Para cada vetor de entrada, encontrar o centro que possui menor distância euclidiana ou, equivalentemente, o centro com o qual o vetor de entrada produz o maior produto interno;
4. Para o centro vencedor, atualizar sua localização segundo a regra:
$$c(\text{novo}) = c(\text{anterior}) + \eta(x - c)$$
em que $\eta \in (0,1)$ é uma taxa de aprendizagem, x é o vetor de entrada e c é o centro vencedor;
5. Voltar ao Passo 2 até que não haja modificações significativas na matriz de centros.

FIGURA 5: Seleção auto-organizada de centros

A diferença deste algoritmo para o algoritmo SOM clássico é justamente a ausência de um “mapa” de neurônios, ou seja, nenhuma vizinhança é sequer considerada na atualização dos neurônios. Isso equivale a dizer que é um algoritmo SOM com o princípio *Winner Takes All*

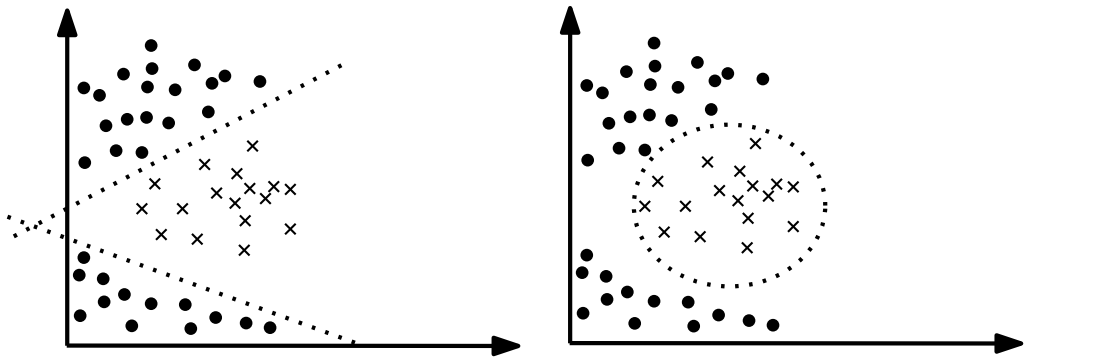
(O Vencedor Leva Tudo) pois somente o neurônio vencedor é atualizado. Uma descrição do princípio *Winner Takes All*, aplicado ao problema de designação usando uma rede recorrente, pode ser encontrado em Siqueira, Scheer e Steiner (2005).

Seleção Supervisionada de Centros

Esta é a forma mais genérica de se realizar a seleção de centros. A ideia é ajustar os neurônios na matriz de centros considerando aprendizagem por correção de erro e, desse modo, a RBFNN se assemelha ao *Perceptron* clássico. Um candidato natural para resolver este problema é o algoritmo conhecido como descida do gradiente, que é utilizado para minimizar uma função de custo/erro.

Grau de de Abertura do Raio do Campo Receptivo

O raio de abertura do campo receptivo de uma RBF é o parâmetro que define a capacidade que a função tem de agrupar padrões. Vale ressaltar que ao contrário do *Perceptron*, o qual separa os padrões por meio da construção de hiperplanos, uma RBFNN classifica os padrões por meio de regiões hiperelipsoidais.



(a) *Perceptron* Multicamadas (b) Rede Neural de Base Radial

FIGURA 6: Separação de padrões do *Perceptron* e RBFNN

É possível observar na Figura 6 que esta estratégia de agrupamento pode ser mais eficiente do que a construção dos hiperplanos. Segundo Haykin (2001), o campo receptivo de uma

RBF é dado por uma matriz de covariância Σ e é possível identificar três diferentes cenários de sua influência sobre a forma, tamanho e orientação do campo receptivo:

1. $\Sigma = \sigma^2 \cdot \mathrm{I}$, onde I é a matriz identidade e σ^2 é uma variância comum. Neste caso, o campo receptivo é uma hiperesfera;

2. $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2)$, quando forem considerados p exemplos para o treinamento. Neste caso, o campo receptivo consiste numa hiper-elipse com a extensão dos j-ésimos eixos determinados por σ_j, com $j = 1, \ldots, p$;

3. Σ é uma matriz não-diagonal. Nesse caso, como Σ é uma matriz definida positiva, por ser uma matriz de covariâncias, pode ser decomposta em:

$$\Sigma = Q^T \Lambda Q$$

   em que Λ é uma matriz diagonal e Q é uma matriz ortonormal de rotação. Neste caso, a matriz Λ determina a forma e o tamanho do campo receptivo enquanto Q determina sua orientação.

Vale salientar que o tamanho do raio de abertura influencia diretamente no tamanho do campo receptivo de um neurônio. O ideal é que o raio de abertura não cause sobreposição nem disjunção excessiva dos campos receptivos, pois em ambos os casos a rede pode ter dificuldades em classificar padrões que estão perto dos limites dos campos receptivos.

Segundo Braga e Ludermir (2007), várias heurísticas vem sendo propostas para a definição do raio de abertura das RBFNNs. Uma delas define cada σ_j como sendo a média das distâncias euclidianas entre cada centro e seu vizinho (centro) mais próximo, ou seja, se considerarmos p neurônios na camada intermediária, consequentemente com p centros, sendo d a distância euclidiana, os valores de σ_j são dados pela Equação 2.10.

$$\sigma_j = \min(d_{i \neq j}(c_i, c_j)) \text{ para } i = 1, \ldots, p \text{ e } j = 1, \ldots, p. \tag{2.10}$$

Outra heurística, encontrada em Silva, Spatti e Flauzino (2010) e Braga e Ludermir (2007), considerando n exemplos de treinamento e p neurônios na camada intermediária, atribui a

cada σ_j a média das distâncias dos $N \leq n$ vetores de entrada mais próximos ao centro c_j do que aos outros centros. A equação 2.11 representa esta heurística.

$$\sigma_j = \frac{1}{N} \sum_{i=1}^{N} d(x_i, c_j) \text{ para } j = 1, \ldots, p, . \tag{2.11}$$

em que x_i é um dos vetores de entrada mais próximos de c_j e d é a distância euclidiana. Esta última heurística mostrou um desempenho adequado em Mota et al. (2012), no sentido de proporcionar percentuais de acerto na classificação acima de 80%.

Cálculo da Matriz de Pesos

Seja $X \subset \mathbb{R}^{m \times n}$ a matriz de observações contendo n observações e m variáveis:

$$X = \begin{pmatrix} x_{11} & \ldots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \ldots & x_{mn} \end{pmatrix}$$

O valor de ativação do s-ésimo neurônio da camada escondida depende da distância euclidiana quadrática entre a entrada x^i e o centro c_j, onde $c_j \in C$ é o centro do m-ésimo neurônio. Então, de acordo com a abordagem clássica, treinar uma RBFNN é, de fato, calcular a matriz de pesos,

$$w = \begin{pmatrix} w_{10} & w_{11} & \ldots & w_{1s} \\ w_{20} & w_{21} & \ldots & w_{2s} \\ \vdots & \ddots & \vdots & \vdots \\ w_{p0} & w_{p1} & \ldots & w_{ps} \end{pmatrix}$$

de maneira a ajustar aos alvos t cada y, como podemos ver na Equação 2.12,

$$y_r = w_{0r}\phi_{0r} + \sum_{k=1}^{s} w_{kr}\phi\left(||x^i - C||\right). \tag{2.12}$$

Resolver o problema da Equação 2.12 é equivalente a resolver a Equação 2.8. A única

diferença é a forma de ϕ, que em 2.12 é retangular. Ou seja, treinar uma RBFNN é equivalente a resolver um sistema não linear retangular contendo funções de base radial.

Entretanto, existem, pelo menos, mais duas abordagens: uma baseada em correção de erro, que pode ser encontrada em Silva, Spatti e Flauzino (2010) e a outra, na qual esta pesquisa está focada, que é a utilização de Meta-heurísticas de busca para o cálculo dos pesos, produzindo um sistema híbrido, como por exemplo em Mota et al. (2012). A criação de Metodologias Híbridas com RBFNNs e Meta-heurísticas de busca baseadas em população está descrita no Capítulo 3.

2.1.2 O Projeto de uma RBFNN

Uma vez que sua arquitetura é pré-definida, sendo uma rede alimentada adiante, projetar uma RBFNN consiste em definir os seguintes aspectos:

- **Separação e Tratamento dos Dados**: separar os exemplos que serão utilizados no treinamento, validação (caso haja) e testes da RBFNN, bem como realizar um tratamento estatístico de dados visando identificar padrões que possam atrapalhar o desempenho da rede por serem muito atípicos, quando comparados com o restante dos padrões;
- **Topologia**: consiste em definir a quantidade de neurônios na camada intermediária e da camada de saída;
- **Função de Ativação da Camada Intermediária**: novamente, como a função de ativação da camada de saída também é pré-definida, como sendo a linear pura, existe apenas a necessidade de escolher a função de ativação de cada neurônio na camada escondida;
- **Seleção dos Centros**: escolher o método de seleção dos centros que melhor defina suas localizações;
- **Raio de Abertura**: escolher o método para definir a amplitude do campo receptivo de cada neurônio na camada intermediária e;
- **Cálculo da Matriz de Pesos**: escolher o método para o cálculo da matriz de pesos entre a camada intermediária e a camada de saída;

Separação e Tratamento dos Dados

Há dois processos de separação dos exemplos para treinamento, teste e validação conhecidos como *hold-out* e validação cruzada passíveis de aplicação às RBFNNs. Ambos os procedimentos utilizam-se de sorteios aleatórios para definir quais exemplos pertencem a cada grupo, sendo que seus tamanhos são pré-definidos.

Hold-out

O *hold-out*, em problemas de classificação, consiste em dividir os exemplos amostrais em três grupos: treinamento, validação e testes. Na maioria das pesquisas os percentuais encontrados para cada conjunto são 60–20–20, respectivamente, sendo o padrão 70–15–15 uma segunda opção adequada. Nesta abordagem, a Rede Neural em questão é treinada enquanto o percentual de acerto no conjunto de validação estiver aumentando, uma vez que este é utilizado como auxiliar na medida de capacidade de generalização da rede.

A segunda abordagem do *hold-out*, aplicável em problemas de previsão em Séries Temporais (ST), consiste em dividir os exemplos amostrais em dois grupos: treinamento e testes. No caso das ST, os exemplos são ordenados de acordo com sua disposição no tempo, ou seja, a escolha aleatória de quais exemplos pertencerão a cada grupo não se aplica a esse caso. Os percentuais mais encontrados na literatura para treinamento e teste são 90–10, respectivamente.

Validação Cruzada

A Validação Cruzada consiste em dividir os exemplos amostrais em k grupos de mesmo tamanho (ou aproximadamente iguais), sorteando aleatoriamente exemplos para integrar os grupos, e para executar os seguintes procedimentos:

1. Sortear um dos k grupos de mesmo tamanho (ainda não sorteado) e definí-lo como conjunto de testes;
2. treinar a rede com os $k-1$ grupos restantes;
3. verificar a capacidade de generalização da rede no conjunto de testes;
4. Voltar ao passo um até que todos os grupos tenham figurado como conjunto de testes.

Nesse procedimento, o percentual de acerto da rede que mede sua capacidade de generalização é a média aritmética simples dos percentuais de acerto em todos os conjuntos de testes.

Função de Ativação

A escolha da função de ativação na camada escondida de uma RBFNN não possui um método específico. De forma empírica, faz-se necessária a realização de testes iniciais para verificar qual função apresenta melhor ajuste aos dados. A função mais utilizada na literatura é a Gaussiana, dada na Equação 2.2. Vale ressaltar que os neurônios da camada escondida podem utilizar funções de ativação distintas.

Seleção dos Centros

A seleção dos centros (ou centróides) consiste em definir quais vetores serão os centros do campo receptivo de cada neurônio na camada escondida. Em geral, qualquer algoritmo de agrupamento executará bem esta tarefa, uma vez que a seleção de centróides é um problema clássico de k-medianas.

Raio de Abertura

O raio de abertura da função de ativação delimita a amplitude do campo receptivo do neurônio em questão. Novamente não há apenas um método para a realização dessa tarefa nem relação conhecida *a priori* entre a distribuição dos dados e a escolha do método, podendo ser um dos descritos anteriormente.

É necessário que sejam realizados alguns testes preliminares para verificar qual método mostra tendência em produzir melhores resultados. Pode haver melhores resultados com métodos diferentes, em conjuntos de dados de uma mesma área de estudo.

Matriz de Pesos

Na abordagem clássica das RBFNNs, o cálculo da matriz de pesos, que é o treinamento da rede em si, consiste em pseudo-inverter a matriz com os valores de ativação dos neurônios da camada intermediária.

Estratégias para a Definição da Topologia

Conforme mencionado anteriormente, o treinamento de uma RBFNN consiste na pseudo-inversão da matriz com os valores de ativação dos neurônios da camada escondida e a definição de sua topologia utiliza a mesma ideia principal do treinamento de uma Rede Neural.

Na maioria dos trabalhos encontrados na literatura o procedimento adotado é o seguinte: executa-se um processo iterativo, aumentando a quantidade de neurônios enquanto o desempenho do conjunto de validação estiver melhorando. O “treinamento” deve ser abortado quando houver uma determinada quantidade de decréscimos no desempenho do conjunto de validação. Esses decréscimos são chamados de “falhas de validação”. Adota-se, então, como capacidade de generalização da rede, o percentual de acertos no conjunto de testes correspondente à iteração em que foi encontrado o melhor resultado no conjunto de validação.

2.2 COMPUTAÇÃO EVOLUCIONÁRIA

A princípio, a Matemática e a Biologia são duas Ciências com sistematizações e abordagens diferentes com um único objetivo em comum, aparentemente, de fazer parte de uma gama de conhecimentos necessários à compreensão do Universo. Entretanto, o surgimento de problemas mais específicos e complexos fez surgir uma parceria sistematizada incialmente por Holland (1975) entre a Teoria da Evolução, proposta por Charles Darwin e a Álgebra Linear, posteriormente a Informática veio fazer parte deste grupo, fornecendo ferramentas para a implementação destes modelos.

A Computação Evolucionária (CE) tem como princípio básico modelar a evolução natural. Para que isso seja possível, segundo Rutkowski (2008) e Engelbrecht (2007), é necessário levar em consideração os principais conceitos de sobrevivência: adaptação e aptidão. Para isso, os indivíduos melhores adaptados ao meio sobrevivem e têm chance de passar os códigos genéticos que garantiram sua sobrevivência aos seus descendentes e os menos adaptados têm maior probabilidade de morrer e perder esta chance, como afirmam Holland (1975) e Goldberg (1989).

Essencialmente, a CE consiste em modelar uma população de indivíduos e fazer com que esta população se reproduza por meio de cruzamentos, gerando descendência, e evolua por meio de mutação, sendo levados em consideração os princípios de seleção natural. Existem diferentes classes de algoritmos já desenvolvidos, em Engelbrecht (2007) estão listados os principais deles:

- **Algoritmos Genéticos (AG)**: Têm como principal objetivo modelar a evolução genética;
- **Programação Genética**: São baseados em AG, mas os indivíduos são programas computacionais, em vez de vetores, e tais programas são representados em árvores;
- **Programação Evolucionária**: é uma derivação da simulação de comportamento adaptativo na evolução, também chamado de Evolução Fenotípica;
- **Estratégias Evolucionárias**: orientam-se pela modelagem dos parâmetros que controlam a variação na evolução;
- **Evolução Diferencial**: difere dos AGs apenas no mecanismo utilizado para a reprodução;

- **Evolução Cultural**: modela a evolução da cultura de uma população e como como esta cultura influencia a evolução genética e fenotípica dos indivíduos;
- **Co-evolução**: nesta abordagem, inicialmente são gerados indivíduos “burros” que cooperam ou competem entre si, com o objetivo de adquirir as características necessárias para sobreviver.

Segundo Engelbrecht (2007), a CE tem sido utilizada com sucesso em vários problemas do mundo real, tais como: Mineração de Dados, Otimização Combinatória, Diagnóstico de Falhas, Classificação e Agrupamento de Padrões, Quadro de Horários e Previsão de Séries Temporais.

2.2.1 Algoritmos Genéticos

Os Algoritmos Genéticos (AG) fazem parte de uma subárea da Inteligência Computacional (IC) conhecida como Computação Evolucionária (CE), juntamente com outras técnicas, tais como: a Programação Genética (PG), Programação Evolucionária e Evolução Diferencial. Nesta seção os conceitos principais sobre os algoritmos genéticos são apresentados, bem como as principais características e operadores dos AG contínuos, que fazem parte do foco deste trabalho.

Princípios Fundamentais

Os Algoritmos Genéticos (AG) conforme descrito em Goldberg (1989) e Holland (1975) são, basicamente, um conjunto de algoritmos baseados nos princípios de biologia evolutiva enunciados por Charles Darwin em seu livro intitulado *A Evolução das Espécies*. Em sua abordagem clássica, os AGs utilizam strings vetoriais como candidatos à solução de um problema de otimização.

Os princípios considerados na elaboração de um AG são a seleção natural, hereditariedade, mutação e a recombinação (*crossover*). Quando são traduzidos para a linguagem matemática, tais princípios se transformam em operadores genéticos de cruzamento e mutação,

já a seleção natural se transforma numa regra de decisão. A Figura 7 apresenta o algoritmo genético básico, em sua forma genérica e resume seu funcionamento.

Seja $g = 0$ o contador de gerações;
Criar e inicializar uma população de tamanho pré-definido;
    Enquanto *nenhum critério de parada for satisfeito* faça;
        Avalie o *fitness* (aptidão) de cada indivíduo da população;
        Selecione os indivíduos;
        Realize a reprodução para gerar descendência;
        Selecione a nova população;
        Avance para a nova geração, ou seja, $g = g + 1$;
    Fim Enquanto

FIGURA 7: Algoritmo Genético Original

Vale ressaltar que, inicialmente, a base numérica binária (representação binária) foi considerada para implementação e aplicação dos Algoritmos Genéticos na literatura. Contudo, hoje é bastante comum encontrarmos aplicações em que a base numérica decimal é utilizada, podendo utilizar valores inteiros e/ou reais (Algoritmos Genéticos Contínuos).

A escolha da base decimal em vez da base binária se justifica pela maior proximidade entre soluções ligeiramente diferentes, uma vez que, na base binária, uma pequena variação no vetor de resposta pode ocasionar uma mudança radical na direção de busca, podendo ocasionar perdas maiores na função objetivo do que na base decimal.

Os componentes da configuração de um AG são:

1. uma função de aptidão;
2. uma população inicial de tamanho predefinido;
3. um operador de seleção de pares para a reprodução;
4. os operadores de reprodução;
5. critérios de medida de convergência, ou critérios de parada.

Função de Aptidão (Fitness)

Segundo Goldberg (1989) e Engelbrecht (2007), a Função de *Fitness* (FF) (aptidão) é, possivelmente, o componente mais importante de um Algoritmo Evolucionário (AE), consequentemente, de um AG. Uma FF tem como principal tarefa mapear a representação de um cromossomo (indivíduo), a um valor escalar, geralmente real, ou seja:

$$\mathbb{F} : \mathbf{C}^I \rightarrow \mathbb{R} \tag{2.13}$$

onde $\mathbb{F}$ é a FF e $\mathbf{C}$ representa o cromossomo I-dimensional.

Considerando o fato de que cada indivíduo representa uma potencial solução ótima, de acordo com critérios pré-definidos, para o problema, uma FF tem, como consequência de sua função primordial, quantificar o quão bom um indivíduo é, o quão perto a solução está do ótimo. A maioria dos operadores genéticos fazem uso da FF como critérios internos de operação.

Uma FF deve ser um bom modelo matemático do problema, isto é, deve refletir todos os critérios para ser otimizada e, inclusive, deve levar em consideração as restrições do problema, conforme afirmam Goldberg (1989) e Engelbrecht (2007). A inclusão das restrições numa FF pode ser feita por meio de penalizações. Vale observar que as restrições podem ser incorporadas em outras etapas do AG, como nos operadores de cruzamento e mutação.

População Inicial

Uma população inicial, de tamanho pré-definido, equivale a gerar soluções para o problema em questão. Geralmente, esta etapa consiste em gerar aleatoriamente vetores (indivíduos) que estejam no espaço de soluções. Para garantir a diversidade genética e representatividade do espaço de busca, a *priori*, a única restrição nesta operação é a de que os indivíduos sejam distintos.

Segundo Engelbrecht (2007), se houver informações disponíveis sobre quais características são desejáveis nos elementos da população inicial é possível criar heurísticas que levem isso

em consideração. Entretanto, é necessário ponderar que esta estratégia pode levar à convergência prematura e a um ótimo local, uma vez que indivíduos que poderiam contribuir para melhoria da população, por diversidade genética, podem não ser selecionados para a população inicial.

Com relação ao tamanho da população inicial, não existe um consenso do que pode ser considerada uma população inicial “grande” ou “pequena”. Entretanto, quase um uníssono na literatura é a afirmação de que uma população inicial pequena, embora diminua o tempo computacional de execução do algoritmo, pode requerer mais gerações para convergir e levar a um ótimo local, enquanto que uma população inicial grande requer menos iterações para convergir e têm mais chances de chegar ao ótimo global.

É claro que a falta de diversidade em populações iniciais pequenas pode ser amenizada aumentando-se a taxa de mutação, mas não há garantias de que esta estratégia resolva seu problema de miopia, ou seja, o tropeço em ótimos locais.

Operadores de Seleção

Segundo Holland (1975) e Engelbrecht (2007), os Operadores de Seleção (OS) para a reprodução têm a função de garantir que os melhores indivíduos de uma população sejam destacados. Alguns dos OS a serem considerados são: Seleção Randômica, Seleção Proporcional, Seleção via Torneio, Seleção Baseada em Ranqueamento e Elitismo, os quais estão descritos a seguir:

- **Seleção Randômica**: na qual os indivíduos são selecionados sem critério algum. Indivíduos bons e ruins têm a mesma chance de passar seu código genético às gerações futuras;
- **Seleção Proporcional**: a chance de um indivíduo ser selecionado para se reproduzir é proporcional a seu *fitness* e expresso em probabilidade, de acordo com a Equação 2.14,

$$\text{Prob}(\mathbf{C}) = \frac{\mathbf{F}(\mathbf{C})}{\sum_{n=1}^{N} \mathbf{F}(\mathbf{C})} \tag{2.14}$$

em que Prob($\mathbf{C}$) representa a probabilidade de um indivíduo ser selecionado e $\mathbf{F}(\mathbf{C})$ é o *fitness* de cada um dos N indivíduos da população;

- **Seleção via Torneio**: Nesta modalidade, $k < N$ indivíduos são pré-selecionados aleatoriamente e apenas o indivíduo que possui o melhor *fitness* é escolhido para se reproduzir;
- **Seleção Baseada em Ranqueamento**: Este tipo de seleção usa a ordenação em *ranking* do *fitness* para determinar a probabilidade de seleção de um indivíduo. Isso quer dizer que a seleção de um indivíduo, por este método de seleção, independe do valor real do *fitness*. Essa estratégia garante que nenhum indivíduo da população, não importando a magnitude de sua aptidão, dominará o processo no sentido de sempre ser escolhido para gerar descendentes.
- **Elitismo**: O Elitismo é um operador de seleção que se preocupa em definir quais indivíduos sobreviverão para a próxima geração (sem sofrer mutação), selecionando assim os possíveis indivíduos que poderão, eventualmente, gerar descendência duas gerações à frente da atual. A seleção pode ser baseada nos melhores *fitness* ou qualquer um dos critérios mencionados anteriormente.

Operadores de Reprodução

Existem, essencialmente dois operadores de reprodução a serem considerados: *crossing-over* ou *crossover* (cruzamento) e mutação. Aqui um indivíduo é considerado melhor adaptado ou "bom" se o valor de seu *fitness* for adequado. Por outro lado é considerado "ruim" se o valor de seu *fitness* for inadequado.

O objetivo principal dos operadores de reprodução é gerar descendentes, de indivíduos previamente selecionados, utilizando-se os operadores genéticos citados anteriormente.

O objetivo dos operadores de cruzamento é a combinação de materiais genéticos já existentes a fim de obter novos indivíduos que sejam compostos pelas melhores características de seus geradores, formando assim um indivíduo bem adaptado, isto é, com um valor adequado de *fitness*. Segundo Engelbrecht (2007), pode-se escolher a substituição dos geradores (pais)

por sua descendência (filhos), caso os *fitness* dos pais sejam menos adequados do que dos filhos.

A mutação é o processo de modificar randomicamente os valores dos genes nos indivíduos. O principal objetivo desse grupo de operadores genéticos é o de introduzir material genético novo na população, com o intuito específico de aumentar o espaço de busca por soluções. Segundo Holland (1975) e Engelbrecht (2007) a mutação de uma população geralmente ocorre a uma baixa probabilidade (0,01–0,1) para tentar evitar que indivíduos melhores adaptados percam as características que os tornam bons.

Critérios de Parada

Diversos critérios de parada para um algoritmo com essas características podem ser aplicados:

- quando o *fitness* do melhor indivíduo alcançar determinado valor;
- quando houver uma certa quantidade de indivíduos iguais (ou muito parecidos) na população;
- quando atingir uma certa quantidade de gerações (iterações);
- quando o melhor indivíduo não for superado após uma certa quantidade de gerações.

Entre algumas aplicações de AG, pode-se citar o trabalho de Wang e Kong (2011) que utilizaram como critério de parada o evento do melhor indivíduo não ser superado após uma quantidade de gerações. Mais especificamente, o critério de parada escolhido foi o de parar o algoritmo após o melhor indivíduo não ser superado mais de $0,1\%$ em 5000 gerações. Também, Hoffmann, Medina e Wolisz (2011) utilizaram o critério da quantidade de gerações, assumindo que o algoritmo convergiu após 5000 gerações.

Algoritmos Genéticos Contínuos e Operadores de Michalewicz

Conforme mencionado anteriormente, já existem inúmeras abordagens via AG que utilizam valores contínuos, os denominados Algoritmos Genéticos Contínuos. Para maiores esclarecimentos sobre os operadores clássicos desenvolvidos para AG binário/discreto, é possível consultar Holland (1975) e Goldberg (1989).

Uma das pesquisas que utiliza a representação real foi realizada por Michalewicz, Logan e Swaminathan (1994) que sistematizou três operadores de cruzamento (*Crossover*) e quatro operadores de mutação, descritos a seguir.

Crossover Simples

Dados dois indivíduos, o *crossover* simples consiste em sortear aleatoriamente um ponto de cruzamento e, a partir daquele ponto, ocorre a troca de genes entre os pais. A Figura 8 ilustra a ideia do ponto de cruzamento e a mistura genética que acontece entre os pais para gerar os filhos.

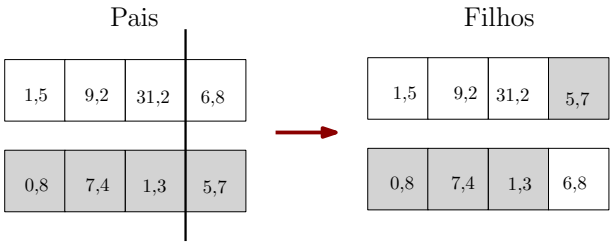


FIGURA 8: Cruzamento genético de um ponto

Crossover Aritmético

Dados dois indivíduos p_1 e p_2 para o cruzamento, o *crossover* aritmético consiste em gerar os elementos,

$$c_1 = \beta p_1 + (1-\beta)p_2 \text{ e } c_2 = \beta p_2 + (1-\beta)p_1 \text{ com } \beta \sim U(0,1). \tag{2.15}$$

Vale ressaltar que este operador não ultrapassa os intervalos (p_1, p_2).

Crossover Heurístico

Dados dois indivíduos p_1 e p_2 para o cruzamento, tal que a aptidão de p_1 é menos adequada do que a de p_2, este operador é a extrapolação linear entre os geradores,

$$c = p_1 + \beta(p_2 - p_1) \text{ onde } \beta \sim U(0,1). \tag{2.16}$$

Mutação Uniforme

Dado um indivíduo p, com $S = \{1, \ldots, s\}$ genes, este operador sorteia um gene $j \in S$ e o substitui por um número aleatório oriundo de uma distribuição uniforme, ou seja,

$$c_i = \begin{cases} U(a_i, b_i), & \text{se } i = j \\ p_i, & \text{caso contrário.} \end{cases} \tag{2.17}$$

Os valores a_i e b_i representam os limites do intervalo permitido para o indivíduo p em sua i-ésima componente, caso haja alguma restrição de factibilidade ou simplesmente de intervalo de busca.

Mutação de Limite

Dado um indivíduo p, com $S = \{1, \ldots, s\}$ genes, este operador substitui um gene $j \in S$ por um dos limites do intervalo factível $[a_i, b_i]$, caso haja alguma restrição de factibilidade ou apenas

intervalo de busca, onde $r \sim U(0,1)$,

$$c_i = \begin{cases} a_i, & \text{se } r < 0,5 \text{ e } i = j \\ b_i, & \text{se } r \geqslant 0,5 \text{ e } i = j \\ p_i, & \text{caso contrário .} \end{cases}$$

Mutação Não-uniforme Simples e Múltipla

Dado um indivíduo p, com $S = \{1, \ldots, s\}$ genes, este operador substitui um gene $j \in S$ por um número extraído de uma distribuição não-uniforme,

$$p_i = \begin{cases} p_i + (b_i - p_i) f(G), & \text{se } r_1 < 0,5 \text{ e } i = j \\ p_i - (p_i - a_i) f(G), & \text{se } r_1 \geq 0,5 \text{ e } i = j \\ p_i, & \text{caso contrário .} \end{cases}$$

com,

$$f(G) = \left[r_2 \left(1 - \frac{G}{G_{\max}} \right) \right]^b,$$

sendo G a geração atual, $G_{\max}$ o número máximo de gerações e $r_1, r_2 \sim U(0,1)$. Além disso, os valores a_i e b_i são os limites do intervalo factível. A aplicação da mutação não-uniforme em todos os genes deste indivíduo é chamada de mutação não-uniforme múltipla.

2.3 INTELIGÊNCIA DO ENXAME

A Inteligência do Enxame (IE) é uma área de estudo que tem suas origens nas colônias, também chamados de enxames, de seres vivos. São vários os seres vivos que se beneficiam da vida em sociedade, alguns destes inspiraram algoritmos de otimização. Os exemplos mais encontrados na literatura são: pássaros, abelhas e formigas.

Segundo Abraham, Guo e Liu (2006), a técnica conhecida como *Particle Swarm Optimization* (PSO), traduzida para o português como “Nuvem de Partículas”, tem sua inspiração no estudo do comportamento de pequenos grupos (*blocks*) de pássaros e é um método de otimização global.

A PSO é um método de busca baseado em população no qual os indivíduos (partículas) são agrupadas num enxame (nuvem). Cada partícula representa um candidato à solução do problema, em geral um vetor. Num sistema baseado em PSO, cada partícula se movimenta no espaço de busca, atualiza sua posição, de acordo com sua própria experiência e, também, levando em consideração a experiência do grupo.

Assim como em outros métodos, a medida de otimalidade de uma solução é baseada numa função de aptidão (*fitness*) do indivíduo, podendo variar sua medida desejável de acordo com o significado de tal função para o problema em questão. Ainda segundo Engelbrecht (2007), as aplicações da PSO incluem agrupamento, como em Marini e Walczak (2011) e Lin e Tzeng (2010), otimização de estruturas mecânicas, ver Omkar et al. (2009) e Li et al. (2007) e, consequentemente, a solução de sistemas lineares.

Outra colônia que encontramos na natureza é a de formigas. A maneira como esses insetos encontram o menor caminho até uma fonte de comida por meio do depósito de feromônios, como uma trilha deixada aos indivíduos da colônia, inspirou o algoritmo de otimização conhecido como *Ant Colony Optimization* (ACO) ou “Colônia de Formigas”.

Segundo Engelbrecht (2007), a aplicação primordial do algoritmo ACO é a determinação da rota mais curta, entretanto, há outras aplicações tais como: roteirização de veículos, Lee et al. (2010) e Yu, Yang e Yao (2009), problemas de quadro horário e coloração de imagens, a classificação de padrões, como podemos ver em Villwock, Steiner e Siqueira (2011) e aplicação em mineração de dados oriundos da Internet como é possível observar em Abraham, Guo e Liu (2006).

Em suma, a Inteligência do Enxame é um campo de estudo ainda jovem e emergente, como ponderam Bonabeau, Corne e Poli (2010). Ademais, segundo Engelbrecht (2007), há muito potencial nesta área da IA e as aplicações iniciais, à sua época de publicação, se mostraram promissoras.

2.3.1 **Nuvem de Partículas**

O algoritmo PSO - *Particle Swarm Optimization*, ou “Nuvem de Partículas” é um algoritmo de busca baseado em população que simula o comportamento social de pássaros num bando.

Segundo Kennedy e Eberhart (1995), o intuito inicial do algoritmo foi o de “simular graficamente a graciosa e imprevisível coreografia de um bando de pássaros” e segundo Engelbrecht (2007), a técnica PSO se transformou numa técnica de otimização considerada simples e eficiente.

No PSO os indivíduos da população, também chamados de partículas, “voam” pelo hiper-espaço de busca, sendo que sua posição representa uma solução. As mudanças de posição das partículas no espaço de busca são baseadas na tendência comportamental psicossocial de indivíduos de imitar o sucesso de outros indivíduos. Isso significa que as mudanças de uma partícula na nuvem é influenciada pela experiência ou conhecimento de seus vizinhos.

Para Engelbrecht (2007), o PSO é um tipo de algoritmo cooperativo simbiótico e isso ocasiona o fenômeno de que as partículas estocasticamente retornam a regiões de “sucesso” alcançadas anteriormente por causa dessa influência dos vizinhos (comportamento social).

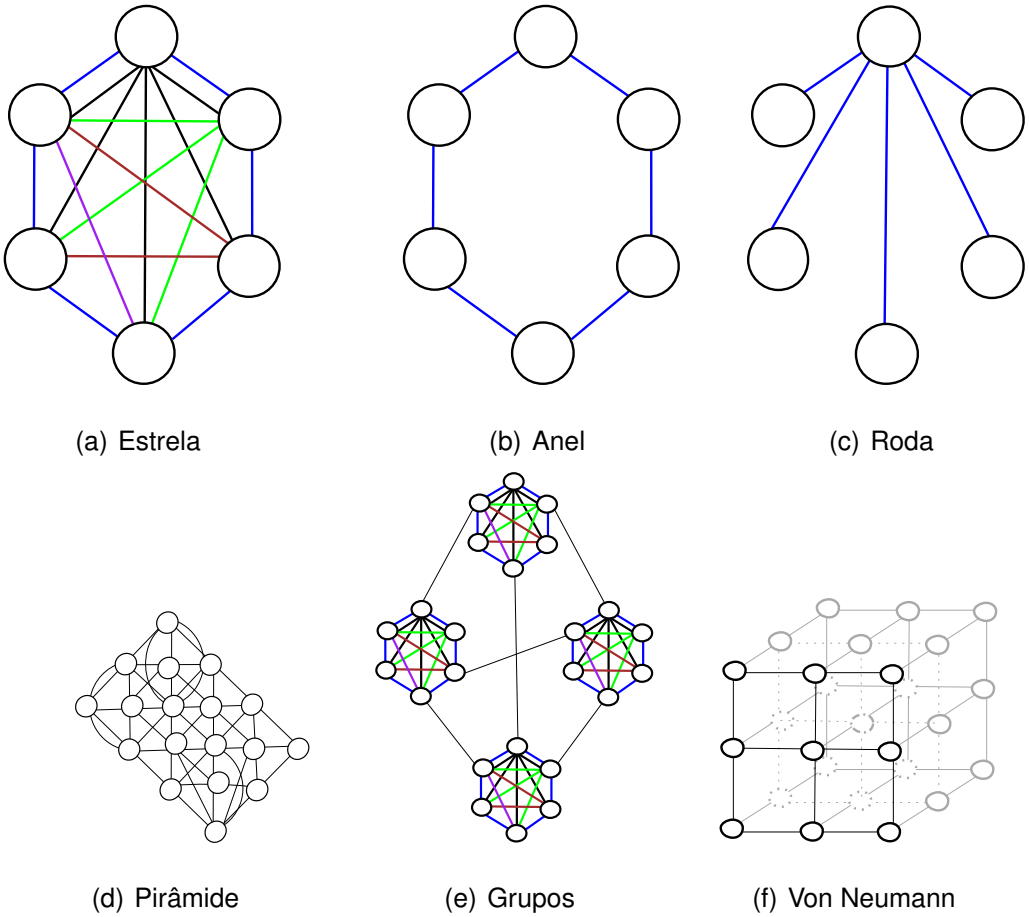
Princípios Fundamentais

Segundo Engelbrecht (2007), a característica que guia o PSO é a interação social entre as partículas. Os indivíduos do bando aprendem dos mais próximos e, então, se movem para posições mais próximas aos seus melhores vizinhos. Kennedy (1999) descreve algumas topologias usadas em PSO para a interação das partículas com seus vizinhos:

- **Topologia Estrela**: cada partícula pode se comunicar com todas as outras, formando uma rede social totalmente conectada, como é possível ver na Figura 9(a). Uma consequência dessa conectividade total é que todas as partículas tentarão imitar a melhor do enxame. Esta topologia foi utilizada na primeira versão do algoritmo PSO, conhecido como *gbest*;
- **Topologia Anel**: cada partícula se comunica com seus n vizinhos imediatos, no caso de $n = 2$, tem-se o caso da Figura 9(b). Uma consequência dessa topologia é a de que cada partícula tenta se mover para perto do seu melhor vizinho. Esta versão de topologia é utilizada na versão do algoritmo PSO conhecida como *lbest*;
- **Topologia Roda**: nesta estrutura os indivíduos numa vizinhança são isolados dos in-

divíduos numa outra vizinhança e entre si. Apenas um indivíduo da vizinhança serve como ponto focal, o qual compara as posições de todas as partículas a ele conectadas e ajusta sua posição em direção a melhor delas, esta topologia está mostrada na Figura 9(c);

- **Topologia Pirâmide**: estrutura social que tem forma de arame tri-dimensional, aparentemente sem uma padronização específica, como é possível ver na Figura 9(d);



(a) Estrela (b) Anel (c) Roda

(d) Pirâmide (e) Grupos (f) Von Neumann

FIGURA 9: Topologias para o Algoritmo PSO

- **Topologia com Quatro Grupos**: nesta estrutura, quatro grupos são formados com duas conexões entre os grupos. As partículas dentro de um grupo são conectadas a cinco vizinhos (Figura 9(e));

- **Topologia de Von Neumann**: nesta estrutura social as partículas são conectadas em forma de grade tri-dimensional (Figura 9(f)).

Vale observar que, em geral, o tipo de vizinhança é determinado com base no índice

numérico designado a uma partícula e não em medidas geométricas, como por exemplo, a posição ou a distância euclidiana. Embora desencorajada pelo aumento exponencial do custo computacional, alguns pesquisadores propuseram vizinhanças baseadas em distância euclidiana, como é o caso de Suganthan (1999).

Modelos de Algoritmos

Um enxame consiste num conjunto de partículas no qual cada partícula representa uma solução em potencial. Isso significa dizer que as partículas "voam" pelo hiperespaço de soluções no qual a posição de cada partícula muda de acordo com sua experiência e de seus vizinhos.

Seja $x_i(t)$ a posição da partícula P_i no hiperespaço no tempo $t-1$. Então, a posição de P_i é alterada adicionando-se uma velocidade $v_i(t)$ à sua atual posição, ou seja,

$$x_i(t) = x_i(t-1) + v_i(t). \tag{2.18}$$

O vetor de velocidade guia o processo de otimização e resume a informação social trocada até o momento. Em Engelbrecht (2007), Kennedy e Eberhart (1995) e Abraham, Guo e Liu (2006) é possível encontrar os dois algoritmos a seguir, os quais diferem apenas na amplitude da informação social trocada entre as partículas. Vale notar que estes algoritmos resumem os algoritmos PSO clássicos.

Melhor Global

No algoritmo para essa versão, mostrado na Figura 10, cada indivíduo compara sua posição atual com a melhor posição encontrada dentre todas as partículas do enxame (x_{gbest}). Isso quer dizer que a estrutura social é a conhecida como "estrela". Além disso, as partículas também usam informações sobre a melhor posição já ocupada por elas mesmas (x_{pbest}).

Vale salientar que o termo $\left(x_{pbest_i} - x_i(t)\right)$ é conhecido como Componente Cognitivo e, por sua vez, o termo $\left(x_{gbest_i} - x_i(t)\right)$ é conhecido como Componente Social, ambos referentes à modificação da posição da partícula.

Os coeficientes ρ_1 e ρ_2 são definidos como $\rho_1 = r_1 c_{1j}$ e $\rho_2 = r_2 c_2$, com r_1, r_2 uniformes, ou seja, $r_1, r_2 \sim U(0,1)$, e c_1, c_2 sendo constantes de aceleração positiva. Kennedy (1998) estudou os efeitos de ρ_1 e ρ_2 na trajetória das partículas e afirmou que se $c_1 + c_2 > 4$, a velocidade e a posição das partículas explodem em direção ao infinito.

1. Inicialize o enxame com partículas de posições aleatórias x_t e faça $t = 0$;
2. Avalie o desempenho $\mathbb{F}$ de cada partícula com relação à função objetivo utilizando sua posição atual x_t;
3. Compare o desempenho de cada indivíduo com seu melhor desempenho *pbest* até o momento:

   se $\mathbb{F}(x_t) < pbest_i$ então

   $pbest_i = \mathbb{F}(x_t)$

   $x_{pbest_i} = x_i(t)$;
4. Compare o desempenho de cada indivíduo com o melhor desempenho entre todas as partículas até o momento:

   se $\mathbb{F}(x_t) < gbest_i$ então

   $gbest_i = \mathbb{F}(x_t)$

   $x_{gbest_i} = x_i(t)$;
5. Mude o vetor de velocidade para cada partícula usando:

   $v_i(t) = v_i(t-1) + \rho_1 \cdot \left(x_{pbest_i} - x_i(t)\right) + \rho_2 \cdot \left(x_{gbest_i} - x_i(t)\right)$
6. Mova cada partícula para uma nova posição:

   $x_i(t) = x_i(t-1) + v_i(t)$ e $t = t + 1$
7. Volte ao Passo 2 até que ocorra convergência.

FIGURA 10: Algoritmo *gbest* - melhor indivíduo

Melhor Local

Nessa versão, mostrada na Figura 11, cada indivíduo compara sua posição atual com a

melhor posição encontrada dentre seus vizinhos. Obrigatoriamente, a topologia utilizada aqui é a de grupos (*Clusters*). Além disso, as partículas também usam informações sobre a melhor posição já ocupada por elas mesmas.

1. Inicialize o enxame com partículas de posições aleatórias x_t e faça $t = 0$;
2. Avalie o desempenho $\mathbb{F}$ de cada partícula com relação à função objetivo utilizando sua posição atual x_t;
3. Compare o desempenho de cada indivíduo com seu melhor desempenho *pbest* até o momento:

   se $\mathbb{F}(x_t) < pbest_i$ então

   $pbest_i = \mathbb{F}(x_t)$

   $x_{pbest_i} = x_i(t)$;
4. Compare o desempenho de cada indivíduo com o melhor desempenho entre seus vizinhos até o momento:

   se $\mathbb{F}(x_t) < lbest_i$ então

   $lbest_i = \mathbb{F}(x_t)$

   $x_{lbest_i} = x_i(t)$;
5. Mude o vetor de velocidade para cada partícula usando:

   $v_i(t) = v_i(t-1) + \rho_1 \cdot \left(x_{pbest_i} - x_i(t)\right) + \rho_2 \cdot \left(x_{lbest_i} - x_i(t)\right)$
6. Mova cada partícula para uma nova posição:

   $x_i(t) = x_i(t-1) + v_i(t)$ e $t = t + 1$
7. Volte ao Passo 2 até que ocorra convergência.

FIGURA 11: Algoritmo *lbest* - melhor indivíduo local

Vale salientar que o algoritmo *lbest* é mais lento no quesito convergência do que o melhor global, entretanto, segundo Engelbrecht (2007), pode levar a soluções melhores, pois faz uma busca mais abrangente no hiperespaço de soluções. Os coeficientes ρ_1 e ρ_2 têm as mesmas características do algoritmo *gbest*.

Parâmetros de um PSO

De acordo com Engelbrecht (2007), são seis os parâmetros a serem definidos num algoritmo PSO, a saber: a dimensão do problema, quantidade de indivíduos, limite superior de ρ, limite superior para a velocidade $v_i(t)$, tamanho da vizinhança e o peso de inércia ϕ. Uma vez que a influência de ρ sobre a posição e a velocidade das partículas já foi discutida, os outros parâmetros do PSO são discutidos a seguir:

- **Velocidade Máxima** (v_{max}): O objetivo de se colocar um limite na velocidade das partículas é o de evitar que estas se movam muito rapidamente de uma região à outra do hiperespaço. Entretanto, Clerc e Kennedy (2002) mostraram que v_{max} é desnecessária se,

$$v_i(t) = \kappa \cdot \left(v_i(t-1) + \rho_1 \cdot \left(x_{pbest_i} - x_i(t)\right) + \rho_2 \cdot \left(x_{gbest_i} - x_i(t)\right)\right)$$

  onde

$$\kappa = 1 - \frac{1}{\rho} + \frac{\sqrt{|\rho^2 - 4\rho|}}{2}$$

  com $\rho = \rho_1 + \rho_2 > 4$ e κ como de "coeficiente de constrição".

- **Tamanho da Vizinhança**: Aqui, Engelbrecht (2007) faz apenas uma ponderação com relação ao tamanho da vizinhança e a velocidade da convergência. Quanto menor for a vizinhança, ocasionando aumento na quantidade de vizinhanças, menos suscetível à mínimos locais o PSO se torna. Entretanto, se houver uma quantidade demasiada de vizinhanças a convergência se torna lenta. Por esse motivo só é possível definir este parâmetro empiricamente;

- **Peso de Inércia**: Melhores desempenhos podem ser alcançados aplicando-se um coeficiente (peso) de inércia ϕ à velocidade imediatamente anterior conforme a equação,

$$v_i(t) = \phi \cdot v_i(t-1) + \rho_1 \cdot \left(x_{pbest_i} - x_i(t)\right) + \rho_2 \cdot \left(x_{gbest_i} - x_i(t)\right).$$

  Pesos de inércia grandes fazem com que o algoritmo PSO explore mais o hiperespaço de soluções do que pesos de inérica pequenos. Geralmente, na literatura, inicia-se o PSO com um peso de inércia grande (próximo de "1") que decresce ao longo do tempo até chegar próximo de "0", ou seja, $\phi \in (0,\ 1)$.

Convergência e Critérios de Parada

Com relação à convergência, os dois parâmetros mais influentes que direcionam o comportamento do algoritmo PSO são o peso de inércia ϕ e as constantes de aceleração c_1 e c_2. O trabalho de Berg (2002) mostrou que, para garantir a convergência, nos casos em que não é pré-fixado um limite para a velocidade, a seguinte relação deve ser satisfeita,

$$\phi > 0,5 \cdot (c_1 + c_2) - 1 \tag{2.19}$$

com $\phi \leqq 1$. Vale notar que o PSO exibe um comportamento cíclico ou divergente caso a Equação 2.19 não seja satisfeita.

De acordo com Engelbrecht (2007), há quatro critérios de parada comumente encontrados na literatura para um PSO:

1. quando o algoritmo atingir um certo número de iterações realizadas;
2. quando a função objetivo for avaliada um certo número de vezes e/ou;
3. quando as mudanças na velocidade começarem a se aproximar de zero;
4. quando a partícula com valor mais adequado na função objetivo não for superada por uma certa quantidade de iterações.

É prática a utilização de mais de um destes critérios empregados no algoritmo. Os que mais aparecem na literatura são “quantidade de iterações” e “quantidade de avaliações da função objetivo”, por não dependerem de características do problema para serem alcançados.

2.4 SISTEMAS DIFUSOS

A lógica booleana se baseia na premissa de que elementos pertençam ou não a um conjunto. Por exemplo, uma pessoa pode ou não pertencer ao conjunto daqueles que assistem jornal à noite, assim como outra pessoa pode pertencer ou não ao conjunto dos torcedores de um time de futebol qualquer.

Entretanto, de acordo com Engelbrecht (2007), a razão humana sempre inclui um grau de incerteza em suas afirmações. Por exemplo, os seres humanos, são capazes de entender a frase “Alguns estudantes de matemática são capazes de resolver problemas em diversas áreas”.

A questão que surge é: como um computador poderia representar e entender essa afirmação?

Os Sistemas Difusos (SD), compostos por conjuntos difusos e lógica difusa, permitem trazer uma representação bastante completa desse tipo de sentença. Num conjunto difuso, um elemento tem um grau de probabilidade de pertencer a um conjunto, fazendo com que a incerteza inerente a afirmações como essa sejam levadas em conta em sua forma de representação.

Isso quer dizer que, basicamente, a lógica difusa tem uma sistematização similar à lógica booleana, com a principal diferença na maneira como as relações de pertinência são encaradas. A decisão da pertinência ou não de um elemento a um conjunto deixa de ser binária para ser probabilística. Os SD também são aplicáveis em Sistemas de Controle, Transmissão e Sistemas de Freios em automóveis, Controle de Tráfego, entre outros.

2.5 MÉTODOS ESTATÍSTICOS

Esta seção apresenta, de maneira breve, os conceitos relacionados aos métodos estatísticos utilizados ao longo da pesquisa. Como a abordagem estatística não é o foco da pesquisa, maiores esclarecimentos sobre os fundamentos dos métodos em questão podem ser encontrados nas bibliografias indicadas.

2.5.1 **Função Discriminante Linear de Fisher**

Em 1936, Fisher *apud* Mingoti (2005) introduziu a ideia de se construir funções discriminantes entre duas e várias populações a partir de combinações lineares das variáveis originais, algo bastante similar ao que é feito em Análise de Componentes Principais e Análise Fatorial.

Este método, segundo Johnson e Wichern (2002) e Mingoti (2005), não precisa da suposição de que as diversas populações sejam normais, entretanto, assume-se que as matrizes

de covariâncias populacionais Σ das g populações com p variáveis aleatórias (cada) são, no mínimo, aproximadamente iguais, ou seja,

$$\Sigma_1 = \Sigma_2 = \ldots = \Sigma_g = \Sigma.$$

Desse modo, se $\bar{X}$ for o vetor de médias das g populações e B a matriz de variação entre grupos populacionais, tal que

$$B_{p\times p} = \sum_{i=1}^{g} \left(\bar{X}^i - \bar{X}\right)\left(\bar{X}^i - \bar{X}\right)', \tag{2.20}$$

então, a combinação linear $Y = C'\bar{X}$ tem esperança dada por

$$E(Y) = C' \cdot E(\bar{X}) = C'\bar{X}^i$$

para $i = 1, 2, \ldots, g$, e variância dada por,

$$\sigma_Y^2 = V\left(C'\bar{X}\right) = C'V(\bar{X})C = C'\Sigma C$$

para todas as populações. Assim, o valor esperado para a média univariada,

$$E(Y) = \mu_i = C'\mu_i$$

se altera quando a população da qual $\bar{X}$ é selecionada é outra. Tem-se então uma média global dada por,

$$\bar{\mu} = \frac{1}{g}\sum_{i=1}^{g}\mu_i$$

e consequentemente a razão entre a soma dos quadrados das distâncias das populações para a média global de Y e a variância de Y, que denotaremos por Π é dada por,

$$\Pi = \frac{\sum_{i=1}^{g}(\mu_i - \bar{\mu})}{\sigma_Y^2}$$

ou seja,

$$\Pi = \frac{C^{'} BC}{C^{'} \Sigma C} \tag{2.21}$$

que é uma generalização para o caso de duas populações.

Esta razão mede a variabilidade entre os grupos de valores Y em relação à variabilidade comum entre os grupos. A fim de maximizar a razão na Equação 2.21, escalona-se C de maneira que $C^{'} \Sigma C = 1$.

Além de B como variância entre grupos, podemos definir uma estimativa da variância Σ com base na variância amostral dentro dos grupos dada por,

$$W_{p \times p} = \sum_{i=1}^{g} \sum_{k=1}^{n_i} \left(X_{ik} - \bar{X}^i\right) \left(X_{ik} - \bar{X}^i\right)^{'} \tag{2.22}$$

e, consequentemente, a Equação 2.21, torna-se,

$$\Pi = \frac{C^{'} BC}{C^{'} WC}. \tag{2.23}$$

Sejam $\hat{\lambda}_1, \ldots, \hat{\lambda}_s$ os $s = \min(g-1, p)$ autovalores não negativos de $W^{-1}B^j$ e $\hat{e}_1, \ldots, \hat{e}_s$ seus correspondentes autovetores escalonados de maneira que $\hat{e}^{'} S_{\text{ajustado}} \hat{e} = 1$. Então, o vetor de coeficientes que maximiza a razão dada na Equação 2.23 é dado por $C = \hat{e}$. A combinação linear $C_1^{'} X = \hat{e}_1^{'} X$ é então chamada de *primeira discriminante da amostra*. A escolha $C_2 = \hat{e}_2$ gera a *segunda discriminante da amostra*, e assim sucessivamente para as $k \leq s$ discriminantes amostrais.

Após a construção das funções discriminantes, para cada elemento (vetor) amostral X^j, existirá um vetor com seus escores nessas funções, ou seja,

$$\hat{Y}_j^{'} = \left[\hat{e}_1^{'} X^j \hat{e}_2^{'} X^j \ldots \hat{e}_s^{'} X^j\right].$$

O mesmo vale para os vetores de média observados para cada população, isto é,

$$\hat{\bar{Y}}_j^{'} = \left[\hat{e}_1^{'} \bar{X}_j \hat{e}_2^{'} \bar{X}_j \ldots \hat{e}_s^{'} \bar{X}_j\right]$$

com $j = 1, 2, 3, \ldots, g$. Calcula-se então a distância euclidiana entre os vetores $\hat{Y}_j$ e $\hat{\bar{Y}}_j$, para todo $j = 1, 2, 3, \ldots, g$, sendo o correspondente elemento amostral j classificado na população cuja distância for menor.

Uma prática bastante comum ao aplicar a FDLF é utilizar apenas a primeira função discriminante canônica, por ser a de maior poder de separação entre as populações. Entretanto, de acordo com Johnson e Wichern (2002), a utilização das funções "mais importantes", isto é, aquelas associadas a autovalores significativamente maiores do que zero, pode levar a melhores resultados, pois essa estratégia leva a uma melhor aproximação de W e B.

2.5.2 **Análise de Variância Multivariada**

Frequentemente, as médias de mais de duas populações precisam ser comparadas para fins teóricos ou práticos. Considerando *g* amostras aleatórias coletadas de *g* populações, comumente são arranjadas como

$$\begin{aligned} &\text{População 1:} && X_{11}, X_{12}, \ldots, X_{1n_1} \\ &\text{População 2:} && X_{21}, X_{22}, \ldots, X_{2n_2} \\ &\qquad\vdots && \vdots \\ &\text{População g:} && X_{g1}, X_{g2}, \ldots, X_{gn_g} \end{aligned}$$

a Análise de Variância Multivariada, ou simplesmente MANOVA (*Multivariate Analysis of Variance*), é utilizada para verificar se existe ou não diferença estatisticamente significativa entre os vetores médios das *g* populações, considerando-se um nível de significância α.

Ao aplicar a MANOVA, supõe-se que:

1. as amostras aleatórias provenientes das *g* populações são independentes estatisticamente;
2. que existe homocedasticidade entre as populações, ou seja, que a matriz de covariâncias Σ é a mesma (ou aproximadamente a mesma) para todas as populações e;
3. cada população, isoladamente, é normal multivariada.

Entretanto, segundo Johnson e Wichern (2002) "a condição 3 pode ser relaxada 'apelando' ao teorema central do limite quando as amostras aleatórias são grandes". Em geral, uma amostra com, pelo menos, 30 observações em cada grupo é suficiente.

Ainda segundo Johnson e Wichern (2002), o modelo MANOVA para a comparação de médias de *g* populações é dado por,

$$X_{lj} = \mu + \tau_l + e_{lj},\ j = 1,\ldots,n_l \text{ e } l = 1,\ldots,g \tag{2.24}$$

onde $e_{lj} \sim N_p(0,\Sigma)$, μ é uma média geral e τ representa o efeito do *l*-ésimo tratamento com

$$\sum_{l=1}^{g} n_l \tau_l = 0.$$

Os passos para a aplicação do modelo MANOVA, de acordo com Krzanowski (1988) e Johnson e Wichern (2002), onde é possível encontrar mais detalhes sobre o fundamento do método, são:

1. Enunciar as hipóteses:

   H_0: $\mu_1 = \mu_2 = \ldots \mu_g$ (hipótese nula)

   H_1: pelo menos uma das médias é diferente das demais (hipótese alternativa).

2. Construir a tabela MANOVA, conforme a Tabela 1, abaixo:

TABELA 1: Tabela Manova

Fonte de Variação	**Matriz de Somas de Quadrados e Produtos Cruzados**	**Graus de Liberdade**
Tratamento	$B = \sum_{l=1}^{g} n_t (\bar{x}_l - \bar{x})(\bar{x}_l - \bar{x})'$	$g-1$
Resíduo (Erro)	$W = \sum_{l=1}^{g} n_t \sum_{j=1}^{n_l} (x_{lj} - \bar{x}_l)(x_{lj} - \bar{x}_l)'$	$\sum_{l=1}^{g} n_t - g$
Total	$B+W = \sum_{l=1}^{g} n_t \sum_{j=1}^{n_l} (x_{lj} - \bar{x})(x_{lj} - \bar{x})'$	$\sum_{l=1}^{g} n_t - 1$

3. Calcular a estatística do teste,

$$\Lambda^* = \frac{|W|}{|B+W|}$$

onde Λ^* é conhecido como *Lâmbda de Wilks* e é equivalente ao teste $\mathbb{F}$ da Análise de Variância Multivariada. Ainda x_{lj} é a *j*-ésima observação da *l*-ésima amostra (tratamento), $\bar{x}_l$ é a média da *l*-ésima amostra (tratamento) e $\bar{x}$ é a média global (todas as amostras).

Em alguns casos específicos, a distribuição de Λ^* pode ser determinada de acordo com a Tabela 2, para grandes amostras e outros casos, a Equação 2.25 fornece a distribuição amostral.

TABELA 2: Distribuição do Lâmbda de Wilks

# Variáveis	# Grupos	Distribuição amostral
$p=1$	$g \geq 2$	$\left(\frac{\sum n_t - g}{g-1}\right)\left(\frac{1-\Lambda^*}{\Lambda^*}\right) \sim \mathbb{F}_{g-1,\sum n_t - g}$
$p=2$	$g \geq 2$	$\left(\frac{\sum n_t - g - 1}{g-1}\right)\left(\frac{1-\sqrt{\Lambda^*}}{\sqrt{\Lambda^*}}\right) \sim \mathbb{F}_{2(g-1),2(\sum n_t - g - 1)}$
$p \geq 1$	$g=2$	$\left(\frac{\sum n_t - p - 1}{p}\right)\left(\frac{1-\Lambda^*}{\Lambda^*}\right) \sim \mathbb{F}_{p,\sum n_t - p - 1}$
$p \geq 1$	$g=3$	$\left(\frac{\sum n_t - p - 2}{p}\right)\left(\frac{1-\sqrt{\Lambda^*}}{\sqrt{\Lambda^*}}\right) \sim \mathbb{F}_{2p,2(\sum n_t - p - 2)}$

$$-\left(n-1-\frac{p+g}{2}\right)\cdot \ln\Lambda^* \sim \chi^2_{p(g-1)}. \tag{2.25}$$

Finalmente, com base na relação descrita na Equação 2.25, rejeita-se H_0 se

$$-\left(n-1-\frac{p+g}{2}\right)\cdot \ln\Lambda^* > \chi^2_{p(g-1)}(\alpha).$$

em que α é o nível de significância do teste.

2.5.3 **Teste de Mann-Whitney**

O Teste de *Mann-Whitney* é um teste não paramétrico que tem como finalidade testar se duas amostras independentes, de tamanhos n_1 e n_2, provêm ou não de populações idênticas. Segundo Marques (2004), esse teste somente deve ser utilizado quando as amostras forem independentes e os dados envolvidos forem de, pelo menos, nível ordinal de mensuração.

Segundo Fonseca (1996), este método é uma interessante alternativa ao teste paramétrico (teste *t*) para igualdade entre duas médias, pois não é necessário fazer qualquer suposição sobre as distribuições populacionais e suas variâncias.

Os passos para aplicação desse teste, como é possível ver também em Fonseca (1996) e Marques (2004) são:

1. Enunciar as hipóteses:

   H_0 : as duas amostras provêm de populações idênticas

   H_1 : as duas amostras provêm de populações diferentes.

2. Fixar o nível de significância α.

   Executar os seguintes procedimentos:

   (a) combinar as duas amostras e colocá-las em ordem crescente;

   (b) denominar por n_1 o menor tamanho de amostra;

   (c) atribuir posto "1" ao menor valor numérico. Em caso de empate, atribui-se o posto médio.

3. Determinar o valor de

$$U = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1$$

   onde R_1 é a soma dos postos atribuídos à amostra de tamanho n_1.

4. Calcular a estatística,

$$z = \frac{U - \frac{n_1 n_2}{2}}{\sqrt{\frac{n_1 n_2 (n_1+n_2+1)}{12}}}.$$

5. Decisão:

   Se $-z_{\frac{\alpha}{2}} \leq z \leq z_{\frac{\alpha}{2}}$, não se pode rejeitar H_0

   Caso contrário, rejeita-se H_0, com risco α de que não haja diferença entre os grupos.

2.5.4 **Coeficiente *Kappa* de Cohen**

De acordo com Carletta (1996), o coeficiente *kappa* (κ) mede o grau concordância real entre a classificação fornecida por um classificador e a classificação real corrigido pelo grau concordância esperada. De acordo com Carletta (1996) e Sim e Wright (2005), essa concordância é dada pela equação,

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{2.26}$$

onde $P(A)$ é a proporção observada de vezes em que a classificação fornecida e a classificação esperada concordam e $P(E)$ é a proporção de concordâncias esperadas.

O coeficiente *kappa* também possui relação com a chamada “matriz de confusão” de um classificador, a qual apresenta as tendências de acerto e erro do classificador em questão de acordo com as classes, levando em consideração a classificação sugerida pelo classificador e a real, conforme a Tabela 3.

TABELA 3: Típica Matriz de Confusão de um Classificador (2 classes)

		Classe Predita	
		1	**2**
Classe Real	**1**	Verd. Classe 1	Falso Classe 2
	2	Falso Classe 1	Verd. Classe 2

Um método prático para o cálculo de κ pode ser encontrado em Figueiredo e Vieira (2007): dada uma matriz de confusão, como a da Tabela 3, é possível obter *kappa* pela expressão,

$$\kappa = \frac{n\sum_{i=1}^{c} x_{ii} - \sum_{i=1}^{c} x_{i+}x_{+i}}{n^2 - \sum_{i=1}^{c} x_{i+}x_{+i}} \tag{2.27}$$

onde c é a quantidade de classes, x_{ii} é o valor na linha *i* e coluna *i* da matriz de confusão, x_{i+} é a soma da linha *i*, x_{+i} é a soma da coluna *i* e *n* é a quantidade de elementos da matriz de confusão.

2.6 TRABALHOS RELACIONADOS

Os pesquisadores Li e Ling (2011) aplicaram a ideia de obter os pesos da camada intermediária numa RBFNN - *Radial Basis Function Neural Network*, Rede Neural de Base Radial, para desenvolver um modelo de controle preditivo generalizado numa unidade geradora de energia que possuía, na época da pesquisa, duas variáveis de entrada e duas variáveis de

saída. O experimento realizado consiste na comparação do desempenho do algoritmo proposto (chamado de GA-RBF em alusão aos Algoritmos Genéticos e à RBFNN) com o desempenho de um *Perceptron* de Múltiplas Camadas com a utilização do algoritmo de treinamento de retro-propagação do erro. A técnica proposta apresentou um erro quadrático médio da ordem de 10^{-5}, enquanto o Perceptron testado apresentou um erro de aproximadamente 10^{-1}.

O trabalho de Changbing e Wei (2010) aplicou uma RBFNN com treinamento via AG na previsão de risco num ambiente aquático. Os testes mostraram que o modelo proposto apresentou um desempenho superior ao "modelo cinza", que utiliza uma equação diferencial de primeira ordem para estimar valores com base em dados históricos.

Já Ming, Bin e Zhong (2010) aplicaram basicamente a mesma ideia na predição do fluxo numa rede. Os resultados do experimento realizado, no qual o fluxo de uma rede com 40 pontos foi predito, mostrou que o erro relativo na predição de uma RBFNN com o treinamento via pseudo-inversão (tradicional) foi de $0,0247$ enquanto que a RBFNN com treinamento via AG conseguiu errar apenas $0,0177$, uma redução de quase 30% no erro.

A pesquisa de Kurban e Besdok (2009) comparou quatro algoritmos para o treinamento de uma RBFNN: Colônia de Abelhas, Algoritmos Genéticos, Filtro de Kalman e Descida do Gradiente. Nos três bancos de dados testados disponíveis em Frank e Asuncion (2010) e no banco de dados da aplicação proposta pelos autores o algoritmo "Colônia de Abelhas" apresentou um desempenho levemente superior ao AG, ambos apresentando uma acurácia acima de 90% de acerto no conjunto de testes nos bancos de dados em Frank e Asuncion (2010) e acima de 70% no banco de dados da aplicação em sensores.

Uma abordagem bastante abrangente, em relação ao que normalmente se faz para treinar uma rede neural utilizando os Algoritmos Genéticos, pode ser encontrada em Zhangang, Yanbo e Cheng (2007), pois nesta pesquisa todos os parâmetros da RBFNN são convertidos em indivíduos (vetores) para que se aplique a sequência de AG e, posteriormente, ao fim de cada geração, ordenadamente, os componentes deste indivíduo são designados a cada parâmetro da RBFNN tomando elementos do vetor de acordo com a quantidade de informações necessárias a cada parâmetro. O trabalho é relacionado à previsão da série temporal de carga elétrica numa cidade e o maior erro absoluto no conjunto de testes foi de 3,05%.

Um pouco mais conservadora é a pesquisa de Jareanpon et al. (2004) em que o único parâmetro da RBFNN calculado via AG foi o raio de abertura da função gaussiana utilizada para calcular os valores de ativação dos neurônios da camada intermediária numa RBFNN. A cada mudança no modelo uma população inicial era criada e, posteriormente, a sequência de passos característicos de um AG foram aplicados em bancos de dados relacionados a níveis de precipitação pluviométrica do departamento de Meteorologia da Tailândia, mais especificamente em 13 cidades, e os valores da soma quadrática dos erros na predição, em todos os casos, ficaram abaixo de $0,1$, sendo o menor deles $1,02 \cdot 10^{-21}$ e o maior deles $7,67 \cdot 10^{-2}$.

3 MATERIAL E MÉTODO

3.1 MATERIAL

3.1.1 **Bancos de Dados Utilizados nos Experimentos**

Foram utilizados os seguintes bancos de dados para a realização do experimento delineado a fim de avaliar os métodos: *Cancer* (cancer), *Contraceptive Method Choice* (cmc), Car Evaluation (careval), Hepatitis (hepatitis), *Iris* (iris), e Page Blocks Classification (pageblocks). Todos estes bancos de dados podem ser encontrados em Frank e Asuncion (2010).

Vale ressaltar que a tarefa associada a todos estes conjuntos é a classificação de padrões. A Tabela 4 traz uma descrição dos bancos de dados (conjuntos) em questão, as informações sobre as variáveis e seus significados em cada banco de dados também podem ser encontradas em Frank e Asuncion (2010).

TABELA 4: Características dos bancos de dados

Banco de Dados	**Variáveis**	**Classes**	**Quantidade de Observações**
cancer	9	2	699
cmc	9	3	1473
careval	6	4	1728
hepatitis	19	2	155
iris	4	3	150
pageblocks	10	5	5473

Na Tabela 4 é possível visualizar a quantidade de variáveis em cada banco de dados bem como a quantidade de classes existentes. Além disso, também é possível visualizar a quantidade de observações.

3.1.2 ***Hardware* e *Software* Utilizados**

O computador utilizado para a realização do experimento descrito a seguir possui a seguinte configuração: Processador Intel Core i7 2.2 *gigahertz*, 8 *gigabytes* de memória ram e

um disco rígido de 1 *terabyte*. O Sistema Operacional instalado na máquina é o Ubuntu Linux 12.10 e a versão do MATLAB® instalada é a R2011b.

3.2 MÉTODO

3.2.1 **MH de Busca Baseadas em População no Treinamento das RBFNNs**

O objetivo desta seção é mostrar como realizar o treinamento de uma RBFNN utilizando as duas técnicas de IC apresentadas no Capítulo 2. Em outras palavras, serão apresentadas duas metodologias híbridas: uma composta por RBFNN e AG e a outra composta por RBFNN e PSO.

Metodologia Híbrida RBFNN-AG

A sigla RBFNN-AG significa aqui uma metodologia híbrida composta por uma Rede Neural de Base Radial e Algoritmos Genéticos para calcular um ou mais de seus parâmetros. Existem, pelo menos, três diferentes estratégias de modelagem que podem ser encontradas na literatura no que diz respeito ao "nível de hibridez" entre a combinação entre uma RBFNN e um AG, como é possível observar no Capítulo 1.

Tendo em vista que há três parâmetros a serem definidos via algoritmos específicos numa RBFNN: a matriz de centros, o vetor de raios de abertura das funções de ativação da camada escondida (vetor de *spreads*) e a matriz de pesos da camada escondida, é possível, portanto, sistematizar três níveis de hibridez da metodologia RBFNN-AG. Dentro de cada nível há outras subdivisões que serão apresentadas a seguir.

RBFNN-AG-I

Como a própria classificação sugere, uma metodologia RBFNN-AG-I consiste em utilizar os AG para obter um dos três parâmetros via algoritmos específicos.

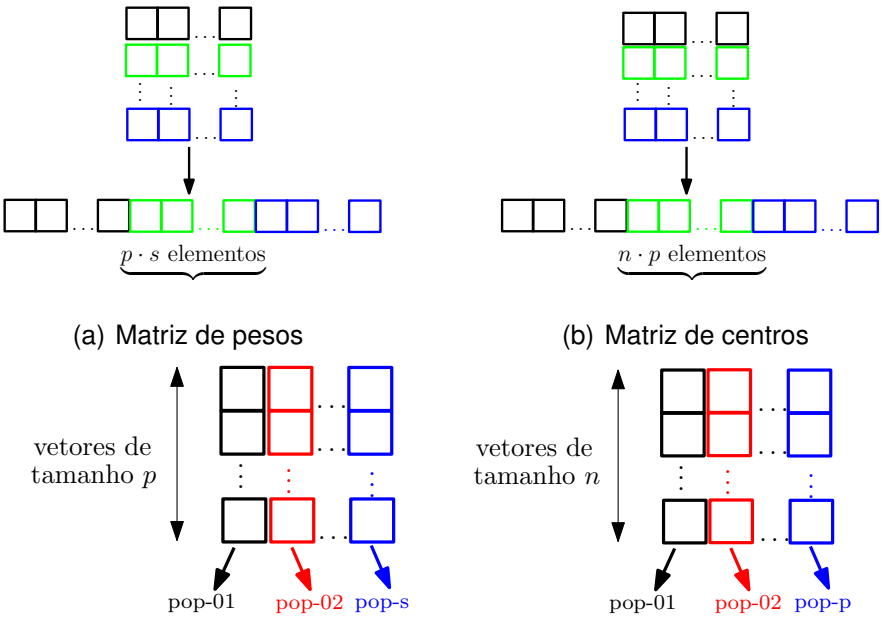
Dos três parâmetros, o único na forma vetorial é o vetor de raios de abertura (*spreads*) e, neste caso, nenhuma adaptação na forma como o AG recebe as informações é necessária. Basta adotar um procedimento padrão de aplicação de Algoritmos Genéticos a um problema vetorial.

No caso da matriz de centros C e/ou a matriz de pesos W,

$$C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ c_{p1} & c_{p2} & \dots & c_{pn} \end{pmatrix}$$

$$w = \begin{pmatrix} w_{10} & w_{11} & \dots & w_{1s} \\ w_{20} & w_{21} & \dots & w_{2s} \\ \vdots & \ddots & \vdots & \vdots \\ w_{p0} & w_{p1} & \dots & w_{ps} \end{pmatrix}$$

há duas estratégias bastante distintas:



(a) Matriz de pesos (b) Matriz de centros

(c) Matriz de centros (d) Matriz de centros

FIGURA 12: Configuração de indivíduos para RBFNN-AG

- **População Genética Única**: Uma vez que, normalmente, um AG assume uma função

do tipo $\mathbb{R}^n \rightarrow \mathbb{R}$ como FF, é necessário um algoritmo que transforme as matrizes $W_{p\times s}$ e $C_{n\times p}$ em vetores de ordem $p \cdot s$ e $n \cdot p$, respectivamente, conforme ilustram as Figuras 12(a) e 12(b). Isso equivale a transformar dois problemas $\mathbb{R}^{n\times p} \rightarrow \mathbb{R}$ e $\mathbb{R}^{p\times s} \rightarrow \mathbb{R}$ em $\mathbb{R}^{(n\cdot p)} \rightarrow \mathbb{R}$ e $\mathbb{R}^{(p\cdot s)} \rightarrow \mathbb{R}$, respectivamente.

- **Múltiplas Populações Genéticas**: Aqui, em vez de transformar a matriz num vetor, a ideia é trabalhar com s colunas representando populações contendo vetores com ordem igual ao número de linhas da matriz, como é possível visualizar nas Figuras 12(c) e 12(d). Em ambos os casos, as populações podem compartilhar valores dos parâmetros ou ter estes valores independentes.

RBFNN-AG-II

Seguindo a sistematização, um algoritmo RBFNN-AG-II possui nível dois de hibridez, ou seja, utiliza o algoritmo AG para calcular dois dos três parâmetros mencionados no início desta Seção, podendo combinar:

- cálculo da matriz de centros (C) e cálculo do vetor de *spreads* (S);
- cálculo do vetor de *spreads* (S) e cálculo da matriz de pesos (W) e;
- cálculo da matriz de centros (C) e da matriz de pesos (W).

Geralmente, independentemente da combinação escolhida, os valores dos parâmetros do algoritmo genético são independentes em relação a um parâmetro e outro da RBFNN. Isso quer dizer que os valores dos parâmetros utilizados, por exemplo, para calcular a matriz (W), em geral, não são os mesmos utilizados para calcular a matriz (C) e o mesmo vale para as outras combinações.

RBFNN-AG-III

Finalmente, uma RBFNN-AG-III utiliza um AG para calcular os seus três principais parâmetros, mencionados no início desta Seção. Geralmente também são utilizados AGs distintos (com parâmetros distintos) para cada problema (cálculo de parâmetro) da RBFNN.

Definição da Função de Fitness

Independentemente da estratégia de modelagem escolhida e do grau de hibridez do modelo adotado, o objetivo é obter os parâmetros de maneira a fazer com que a RBFNN em questão consiga obter o maior percentual de classificações corretas possível.

Isso quer dizer que pode-se ter três tipos de problema no que diz respeito ao domínio e imagem da Função de *Fitness*:

1. no caso da RBFNN-AG-I, pode-se ter:
   (a) $\mathbb{R}^{p \times n} \rightarrow \mathbb{R}$, caso o parâmetro calculado seja a matriz de centros;
   (b) $\mathbb{R}^{p} \rightarrow \mathbb{R}$, caso seja o vetor de *spreads* ou;
   (c) $\mathbb{R}^{p \times s} \rightarrow \mathbb{R}$, caso o parâmetro seja a matriz de pesos.
2. no caso da RBFNN-AG-II, pode-se ter:
   (a) $\mathbb{R}^{p \times n} \rightarrow \mathbb{R}$, caso o parâmetro calculado seja a matriz de centros e $\mathbb{R}^{p} \rightarrow \mathbb{R}$, caso seja o vetor de *spreads*;
   (b) $\mathbb{R}^{p \times n} \rightarrow \mathbb{R}$, caso o parâmetro calculado seja a matriz de centros e $\mathbb{R}^{p \times s} \rightarrow \mathbb{R}$, caso o parâmetro seja a matriz de pesos ou;
   (c) $\mathbb{R}^{p} \rightarrow \mathbb{R}$, caso seja o vetor de *spreads* e $\mathbb{R}^{p \times s} \rightarrow \mathbb{R}$, caso o parâmetro seja a matriz de pesos.
3. no caso da RBFNN-AG-III, tem-se os domínios dos três itens mencionados anteriormente e imagens reais, ou seja, $\mathbb{R}^{p \times n} \rightarrow \mathbb{R}$, $\mathbb{R}^{p} \rightarrow \mathbb{R}$ e $\mathbb{R}^{p \times s} \rightarrow \mathbb{R}$.

Seleção para a Reprodução

Conforme já exposto anteriormente, não existe apenas um método de seleção para definir quais indivíduos da população geram descendência. Isso quer dizer que são necessários alguns testes preliminares a fim de verificar qual dos métodos apresentados na Seção 2.2.1 mostra o melhor desempenho.

Operadores Contínuos de Reprodução

Uma das pesquisas que utiliza a representação real foi realizada por Michalewicz, Logan e Swaminathan (1994) que sistematizou três operadores de cruzamento (*Crossover*) e quatro operadores de mutação, sendo esses: *Crossover* Simples, *Crossover* Aritmético, *Crossover Heurístico*, Mutação Uniforme, Mutação de Limite, Mutação Não Uniforme (simples e múltipla). Todos esses operadores foram utilizados nessa pesquisa e suas descrições podem ser encontrados na Seção 2.2.1.

Critérios de Parada

O(s) critério(s) de parada pode(m) ser um ou mais entre os apresentados na Seção 2.2.1. A escolha pelos critérios “quantidade máxima de gerações” e “melhor indivíduo não ser superado por uma certa quantidade de gerações” são amplamente utilizados em trabalhos encontrados na literatura, pois são os únicos que garantem, de fato, que o algoritmo pare em algum momento.

Os critérios “o melhor indivíduo alcançar determinado valor” e “uma certa quantidade de indivíduos serem geneticamente iguais” não possuem garantia de convergência. Por esse motivo figuram como “coadjuvantes” na equipe de critérios responsáveis por evitar que o algoritmo entre num *loop* e não consiga sair.

Metodologia RBFNN-PSO

A sigla RBFNN-PSO significa aqui uma metodologia híbrida composta por uma Rede Neural de Base Radial e *Particle Swarm Optimization* (Nuvem de Partículas) para calcular um ou mais de seus parâmetros. Assim como no caso dos AG, existem, pelo menos, três diferentes estratégias de modelagem que podem ser encontradas na literatura no que diz respeito ao "nível de hibridez" entre a combinação entre uma RBFNN e um PSO.

Como no caso dos AGs, é possível, portanto, sistematizar três níveis de hibridez da metodologia RBFNN-AG. Dentro de cada nível há outras subdivisões que serão apresentadas a seguir.

RBFNN-PSO-I

Assim como no caso da metodologia RBFNN-AG, a indicação "I" significa que apenas um dos parâmetros - Matriz de Centros, Vetor de *Spreads* ou Matriz de pesos - será calculado utilizando-se um dos modelos de algoritmo PSO.

A similaridade nas características de aplicação das duas técnicas continua no que diz respeito à preparação dos indivíduos para a aplicação do algoritmo em si. Novamente, no caso do Vetor de *Spreads* não se faz necessária qualquer preparação e, no caso da Matriz de Centros (C) e da Matriz de pesos (W) são necessárias as adaptações nas Figuras 12(b) e 12(a), para o caso de uma única população, ou aquelas contidas nas Figuras 12(d) e 12(c), para o caso de múltiplas populações.

RBFNN-PSO-II

Seguindo a ideia da metodologia RBFNN-AG-II, a RBFNN-PSO-II consiste em calcular dois dos três parâmetros de uma RBFNN utilizando um PSO, podendo combinar, dois a dois:

- cálculo da matriz de centros (C) e cálculo do vetor de *spreads* (S);
- cálculo do vetor de *spreads* (S) e cálculo da matriz de pesos (W) e;
- cálculo da matriz de centros (C) e da matriz de pesos (W).

Nessa abordagem, em geral, as instâncias do algoritmo não compartilham valores dos parâmetros. Trabalha-se com enxames independentes, seguindo a mesma ideia da abordagem via AG.

RBFNN-PSO-III

Finalmente, uma RBFNN-PSO-III utiliza um PSO para calcular os seus três principais parâmetros, mencionados no início desta Seção. São utilizados PSOs com parâmetros distintos para cada cálculo de parâmetro da RBFNN, assim como se faz na abordagem via AG.

Função Objetivo

Seja qual for o grau de hibridez do modelo adotado o objetivo é obter os parâmetros de maneira a fazer com que a RBFNN em questão consiga obter o maior percentual de classificações corretas possível.

Isso quer dizer que, assim como na abordagem via AG, pode-se ter três tipos de problema no que diz respeito ao domínio e imagem da Função Objetivo. Os três tipos de problema são aqueles mencionados anteriormente, diferindo apenas no termo “Função Objetivo” em substituição a “Função de *Fitness*”.

Modelos de Algoritmo

Os dois modelos de algoritmo que resumem o funcionamento do PSO são:

- *gbest* da Seção 2.3.1, que leva em consideração informações de todas as partículas do enxame, incluindo a partícula em análise, para buscar melhores soluções e;
- *lbest* da Seção 2.3.1, que leva em consideração informações da partícula em análise e de suas vizinhas do grupo ao qual ela pertence para buscar melhores soluções.

Vale salientar que modificações nos dois algoritmos já foram feitas por alguns pesquisadores. Porém, estes dois algoritmos citados são suficientes para entender a essência do funcionamento de um PSO.

Critérios de Parada

Foram utilizados dois dos três critérios de parada mencionados na Seção 2.3.1: quantidade de iterações, quantidade de iterações sem melhorias.

3.2.2 **Adaptação Proposta**

Uma Nova Maneira de Enxergar Indivíduos

Em geral, em AG, um indivíduo é visto como um vetor de informações genéticas, como uma "tira" que coloca genes lado a lado, como é possível ver na Figura 13. Entretanto, nem sempre o problema que necessita ser resolvido tem resposta na forma vetorial, como, por exemplo, é o caso do cálculo da matriz de pesos de uma RBFNN.

1,5	9,2	31,2	6,8	0,8	7,4	1,3	5,7

FIGURA 13: Representação de um indivíduo em AG tradicional

Isso quer dizer que para resolver um problema matricial via AG, seria necessário que as soluções fossem transformadas em vetores para que os operadores genéticos pudessem ser aplicados e, posteriormente, transformados em matrizes para cálculo do *fitness* a cada iteração.

Esta tese avalia a hipótese de criar uma extensão dos operadores propostos por Michalewicz, Logan e Swaminathan (1994), para o caso em que os indivíduos são matrizes em substituição aos vetores, como em geral acontece no caso da utilização dos AGs para o cálculo da matriz de pesos da camada intermediária das RBFNNs. Nos trabalhos existentes na literatura, como por exemplo em Mota et al. (2012) a sistematização utilizada é a de transformar a matriz de pesos num vetor contendo todos os elementos das matrizes.

Como consequência da adaptação de operadores para indivíduos matriciais, a proposta deste trabalho consiste também em mudar a forma que se configura um indivíduo em AG, trazendo vantagens de implementação, seccionando um indivíduo não em genes, mas em cadeias genéticas, como ilustra a Figura 14. Dessa forma, cada linha/coluna representa uma cadeia genética responsável por uma característica do indivíduo.

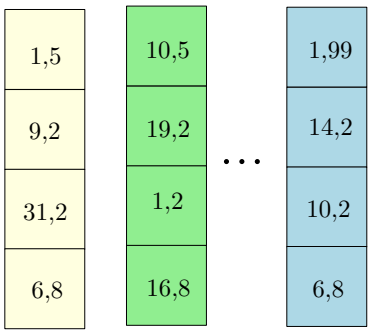


FIGURA 14: Representação de um indivíduo em AG matricial

A ideia que rege essa estruturação é a de que as características num indivíduo, em geral, são formadas por cadeias de genes, como o próprio AG tradicional sugere, entretanto, nesta proposta, estas cadeias não mais podem ser “quebradas” por um operador genético como, teoricamente, pode acontecer com os operadores existentes.

Uma das justificativas para a manutenção das cadeias genéticas é a de que a quebra destas pode ocasionar a perda de uma característica desejável, no que diz respeito à adequação da aptidão aos aos objetivos do problema, que esteja presente num indivíduo, fazendo com que a população tenha que evoluir por mais gerações para adquirir aquela característica novamente, por meio de seus operadores genéticos, havendo ainda a possibilidade de isso não acontecer. Inclusive, é possível que a população deixe de alcançar a otimalidade (ou quase otimalidade) por causa da perda dessa característica.

Outra razão para se considerar essa possibilidade é a de que os novos operadores traba-

lham com os indivíduos já na forma matricial, evitando que seja necessária qualquer subrotina que faça essa transição a cada iteração, trazendo vantagens de implementação, no que diz respeito à quantidade de processos necessários para realização da tarefa, mesmo nos casos em que a quantidade o tempo é estatisticamente equivalente à implementação tradicional.

Adaptação dos Operadores

Crossover Aritmético Matricial

Este operador é uma extensão pura do caso vetorial, não apresentando modificação alguma, além da extensão do espaço vetorial para o espaço matricial e, consequentemente, a forma de visualização de um indivíduo numa população genética.

Consiste em, dados dois indivíduos p_1 e p_2, gerar os elementos,

$$c_1 = \beta p_1 + (1-\beta) p_2 \text{ e } c_2 = \beta p_2 + (1-\beta) p_1 \text{ com } \beta \sim U(0,1).$$

Por exemplo, tomando dois indivíduos matriciais,

$$p_1 = \begin{pmatrix} 1,5 & 3,5 \\ 3,9 & 4,7 \end{pmatrix} \text{ e } p_2 = \begin{pmatrix} 5,5 & 7,1 \\ 8,3 & 1,8 \end{pmatrix},$$

fazendo $\beta = 0,7$, gera-se a descendência,

$$c_1 = \begin{pmatrix} 2,7 & 4,6 \\ 5,2 & 3,8 \end{pmatrix} \text{ e } c_2 = \begin{pmatrix} 4,3 & 6,0 \\ 7,0 & 2,7 \end{pmatrix}.$$

Vale ressaltar novamente que este operador não extrapola os valores contidos em p_1 e p_2, tendo em vista que a combinação linear gerada é sempre convexa, considerando-se os pesos β e $1-\beta$.

Crossover Heurístico Matricial

Do mesmo modo que o Crossover Aritmético, este operador também é uma extensão pura de sua versão vetorial, apresentando apenas uma modificação efetiva de seu significado e, consequentemente, de sua representação.

Este operador consiste em, considerando problemas de minimização, dados dois indivíduos p_1 e p_2, tal que a aptidão de $p_1 \leq p_2$ gerar o indivíduo,

$$c = p_2 + \beta\,(p_2 - p_1) \text{ com } \beta \sim U(0,1).$$

Aproveitando os indivíduos p_1 e p_2 do exemplo anterior e fazendo $\beta = 0,7$, este operador geraria o descendente

$$c = \begin{pmatrix} 8,3 & 9,6 \\ 11,4 & -0,2 \end{pmatrix}$$

Crossovers de um Ponto Matricial

Cruzamento Total Horizontal (CTH)

Este operador supõe que as as cadeias genéticas que definem as características de um indivíduo estão dispostas em vetores-coluna. Sendo assim, sorteia-se uma coluna como ponto de cruzamento e, a partir desta, as cadeias genéticas entre os pais são trocadas, como ilustra a Figura 15.

Em outras palavras, dados p_1 e p_2 de dimensões $m \times n$, sorteia-se um valor inteiro $r \in (1,n)$ e a partir da r-ésima coluna (cadeia genética) ocorre a troca entre os geradores, invertendo as posições em cada filho gerado.

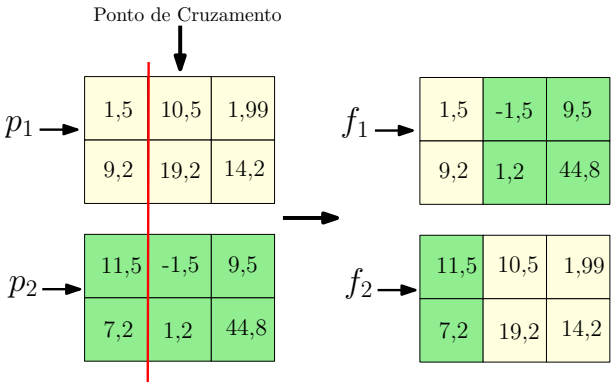


FIGURA 15: Operador CTH

Cruzamento Total Vertical (CTV)

Este operador supõe que as cadeias genéticas que definem as características de um indivíduo estão dispostas em vetores-linha. Sendo assim, sorteia-se uma linha como ponto de cruzamento e, a partir desta, as cadeias genéticas entre os pais são trocadas, como ilustra a Figura 16.

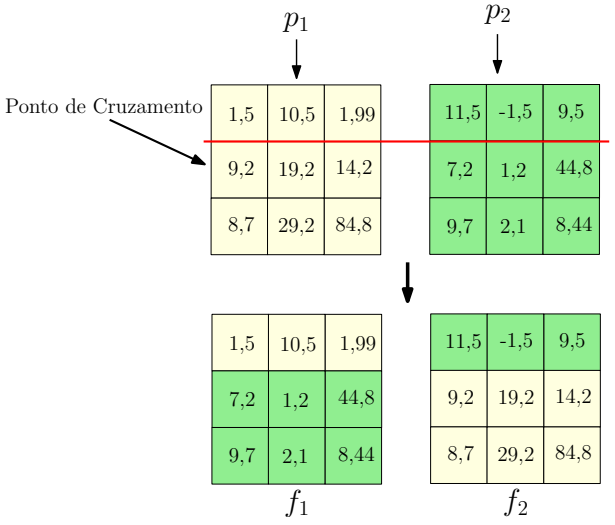


FIGURA 16: Operador CTV

De maneira similar ao operador Total Horizontal, dados p_1 e p_2 de dimensões $m \times n$, sorteia-se um inteiro $r \in (1, m)$ e a partir da r-ésima linha (cadeia genética) ocorre a troca entre os geradores, também invertendo as posições em cada filho gerado.

Cruzamento Parcial Horizontal (CPH)

Este operador é um caso particular do operador Total Horizontal e também supõe que as as cadeias genéticas que definem as características de um indivíduo estão dispostas em vetores-coluna. Entretanto, este operador troca apenas partes de cadeias genéticas e não cadeias genéticas completas. Sendo assim, sorteia-se uma coluna e uma linha da matriz como ponto de cruzamento e, a partir destas, as partes de cadeias genéticas entre os pais são trocadas, como ilustra a Figura 17.

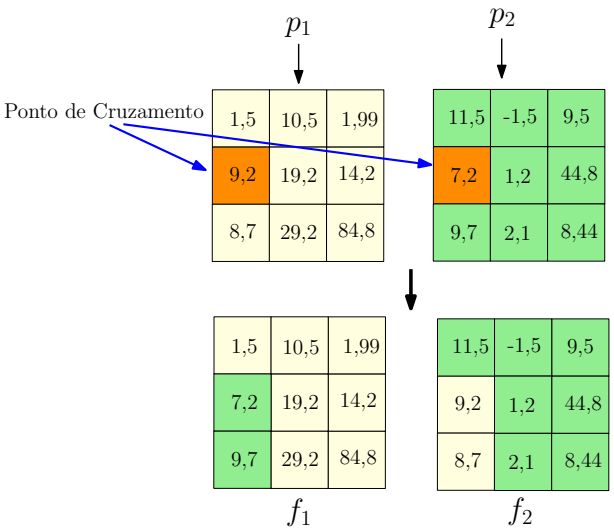


FIGURA 17: Operador CPH

Neste operador, dados p_1 e p_2 de dimensões $m \times n$, sorteia-se dois valores inteiros $r_1 \in (1,m)$ e $r_2 \in (1,n)$ e abaixo da r_1-ésima linha e à esquerda da r_2-ésima coluna (cadeia genética) ocorre a troca entre os geradores.

Cruzamento Parcial Vertical (CPV)

Este operador é um caso particular do operador Total Vertical e também supõe que as as cadeias genéticas que definem as características de um indivíduo estão dispostas em vetores-linha.

Entretanto, este operador troca apenas partes de cadeias genéticas e não cadeias genéticas completas, assim como acontece no Parcial Horizontal. Sendo assim, sorteia-se uma coluna

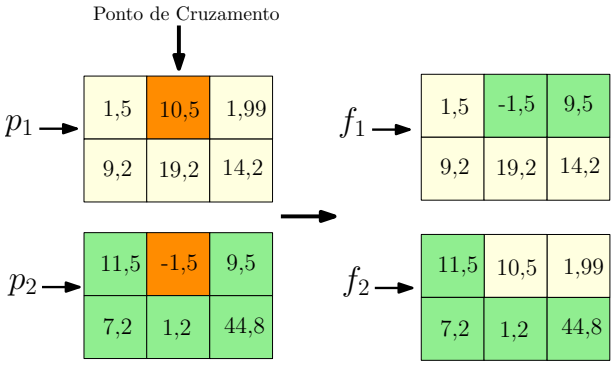


FIGURA 18: Operador CPV

e uma linha da matriz como ponto de cruzamento e, a partir destas, as partes de cadeias genéticas entre os pais são trocadas, como ilustra a Figura 18.

Neste operador, dados p_1 e p_2 de dimensões $m \times n$, sorteia-se dois valores inteiros $r_1 \in (1,m)$ e $r_2 \in (1,n)$ e na r_1-ésima linha e ao lado da r_2-ésima coluna (cadeia genética) ocorre a troca entre os geradores.

Mutação de Limite

Este operador consiste em, de maneira análoga ao caso vetorial, dado um indivíduo p de dimensões $m \times n$, sorteia-se dois valores inteiros $r_1 \in (1,m)$ e $r_2 \in (1,n)$, fazendo,

$$p(r_1,r_2) = r,$$

em que $r \in \{a,b\}$, sendo a e b os limites do intervalo factível ou do espaço de busca para r. Desse modo, por exemplo, se $a = -6$ e $b = 6$ e o indivíduo em questão for

$$p = \begin{pmatrix} 0,3 & 2,6 \\ 1,4 & -0,2 \end{pmatrix}$$

e o sorteio $(r_1,r_2) = (1;2)$ e $r = b$, teremos,

$$p^* = \begin{pmatrix} 0,3 & 6 \\ 1,4 & -0,2 \end{pmatrix}.$$

Mutação Uniforme

Este operador também é apenas uma extensão do caso vetorial para o caso matricial, uma vez que a mutação é realizada em cada gene. Desse modo, dado um indivíduo p de dimensões $m \times n$, sorteia-se dois valores inteiros $r_1 \in (1,m)$ e $r_2 \in (1,n)$, fazendo,

$$p(r_1, r_2) = r,$$

em que $r \sim U(a,b)$, sendo a e b os limites do intervalo factível ou do espaço de busca para r. Em geral, mesmo que, teoricamente, não existam a e b, costuma-se fixar, empiricamente, um intervalo para nortear a busca do AG. Desse modo, por exemplo, se

$$p = \begin{pmatrix} 8,3 & 9,6 \\ 11,4 & -0,2 \end{pmatrix}$$

e o sorteio $(r_1, r_2) = (1;2)$ e num intervalo contendo $r = 12,35$, teremos,

$$p^* = \begin{pmatrix} 8,3 & 12,35 \\ 11,4 & -0,2 \end{pmatrix}.$$

Mutação Não-uniforme Simples e Múltipla

Este operador consiste em, também de maneira análoga ao caso vetorial, dado um indivíduo p de dimensões $m \times n$, sortear dois valores inteiros $r_1 \in (1,m)$ e $r_2 \in (1,n)$, fazendo,

$$p(r_1, r_2) = \begin{cases} p(r_1, r_2) + (b - p(r_1, r_2)) f(G), & \text{se } r < 0,5 \\ p(r_1, r_2) - (p(r_1, r_2) - a) f(G) & \text{, caso contrário,} \end{cases}$$

com,

$$f(G) = \left[s \left(1 - \frac{G}{G_{\max}} \right) \right]^b,$$

em que $r, s \sim U(0,1)$, sendo a e b os limites do intervalo factível ou do espaço de busca para

r, G a geração atual e $G_{\max}$ o número máximo de gerações. Desse modo, por exemplo, se $a = -6$ e $b = 6$, o indivíduo em questão for

$$p = \begin{pmatrix} 0,3 & 2,6 \\ 1,4 & -0,2 \end{pmatrix}$$

e o sorteio $(r_1, r_2) = (1;2)$, $r = 0,7$, $s = 0,8$, $G = 50$ e $G_{\max} = 1000$, teremos,

$$p^* = \begin{pmatrix} 0,3 & 0,9 \\ 1,4 & -0,2 \end{pmatrix}.$$

A Mutação Não-uniforme múltipla consiste em aplicar uma mutação Não-uniforme em mais de um gene do indivíduo, podendo ser, inclusive, em todos os seus genes.

3.2.3 Implementação Computacional

A Implementação Computacional foi realizada com a criação de funções [1], compostas por algoritmos que executam as técnicas envolvidas, e outras rotinas computacionais para a execução de um experimento, sendo que tanto as funções quanto as rotinas do experimento em questão foram desenvolvidos em linguagem MATLAB®.

O experimento realizado tem dois objetivos: realizar uma comparação simples da acurácia entre as seis técnicas mencionadas no Capítulo 1, levando em conta apenas o percentual de classificações corretas, e uma outra comparação mais detalhada, levando em conta também o tempo computacional e quantidade de iterações da classificação de padrões via RBFNN treinada com AG tradicional e com a adaptação proposta deste trabalho.

Configurações da FDLF

Foi implementada a FDLF - Função Discriminante Linear de Fisher em linguagem MATLAB ®, para a classificação de padrões, na qual é possível escolher entre duas configurações no que diz respeito aos percentuais de exemplos escolhidos para treinamento e testes no banco

[1] rotinas computacionais

de dados a ser classificado.

Numa das configurações, o algoritmo utiliza 80% dos exemplos para treinamento e 20% para testes e, na segunda opção, utiliza 60% para treinamento, então, dos 40% restantes, metade dos exemplos são descartados aleatoriamente e os restantes são utilizados para o conjunto de testes.

O algoritmo implementado leva em consideração todas as funções canônicas de Fisher que possuam autovalores associados significativamente maiores do que zero (maiores que 10^{-5}) para realizar a classificação, pois segundo Johnson e Wichern (2002), este é um critério que pode levar a melhores soluções.

Configurações da RBFNN

Foi implementada uma RBFNN na qual é possível escolher a quantidade de neurônios da camada intermediária, um método de cálculo da matriz de pesos da camada intermediária, dentre os cinco implementados, a saber:

- **Método da Pseudo-inversão**: que consiste em calcular a pseudo-inversa da matriz com os valores de ativação dos neurônios da camada intermediária, conforme descrito na Seção 2.1.1, para fins de sistematização da apresentação, este método será referenciado como *pinv*;
- **PSO (Melhor Global)**: que consiste em utilizar o algoritmo descrito na Seção 2.3.1, criando a metodologia híbrida descrita na Seção 3.2.1, este método será referenciado como *psogbest*;
- **PSO (Melhor Local)**: de maneira similar ao item anterior, apenas com a mudança do método, esta abordagem consiste em utilizar o algoritmo descrito também na Seção 2.3.1, criando a metodologia híbrida descrita na Seção 3.2.1, este método será referenciado como *psolbest*;
- **AG Tradicional**: que consiste em utilizar o algoritmo descrito na Seção 2.2.1, com os operadores de Michalewicz, Logan e Swaminathan (1994), descrito na Seção 2.2.1 cri-

ando a metodologia híbrida descrita na Seção 3.2.1, este método será referenciado como *agrbf* ou;

- **AG Modificado**: que consiste em utilizar o algoritmo descrito na Seção 2.2.1, com os operadores genéticos descritos na Seção 3.2, criando a metodologia híbrida descrita na Seção 3.2.1, com a diferença de não precisar de uma subrotina para converter os indivíduos matriciais em vetores, este método será referenciado como *agrbf_mod*.

Na RBFNN implementada, há uma rotina que separa o banco de dados a ser classificado em conjuntos de treinamento, validação e testes, nas proporções mais encontradas na literatura, ou seja, de 60%–20%–20% dos exemplos do banco de dados, respectivamente.

Vale salientar que o conjunto de validação serve apenas para orientar o treinamento da rede neural no sentido de avaliar sua capacidade de generalização durante o treinamento. A única informação do conjunto de validação utilizada no treinamento é o percentual de classificações corretas realizadas pela rede neste conjunto, para fins de avaliação de um dos critérios de parada.

A estratégia comumente adotada é a seguinte: enquanto o percentual de classificações corretas no conjunto de validação estiver aumentando a rede deve continuar seu treinamento, após haver quedas consecutivas é recomendado parar o treinamento e escolher a melhor configuração encontrada até então.

Já do conjunto de testes, que mede o desempenho efetivamente alcançado, nenhuma informação é utilizada durante o treinamento da rede em questão. Este conjunto serve somente para a avaliação final da técnica aplicada e informa uma probabilidade de acerto do modelo construído.

A função de ativação em todos os neurônios da camada intermediária é a Gaussiana, descrita na Seção 2.1.1 com raios de abertura (*spreads*) definidos sempre pelo procedimento heurístico descrito na Seção 2.1.1. O método de seleção dos centros dos neurônios da camada intermediária é a Seleção Auto-organizada de Centros, descrito na Seção 2.1.1, com taxa de aprendizagem $\eta = 0,3$ em todos os casos.

De acordo com a literatura, como por exemplo é possível encontrar em Silva, Spatti e Flauzino (2010) e Haykin (2001), um dos parâmetros mais importantes para uma rede neural,

inclusive as RBFNN é a quantidade de neurônios em suas camadas, no caso específico desta pesquisa, a quantidade de neurônios da camada intermediária. O limite aqui estabelecido para o experimento descrito a seguir será o de testar de dois a 20 neurônios, com incremento de dois, ou seja, dez arquiteturas diferentes de RBFNN.

Configuração das Técnicas Utilizadas no Treinamento da RBFNN

Configuração do AG

A configuração de um AG também não possui um método consolidado, variando de acordo com cada aplicação. Por esse motivo, faz-se necessária a realização de alguns testes iniciais para que se possa observar a influência de cada parâmetro no desempenho do algoritmo.

Nesta pesquisa, a variante de AG utilizada, tanto em *agrbf* quanto em *agrbf_mod*, foi aquela conhecida como *steady state*, numa tradução livre “regime constante”, que tem como principal característica eliminar os indivíduos com pior adequação do *fitness* ao problema em cada geração t, mantendo a população com o tamanho constante, não fazendo distinção se os indivíduos em questão pertencem à geração t ou se fariam parte da descendência para a geração $t+1$.

Foram considerados três tamanhos de população, $TamPop = \{150, 500, 1000\}$, no experimento, com a inteção de cobrir um tamanho de população que possa ser considerado “pequeno”, um “médio” e um “grande”.

Com relação à seleção de indivíduos para gerar descendência (reprodução) e seleção dos indivíduos para sofrerem mutação, em ambos os casos, nenhuma heurística que levasse em consideração o *fitness* foi utilizada. Os indivíduos são sorteados aleatoriamente dentre aqueles presentes na população. A cada geração, tanto do AG clássico quanto da modificação proposta, caso ocorra cruzamento, apenas 20% da população é sorteada para gerar descendência e, dos filhos gerados, apenas a metade destes são sorteados para sofrer mutação.

Com relação aos operadores de cruzamento e mutação, para *agrbf* foram utilizados aqueles descritos na Seção 2.2.1 e para *agrbf_mod* foram aqueles descritos na Seção 3.2. Em

ambos os casos, para os operadores que possuem restrição no intervalo dos números gerados, ou seja,a Mutação Uniforme, a Mutação de Limite e Mutação Não-uniforme (simples e múltipla), tendo em vista que o melhor desempenho foi obtido gerando-se números aleatórios normalmente distribuídos com média zero e desvio-padrão um, o intervalo adotado em todas as situações foi $[-6,6]$, por conter 99,9997% dos dados de uma distribuição normal $N \sim (0,1)$.

Para a probabilidade de ocorrência de cruzamento (*pc*) foram considerados cinco possibilidades, fazendo $pc = \{0,5; 0,55; 0,6; 0,65; 0,7\}$. Já para a probabilidade de ocorrência de mutação (*pm*), foram considerados apenas três possibilidades, fazendo $pm = \{0,01; 0,05; 0,1\}$. Esses valores foram considerados tanto para *agrbf* quanto para *agrbf_mod*.

A quantidade máxima de gerações (iterações) foi de 2000, sendo este um dos critérios de parada. O segundo critério de parada foi: ao atingir mais de 30 gerações sem diminuição no erro atual, o programa exibe a melhor resposta encontrada até o momento.

Configuração do PSO

Alguns dos parâmetros que compõem a configuração de *psogbest* e *psolbest* são comuns àqueles escolhidos para *agrbf* e *agrbf_mod*, a saber:

- as possibilidades consideradas para o tamanho do enxame (*TamEnx*) são as mesmas daquelas consideradas para *TamPop*, ou seja, tem-se os seguintes valores $TamEnx = \{150, 500, 1000\}$;
- os dois critérios de parada: “quantidade máxima de iterações” e “quantidade de iterações sem melhorias” também são 2000 e 30, respectivamente;
- o intervalo para geração do enxame também é $[-6,6]$, com base empírica.

O último parâmetro em comum entre *psogbest* e *psolbest* é o peso de inércia ϕ, que é atualizado de acordo com,

$$\phi = 0,9^{\text{Iter}} \tag{3.1}$$

em que *Iter* é a iteração atual.

Finalmente, para *psolbest*, foram consideradas cinco possibilidades de tamanho de vizinhança das atualizações (*TamViz*), fazendo $TamViz = \{1,2,3,4,5\}$. Vale ressaltar que, nessa pesquisa, foi considerada a vizinhança baseada em índices, por ser mais rápida e a mais utilizada na literatura, conforme afirma Engelbrecht (2007).

3.2.4 **Descrição do Experimento**

Antes da realização do experimento propriamente dito, aplicou-se a Análise de Variância Multivariada (MANOVA) a fim de verificar se havia diferença entre as médias dos grupos em cada banco de dados, para realização do experimento em questão, foram utilizados apenas os bancos de dados nos quais havia essa diferença, buscando garantir a fidedignidade dos resultados encontrados na classificação.

Foi criada uma rotina para a execução do experimento, conforme é possível visualizar na Figura 19. São obtidos os valores das variáveis de interesse, a saber: PCC - Percentual de Classificações Corretas do conjunto de testes, TE - Tempo de Execução do algoritmo e QI - Quantidade de Iterações, sendo que esta última não foi medida no treinamento da RBFNN via método da pseudo-inversão nem na FDL de Fisher, pois ambos não são processos iterativos.

Uma observação a ser feita é que a cada rodada, em todas as técnicas, os padrões que fariam parte dos conjuntos de treinamento, validação (quando aplicável) e teste foram sempre sorteados aleatoriamente. Essa medida teve como objetivo garantir que não fosse utilizado sempre o mesmo conjunto para treinamento.

Após a realização do experimento, verificou-se o melhor resultado obtido em cada técnica para cada um dos seis bancos de dados, a fim de verificar qual técnica foi capaz de realizar melhor a tarefa de classificação em cada caso.

Então, para o treinamento da RBFNN via AG tradicional e com a adaptação proposta, foram calculados a média e o desvio-padrão, de cada variável de interesse, para cada terna (banco de dados, técnica, configuração) e considerou-se como melhor configuração aquela que apresentou a maior média da variável PCC.

Posteriormente, para cada banco de dados, considerando-se as melhores configurações

```
ALGORITMO: ROTINA DO EXPERIMENTO
  Para (cada banco de dados)
    Para (cada técnica)
      Para (cada configuração)
        rodadas = 0
        Enquanto (rodadas < 40)
          Aplicar a técnica e obter
            PCC - Percentual de Classificações Corretas do conjunto de testes
            TE - Tempo de Execução
            QI - Quantidade de Iterações (quando aplicável)
            rodadas = rodadas + 1
        fim Enquanto
      fim Para
    fim Para
  fim Para
```

FIGURA 19: Algoritmo do experimento

de cada uma das duas técnicas, realizou-se um teste de *Mann-Whitney*, ao nível de 95% de confiança, para verificar se houve diferença significativa entre as médias de PCC, TE e QI, qual foi a maior média, no caso do PCC, e a menor média, nos casos de TE e QI.

4 RESULTADOS E DISCUSSÃO

4.1 ANÁLISE DE VARIÂNCIA MULTIVARIADA

Conforme mencionado na Seção 3.2.4, antes da realização do experimento foi realizada a Análise de Variância Multivariada (MANOVA) a fim de verificar se havia diferença significa entre as médias dos grupos em cada banco de dados testado. Os resultados desse teste econtram-se na Tabela 5.

TABELA 5: Resultados da Aplicação da MANOVA

Banco de Dados	**Valores *p***
cancer	$< 10^{-263}$
cmc	$< 10^{-17}$
careval	$< 10^{-3}$
hepatitis	$< 10^{-9}$
iris	$< 10^{-6}$
pageblocks	$< 10^{-199}$

Observando os valores p da Tabela 5 é possível afirmar que, para os seis bancos de dados considerados no experimento, existe diferença significativa entre as médias dos grupos, ao nível de 5% de significância.

4.2 ANÁLISE DA MELHOR CONFIGURAÇÃO ISOLADAMENTE

A Tabela 6 traz a melhor configuração encontrada, isoladamente, para cada técnica durante o experimento:

- *fisher*: há apenas duas configurações possíveis: $[0,6;0,20;0,20]$ significa que foram utilizados 60% dos exemplos no treinamento, 20% foram descartados (aleatoriamente) e os 20% restantes foram utilizados para o teste, já $[0,8;0,20]$ significa que 80% foram utilizados para treinamento e 20% para testes.

- *pinv*: o único parâmetro variado no experimento foi a quantidade de neurônios na camada intermediária, resultando em 10 configurações possíveis. Portanto, $[20]$, por exemplo, significa que, para aquele banco de dados, a melhor configuração foi com 20 neurônios na camada escondida;

- *psogbest*: houve variação da quantidade de neurônios na camada intermediária e do tamanho do enxame, resultando em 30 configurações possíveis. Desse modo, $[2;150]$, por exemplo, é uma configuração com dois neurônios na camada intermediária e 150 partículas no enxame;

TABELA 6: Melhores configurações das técnicas

Banco de Dados	Técnica	Melhor Configuração	PCC	κ
cancer	fisher	$[0,6;0,20;0,20]$	99%	$0,95$
	pinv	$[14]$	98%	$0,95$
	psogbest	$[4;1000]$	99%	$0,95$
	psolbest	$[6;1000;4]$	99%	$0,95$
	agrbf	$[14;150;0,50;0,10]$	100%	$0,95$
	agrbf_mod	$[2;500;0,70;0,05]$	100%	$0,95$
cmc	fisher	$[0,8;0,20]$	49%	$0,96$
	pinv	$[14]$	45%	$0,97$
	psogbest	$[2;150]$	51%	$0,96$
	psolbest	$[12;1000;2]$	54%	$0,96$
	agrbf	$[14;150;0,70;0,05]$	50%	$0,96$
	agrbf_mod	$[20;1000;0,50;0,10]$	49%	$0,96$
careval	fisher	$[0,6;0,20;0,20]$	74%	$0,93$
	pinv	$[16]$	81%	$0,94$
	psogbest	$[14;150]$	74%	$0,96$
	psolbest	$[8;150;2]$	73%	$0,96$
	agrbf	$[12;1000;0,70;0,05]$	76%	$0,95$
	agrbf_mod	$[8;1000;0,70;0,10]$	79%	$0,95$
hepatitis	fisher	$[0,8;0,20]$	77%	$0,88$
	pinv	$[14]$	93%	$0,89$
	psogbest	$[18;500]$	97%	$0,89$
	psolbest	$[6;150;3]$	97%	$0,89$
	agrbf	$[2;150;0,50;0,05]$	93%	$0,89$
	agrbf_mod	$[4;150;0,60;0,10]$	97%	$0,89$
iris	fisher	$[0,6;0,20;0,20]$	97%	$0,18$
	pinv	$[8]$	100%	$0,17$
	psogbest	$[4;1000]$	97%	$0,21$
	psolbest	$[2;500;2]$	97%	$0,21$
	agrbf	$[2;1000;0,50;0,05]$	100%	$0,19$
	agrbf_mod	$[2;150;0,60;0,05]$	100%	$0,19$

Continua...

Continuação...

Banco de Dados	Técnica	Melhor Configuração	PCC	κ
pageblocks	fisher	$[0,6;0,20;0,20]$	81%	$0,98$
	pinv	$[16]$	92%	$0,98$
	psogbest	$[2;500]$	91%	$0,98$
	psolbest	$[12;1000;3]$	93%	$0,98$
	agrbf	$[18;1000;0,60;0,10]$	92%	$0,98$
	agrbf_mod	$[10;150;0,70;0,10]$	93%	$0,98$

- *psolbest*: houve variação da quantidade de neurônios na camada intermediária, tamanho do enxame e do tamanho da vizinhança considerada para a atualização da velocidade, resultando em 150 configurações possíveis. Desse modo, o vetor $[2;150;2]$, por exemplo, é uma configuração com dois neurônios na camada intermediária, 150 partículas no enxame e tamanho de vizinhança igual a dois;

- *agrbf* e *agrbf_mod*: no caso das duas abordagens via AG, foram variados quatro parâmetros: quantidade de neurônios na camada intermediária, tamanho da população inicial, probabilidade de ocorrência dos operadores de cruzamento e de mutação, totalizando 450 configurações possíveis em cada uma das duas técnicas. Então, $[2;150;0,65;0,01]$, por exemplo, é uma configuração com dois neurônios na camada escondida, população inicial de 150 indivíduos, probabilidade de cruzamento igual a $0,65$ e mutação igual a $0,01$.

É possível visualizar na Tabela 6 que as melhores configurações não seguiram uma tendência entre os seis bancos de dados testados, em nenhuma das técnicas testadas, no que diz respeito aos valores dos parâmetros que constituem cada configuração.

Isso ressalta duas características comuns aos métodos para a classificação de padrões, sejam eles estatísticos ou matemáticos: a sensibilidade e adaptabilidade às características de cada problema, no caso específico dessa pesquisa, cada banco de dados.

A Tabela 6 apresenta o melhor PCC para cada técnica e banco de dados e técnica testados. Nota-se que houve uma variação significativa na acurácia (PCC) de todas as técnicas testadas, havendo casos em que nenhuma técnica alcançou 60% de classificações corretas no conjunto de testes, como é o caso de *cmc*, e casos em que todas as técnicas classificaram mais de 80% do conjunto de testes corretamente, como, por exemplo, é o caso dos bancos de dados *cancer* e *iris*.

Embora tenham sido testadas variadas configurações, é comum essa variação acontecer, uma vez que os bancos de dados em questão possuem alguma diversidade de complexidades na realização da classificação de padrões, isto é, cada problema tem especificidades inerentes à área a que este pertence e um experimento com os limites estabelecidos nessa pesquisa, não foram suficientes para gerar resultados que possam ser considerados passíveis de implementação comercial.

Para o banco de dados *cancer*, todos os métodos avaliados: *fisher*, *pinv*, *psogbest*, *psolbest*, *agrbf* e *agrbf_mod* apresentaram valores considerados excelentes pela literatura, tanto para o PCC quanto para o índice *kappa* (κ), este valendo exatamente 0,95 em todos os casos e aquele no intervalo 98–100%.

No banco de dados *cmc*, apesar deste apresentar diferença estatisticamente significativa entre as médias dos grupos, não houve um desempenho tradicionalmente considerado aceitável pela literatura no que diz respeito ao PCC. O índice κ, que foi excelente em todos os casos, variou no intervalo 0,96–0,97 e o PCC no intervalo 49–54%.

Para *careval*, todos os métodos avaliados apresentaram valores acima de 70% para o PCC, o qual variou no intervalo 98–100%. Já o índice *kappa* (κ), apresentou valores próximos de um, variando no intervalo 0,93–0,96.

Em *hepatitis*, todos os métodos avaliados, exceto *fisher*, apresentaram valores acima de 90% para o PCC. Com relação ao índice *kappa* (κ), todos apresentaram valores próximos a 0,9.

Já em *iris*, a exemplo do banco de dados *cancer*, todos os métodos avaliados: *fisher*, *pinv*, *psogbest*, *psolbest*, *agrbf* e *agrbf_mod* apresentaram valores considerados excelentes pela literatura, no que diz respeito ao PCC. Entretanto, no que concerne ao índice *kappa* (κ), os valores obtidos variaram no intervalo 0,17–0,21, valores considerados baixos pela maioria dos pesquisadores.

Finalmente, para *pageblocks*, com exceção novamente de *fisher*, todas as técnicas testadas apresentaram um PCC maior do que 90%. Um fato a se mencionar é o de que o valor de *kappa* foi exatamente o mesmo em todos os casos: 0,98. Esse valor de *kappa* é considerado excelente pela literatura.

4.3 COMPARAÇÃO ENTRE *AGRBF* E *AGRBF_MOD*

A partir do experimento realizado, calculou-se a média e o desvio-padrão das variáveis de interesse, PCC, TE e QI considerando 40 rodadas para *agrbf* e *agrbf_mod* a fim de realizar uma comparação estatística dessas variáveis de interesse.

A Tabela 7 mostra as melhores configurações para *agrbf* e *agrbf_mod* considerando-se a média das 40 rodadas. As configurações aqui apresentadas podem ser diferentes daquelas encontradas na Tabela 6, pois levam em consideração a média em substituição à melhor configuração obtida isoladamente.

TABELA 7: Melhores configurações para *agrbf* e *agrbf_mod*

Banco de Dados	Técnica	Melhor Configuração
cancer	agrbf	$[14; 500; 0,70; 0,05]$
	agrbf_mod	$[4; 150; 0,60; 0,10]$
cmc	agrbf	$[14; 150; 0,60; 0,05]$
	agrbf_mod	$[8; 500; 0,60; 0,10]$
careval	agrbf	$[6; 500; 0,60; 0,10]$
	agrbf_mod	$[16; 500; 0,70; 0,10]$
hepatitis	agrbf	$[6; 500; 0,70; 0,05]$
	agrbf_mod	$[16; 150; 0,60; 0,10]$
iris	agrbf	$[6; 500; 0,60; 0,05]$
	agrbf_mod	$[4; 500; 0,70; 0,10]$
pageblocks	agrbf	$[4; 150; 0,70; 0,05]$
	agrbf_mod	$[6; 500; 0,60; 0,10]$

Os valores-médios e o desvio padrão para PCC, TE e QI estão apresentados na Tabela 8, onde também é possível encontrar o valor *p* do teste de *Mann-Whitney* para as mesmas três variáveis de interesse numa terna. Isso quer dizer que, por exemplo, $(0,04;\ 10^{-7};\ 0,02)$ significa que os valores *p* do teste para PCC, TE e QI são, respectivamente $0,04$, 10^{-7} e $0,02$. Vale lembrar que o nível de significância do teste é de 5%.

TABELA 8: PCC Médio, TE Médio e QI Médio por Banco de Dados

Banco de Dados	Técnica	PCC (%)		TE (s)		QI		Valores *p*
		$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	
cancer	agrbf	$95,7$	$1,4$	$11,7$	$2,4$	$63,7$	$12,1$	$(0,65;\ 10^{-7};\ 0,00)$
	agrbf_mod	$95,5$	$1,5$	$3,2$	$0,5$	$52,1$	$8,2$	

Continua...

Continuação...

Banco de Dados	Técnica	PCC (%)		TE (s)		QI		Valores *p*
		$\bar{x}$	s	$\bar{x}$	s	$\bar{x}$	s	
cmc	agrbf	$37,1$	$7,5$	$12,5$	$1,6$	60	$4,9$	$(0,19;\ 10^{-3};\ 10^{-5})$
	agrbf_mod	$39,6$	$7,4$	$15,1$	$2,3$	$51,5$	$7,7$	
careval	agrbf	$70,9$	$1,9$	$13,6$	3	$63,3$	$11,8$	$(0,04;\ 10^{-7};\ 0,02)$
	agrbf_mod	$72,9$	$3,2$	$30,6$	$8,1$	$76,1$	$12,2$	
hepatitis	agrbf	$78,9$	$7,7$	$4,1$	$0,3$	32	0	$(0,78;\ 10^{-7};\ 0,00)$
	agrbf_mod	$79,7$	$4,9$	$1,5$	$0,2$	33	2	
iris	agrbf	$70,8$	$25,3$	$10,6$	$2,7$	$75,7$	$19,3$	$(0,29;\ 10^{-3};\ 0,27)$
	agrbf_mod	$79,7$	$22,9$	$15,7$	$3,9$	$69,3$	$17,3$	
pageblocks	agrbf	$89,9$	$1,1$	$14,8$	$1,7$	$81,9$	$6,7$	$(0,45;\ 10^{-7};\ 0,03)$
	agrbf_mod	$90,3$	$0,8$	$76,4$	$10,5$	$89,2$	$11,6$	

Para *cancer*, não houve diferença entre as médias de PCC, pois $p = 0,65$. Entretanto, para as outras duas variáveis de interesse, é possível afirmar que, ao nível de 5% de significância, houve diferença entre *agrbf* e *agrbf_mod*, sendo que este apresentou menor tempo de execução e menor quantidade de iterações nesse banco de dados.

No caso de *cmc*, o PCC médio ficou abaixo dos 60% para ambos *agrbf* e *agrbf_mod*, não havendo diferença entre as médias do PCC. O TE médio foi menor em *agrbf* e a média de QI foi menor para *agrbf_mod*.

Para o banco de dados *careval*, embora *agrbf_mod* tenha apresentado ambos TE e QI maiores (piores) do que *agrbf*, apresentou um PCC médio superior à implementação tradicional do treinamento de uma RBFNN via AG, considerando-se o nível de significância.

Em *hepatitis*, não houve diferença significativa para a variável PCC médio, pois $p = 0,78$, mas, no que diz respeito ao TE médio e QI médio, ambos apresentaram diferença significativa, $p = 10^{-7}$ e $p = 0,00$, sendo o menor TE médio obtido com *agrbf_mod* e o menor QI médio com *agrbf*. Uma observação a se fazer é a de que o desvio-padrão de QI para *agrbf*, considerando-se as 40 rodadas do experimento, foi zero.

Para *iris*, não houve diferença para o PCC médio nem QI médio de *agrbf* e *agrbf_mod*, tendo em vista os valores *p*. Entretanto, para o TE médio, a técnica *agrbf* apresentou a menor média, considerando a significância do teste.

Finalmente, para *pageblocks*, houve diferença estatisticamente significativa apenas no TE médio, sendo que *agrbf* apresentou um valor que pode ser considerado até mesmo visualmente menor do que *agrbf_mod*, e o teste estatístico ratifica essa primeira impressão visual.

5 CONSIDERAÇÕES FINAIS

5.1 CONCLUSÕES

Nesta pesquisa, foram testadas seis abordagens para o problema da classificação de padrões. Uma abordagem estatística, representada pela Função Discriminante Linear de Fisher e cinco abordagens via RBFNN, que se distinguem pelo método de cálculo da matriz de pesos da camada intermediária, das quais cinco já existiam na literatura, aqui denominadas: *pinv*, que é a tradicionalmente utilizada, *psogbest* e *psolbest* baseadas no algoritmo PSO, *agrbf* e *agrbf_mod*, ambas baseadas em AG, sendo que esta última foi a proposta nesse trabalho.

Para avaliar as técnicas elencadas anteriormente, foram utilizados seis bancos de dados do repositório de dados para aprendizado de máquina da UCI *Machine Learning Dataset* (Banco de dados para Aprendizado de Máquina da UCI) Frank e Asuncion (2010).

Treinar uma RBFNN consiste basicamente em definir: o posicionamento dos centros dos neurônios, o grau de abertura do raio das funções gaussianas de ativação e a matriz de pesos entre a camada escondida e a camada de saída. Ainda um parâmetro que é obtido empiricamente, mas não é menos importante, é a quantidade de neurônios na camada escondida.

A aplicação da Análise de Variância Multivariada (MANOVA) foi relevante para a pesquisa, pois garantiu a possibilidade da realização da classificação de padrões nos bancos de dados considerados. É uma "garantia" estatística de as médias dos grupos eram diferentes e, consequentemente, os grupos eram separáveis.

Analisando a melhor configuração isoladamente, faz-se necessário elencar:

- **cancer**: todas as técnicas testadas apresentaram percentuais de classificação acima de 95% e coeficientes *kappa* iguais a $0,95$, significando que a qualidade do poder de classificação apresentado pelas técnicas é estatisticamente consistente. Mais especificamente no caso das duas implementações do treinamento da RBFNN via AG, o per-

centual de classificação foi de 100%, o que pode ser considerado um ótimo global nessa tarefa, sendo possível ainda que haja outros ótimos globais;

- **cmc**: apesar de todas as técnicas aplicadas em *cmc* terem apresentado valores de *kappa* acima de $0,95$ em todos os casos, o percentual de classificações corretas não excedeu 60% em nenhum caso, sendo o maior deles na implementação do treinamento de uma RBFNN via PSO com o algoritmo *lbest*, que alcançou 54% de classificações corretas;

- **careval**: todas as técnicas testadas apresentaram percentuais de classificação acima de 70% e coeficientes *kappa* acima de $0,9$, significando que a qualidade do poder de classificação apresentado pelas técnicas é estatisticamente consistente, apesar do PCC não ter ultrapassado os 80% em nenhum caso;

- **hepatitis**: todas as técnicas testadas, exceto *fisher*, apresentaram percentuais de classificação acima de 90% e coeficientes *kappa* próximos a $0,9$, significando que a qualidade do poder de classificação apresentado pelas técnicas é estatisticamente consistente;

- **iris**: todas as técnicas testadas apresentaram percentuais de classificação acima de 90% e coeficientes *kappa* abaixo de $0,2$, significando que a qualidade do poder de classificação apresentado pelas técnicas é estatisticamente inconsistente. Um fato notório, uma vez que em três casos houve um acerto de 100% na classificação dos padrões;

- **pageblocks**: todas as técnicas testadas apresentaram percentuais de classificação acima de 90% e coeficientes *kappa* iguais a $0,98$, significando que a qualidade do poder de classificação apresentado pelas técnicas é estatisticamente consistente, apesar do PCC não ter ultrapassado os 80% em nenhum caso.

Com base nessas evidências, é possível concluir que, no que diz respeito à análise da melhor configuração isolada encontrada, com exceção do banco de dados *cmc*, em todos os outro cinco bancos de dados testados houve percentuais de classificação corretas dentro do que é considerado satisfatório pela literatura com todas as técnicas. Contudo, no banco de dados *iris* os valores de *kappa* apresentados não foram satisfatórios e, por esse motivo, faz-se necessária uma investigação do motivo da ocorrência desse fenômeno.

No que concerne à comparação entre *agrbf* e *agrbf_mod*, é preciso mencionar que:

- **cancer**: não houve diferença estatisticamente significativa na média do PCC, porém o TE médio e o QI médio de *agrbf_mod* foram menores do que o de *agrbf*;
- **cmc**: não houve diferença estatisticamente significativa na média do PCC, que foi baixo com ambas as técnicas, porém o TE médio de *agrbf* foi menor do que o apresentado por *agrbf_mod* e o QI médio de *agrbf_mod* foi menor do que o de *agrbf*;
- **careval**: houve diferença estatisticamente significativa na média do PCC, sendo que a média apresentada por *agrbf_mod* foi maior do que a de *agrbf*;
- **hepatitis**: não houve diferença estatisticamente significativa na média do PCC, porém o TE médio de *agrbf_mod* foi menor do que o de *agrbf*;
- **iris**: não houve diferença estatisticamente significativa na média do PCC, porém o TE médio de *agrbf* foi menor do que o apresentado por *agrbf_mod* e o QI médio de *agrbf_mod* foi menor do que o de *agrbf*;
- **pageblocks**: não houve diferença estatisticamente significativa na média do PCC, porém o TE médio e o QI médio de *agrbf* foram menores do que o de *agrbf_mod*.

Com base nessas evidências, é possível concluir que a adaptação proposta foi capaz de gerar médias do PCC estatisticamente equivalentes ou maiores em relação ao treinamento de uma RBFNN via AG tradicional em todos os conjuntos testados.

A hipótese de que a proposta poderia fazer com que a quantidade de iterações fosse menor no treinamento de uma RBFNN via AG ocorreu em metade dos bancos de dados testados, entretanto, ainda não é possível afirmar que essa redução vá ocorrer na maioria dos casos nem que essa redução esteja atrelada à redução no tempo de execução do algoritmo.

Estas duas análises, em conjunto, configuram uma evidência empírica de que o treinamento de uma RBFNN utilizando como mecanismo de cálculo da matriz de pesos uma Metaheurística de busca baseada em população (PSO e AG) são uma alternativa viável para a classificação de padrões e, também, a adaptação proposta, por ter se mostrado capaz de gerar resultados tão satisfatórios quanto as configurações já existentes, é uma alternativa considerável.

5.2 SUGESTÕES PARA TRABALHOS FUTUROS

A seguir são apresentadas algumas sugestões para trabalhos futuros que complementariam as ideias expostas nessa pesquisa ou que podem vir a melhorar os resultados obtidos pelas técnicas apresentadas:

- aplicar outras variantes dos AG no treinamento das RBFNN, levando em consideração melhorias realizadas na seleção para a reprodução, seleção da nova população ou novos operadores;
- aplicar outras variantes do algoritmo PSO no treinamento das RBFNN, levando em consideração melhorias na atualização da velocidade e a criação de grupos de partículas;
- implementar o cálculo da matriz de pesos de uma RBFNN via evolução diferencial, a fim de avaliar o desempenho dessa técnica para esta tarefa;
- criar outras metodologias híbridas, como a combinação de preditores a fim de avaliar seus desempenhos nos bancos de dados utilizados nessa pesquisa e em outros bancos de dados.

REFERÊNCIAS

ABRAHAM, A.; GUO, H.; LIU, H. Swarm intelligence: foundations, perspectives and applications. In: ____. New York, USA: Springer Berlin Heidelberg, 2006. (Studies in Computational Intelligence, v. 26), cap. Methodologies based on particle swarm intelligence.

BERG, F. V. den. **An Analysis of Particle Swarm Optimizers**. Tese (Doutorado) — Departamento de Ciência da Computação, Universidade de Pretória, África do Sul, 2002.

BONABEAU, E.; CORNE, D.; POLI, R. Swarm intelligence: the state of the art special issue of natural computing. **Natural Computing**, Springer Netherlands, v. 9, p. 655–657, 2010.

BRAGA, A. d. P.; LUDERMIR, A. P. d. L. d. C. e. T. B. **Redes neurais artificiais: teoria e aplicações**. Rio de Janeiro: LTC, 2007.

CARLETTA, J. Assessing agreement on classification tasks: the kappa statistic. **Squibs and Discussions**, v. 2, n. 22, p. 249–254, 1996.

CHANGBING, L.; WEI, H. Application of genetic algorithm-rbf neural network in water environment risk prediction. **2nd International Conference on Computer Engineering and Technology 2010**, p. 239–242, 2010.

CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. **Evolutionary Computation, IEEE Transactions on**, v. 6, n. 1, p. 58–73, 2002.

ENGELBRECHT, A. P. **Computational intelligence: an introduction**. Chichester, England: John Wiley & Sons Ltd, 2007.

FAUSETT, L. **Fundamentals of neural networks: architectures, algorithms, and applications**. Upper Saddle River, USA: Prentice Hall, 1994.

FIGUEIREDO, G. C.; VIEIRA, C. A. O. Estudo do comportamento dos índices de exatidão global, kappa e tau, comumente usados para avaliar a classificação de imagens do sensoriamento remoto. **XIII Simpósio Brasileiro de Sensoriamento Remoto**, p. 5755–5762, 2007.

FONSECA, J. S. **Curso de estatística**. 6. ed. São Paulo: Atlas, 1996.

FRANK, A.; ASUNCION, A. **UCI machine learning repository**. 2010.

GOLDBERG, D. E. **Genetic algorithms in search, optimization, and machine learning**. New Jersey, USA: Addison Wesley Longman, 1989.

HAYKIN, S. **Redes neurais - princípios e práticas**. São Paulo: Bookman, 2001.

HOFFMANN, F.; MEDINA, D.; WOLISZ, A. Optimization of routing and gateway allocation in aeronautical ad hoc networks using genetic algorithms. In: **Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International**. [S.l.: s.n.], 2011. p. 1391–1396.

HOLLAND, J. **Adaption in Natural and Artificial Systems**. Cambridge, England: MIT Press, 1975.

JAREANPON, C.; PENSUWON, W.; FRANK, R. J.; DAVEY, N. An adaptive rbf network optimised using a genetic algorithm applied to rainfall forecasting. **International Simposium on Communications and Information Technologies 2004**, Sapporo, Japan, p. 1005–1010, 2004.

JOHNSON, R. A.; WICHERN, D. W. **Applied multivariate statistical analysis**. New Jersey, USA: Prentice Hall, 2002.

KENNEDY, J. The Behavior of Particles. In: **Proceedings of the 7th International Conference on Evolutionary Programming VII**. London, UK: Springer-Verlag, 1998. p. 581–589.

KENNEDY, J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In: **Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on**. [S.l.: s.n.], 1999. v. 3.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: **IEEE International Conference on Neural Networks Proceedings**. [S.l.: s.n.], 1995. v. 4, p. 1942–1948.

KOHONEN, T. The self-organizing map. **Neurocomputing**, v. 21, n. 1–3, p. 1–6, 1998.

KRZANOWSKI, W. J. **Principles of multivariate analysis: a user's perspective**. New York, USA: Oxford University Press, 1988.

KURBAN, T.; BESDOK, E. A comparsion of rbf neural netowrk training algorithms for inertial sensor based terrian classification. **Sensors**, p. 6312–6329, 2009.

LEE, C. Y.; J., L. Z.; LIN, S.; YING, K. C. An enhanced ant colony optimization (EACO) applied to a capacitated vehicle routing problem. **Applied Intelligence**, Springer Netherlands, v. 32, p. 88–95, 2010.

LI, L. J.; HUANG, Z. B.; LIU, F. H.; WU, Q. A heuristic particle swarm optimizer for optimization of pin connected structures. **Computers & Structures**, v. 85, n. 7-8, p. 340–349, 2007.

LI, N.; LING, H. Study of an algorithm of ga-rbf neural network generalized predictive control for generating unit. **International Conference on Electric Information and Control Engineering**, p. 1723–1726, 2011.

LIN, J.-C.; TZENG, H.-Y. Applying Particle Swarm Optimization to estimate software effort by multiple factors software project clustering. In: **International Computer Symposium**. [S.l.: s.n.], 2010. p. 1039–1044.

MARINI, F.; WALCZAK, B. Finding relevant clustering directions in high-dimensional data using Particle Swarm Optimization. **Journal of Chemometrics**, John Wiley & Sons, Ltd., v. 25, n. 7, p. 366–374, 2011.

MARQUES, J. M. **Testes estatísticos: para cursos das áreas biológica e da saúde com uso do computador**. Curitiba: Domínio do Saber, 2004.

MICHALEWICZ, Z.; LOGAN, T. D.; SWAMINATHAN, S. Evolutionary operators for continuous convex parameter spaces. **Proceedings of the 3rd Annual Conference on Evolutionary Programming**, p. 84–97, 1994.

MING, Z. Y.; BIN, Z. Y.; ZHONG, L. L. Application of genetic algorithm and rbf neural network in network flow prediction. **3rd IEEE International Conference on Computer Science and Information Technology**, p. 298–301, 2010.

MINGOTI, S. A. **Análise de dados através de métodos de estatística multivariada: uma abordagem aplicada**. Belo Horizonte: Editora da UFMG, 2005.

MOTA, J. F. da; SIQUEIRA, P. H.; SOUZA, L. V. de; VITOR, A. Training radial basis function neural netowrks by genetic algorithms. In: **International Conference on Angents and Artificial Intelligence 2012**. Vilamoura, Algarve, Portugal.: [s.n.], 2012.

OMKAR, S.; R., K.; T.V.S., A.; NAIK, G. N.; GOPALAKRISHNAN, S. Quantum behaved Particle Swarm Optimization (QPSO) for multi-objective design optimization of composite structures. **Expert Systems with Applications**, v. 36, n. 8, p. 11312–11322, 2009.

POWELL, M. J. D. Radial basis function approximation to polynomials. **Numerical Analysis 1987 Proceedings**, Dundee, UK, p. 223–241, 1988.

RUTKOWSKI, L. **Computational intelligence: methods and techniques**. [S.l.]: Springer-Verlag Berlin Heidelberg, 2008.

SILVA, I. N.; SPATTI, D. H.; FLAUZINO, R. A. **Redes neurais artificiais - para engenharia e ciências aplicadas**. São Paulo: Artliber, 2010.

SIM, J.; WRIGHT, C. C. The kappa statistics in reliability studies: use, interpretation and sample size requirements. **Physical Therapy**, v. 85, n. 3, p. 257–268, 2005.

SIQUEIRA, P.; SCHEER, S.; STEINER, M. T. A. Application of the ”winner takes all”principle in wang's recurrent neural network for the assignment problem. **Lecture Notes in Computer Science**, v. 3496, n. 1, p. 731–738, 2005.

SUGANTHAN, P. N. Particle swarm optimiser with neighbourhood operator. In: **Proceedings of the 1999 Congress on Evolutionary Computation**. [S.l.: s.n.], 1999. v. 3.

VILLWOCK, R.; STEINER, M. T. A.; SIQUEIRA, P. H. Pattern clustering using ants colony, ward method and kohonen maps. In: **Proceedings of the International Conference on Evolutionary Computation Theory and Applications,**. [S.l.: s.n.], 2011. p. 137–145.

WANG, Y.-R.; KONG, S.-L. Applying genetic algorithms for construction quality auditor assignment in public construction projects. **Automation in Construction**, 2011.

YU, B.; YANG, Z.-Z.; YAO, B. An improved ant colony optimization for vehicle routing problem. **European Journal of Operational Research**, v. 196, n. 1, p. 171–176, 2009.

ZHANGANG, Y.; YANBO, C.; CHENG, K. W. E. Genetic algorithm-based rbf neural network load forecasting model. **Power Engineering Society General Meeting**, IEEE, p. 1–6, 2007.