CAPÍTULO 6

APLICAÇÕES EM ENGENHARIA

Este capítulo apresenta os resultados computacionais provenientes da aplicação das quatro ferramentas de engenharia imunológica propostas no Capítulo 5 (SAND, CLONALG, ABNET e aiNet) a diversos problemas de engenharia, enfocando as áreas de aprendizagem de máquina, reconhecimento de padrões, aproximação de funções, análise de dados e otimização. Juntamente com as aplicações das ferramentas, são feitas comparações de desempenho com outros algoritmos conhecidos na literatura, onde a grande maioria deles foi revisada no Capítulo 4.

"As aplicações práticas de uma ciência geralmente antecedem o desenvolvimento da própria ciência" – N. K. Jerne

6.1. Introdução

Existe uma grande quantidade de problemas em engenharia cujas soluções, inclusive aquelas obtidas por estratégias de inteligência computacional, ainda são insatisfatórias. Destas aplicações, podemos destacar os problemas de busca (ou otimização) multimodal, análise de dados e classificação de padrões. Além disso, muitas das estratégias de sistemas inteligentes como as redes neurais artificiais e os algoritmos evolutivos possuem deficiências que vão da inicialização à estruturação, chegando até suas estratégias de aprendizagem/evolução. Este capítulo visa demonstrar como as ferramentas propostas nesta tese podem ser utilizadas na solução de problemas de engenharia como os citados acima e, também, como elas podem ser aplicadas na otimização e desenvolvimento de outras abordagens de sistemas inteligentes, com destaque para as redes neurais artificiais e os algoritmos evolutivos.

O modelo proposto para geração de diversidade populacional, SAND, foi o primeiro a ser desenvolvido nesta tese e, por isso, o primeiro a ser apresentado no Capítulo 5. Por outro lado, o algoritmo SAND possui aplicações em diversas áreas, inclusive em outras estratégias propostas nesta tese como o algoritmo CLONALG. Sendo assim, as aplicações em engenharia do algoritmo de seleção clonal (CLONALG) serão apresentadas antes das aplicações do algoritmo SAND.

A Seção 6.3.1 que aplica a ABNET ao problema de decisão de lógica majoritária (MLD – *major logic decision*) foi desenvolvida em cooperação com Getúlio A. de Deus Júnior, em fase de doutoramento pelo Departamento de Comunicações (DECOM) da Faculdade de Engenharia Elétrica e de Computação (FEEC) da Unicamp.

6.2. CLONALG

O algoritmo de seleção clonal (descrito na Seção 5.3), CLONALG, foi desenvolvido em duas versões: uma para resolver problemas de reconhecimento de padrões e aprendizagem de máquina e outra para solucionar problemas de otimização, com destaque para otimização multimodal. Com o objetivo de avaliar o desempenho da versão para o reconhecimento de padrões, ele será aplicado ao reconhecimento de caracteres binários. No caso de problemas de otimização, o algoritmo será testado em diversas funções uni- e multi-dimensionais, e seus resultados comparados com o algoritmo genético clássico e o método de fitness sharing discutido na Seção 4.4.5.

6.2.1. Reconhecimento de Padrões

A capacidade de aprendizagem e aquisição de memória do algoritmo CLONALG foi verificada através de sua aplicação a um problema de reconhecimento de caracteres binários.

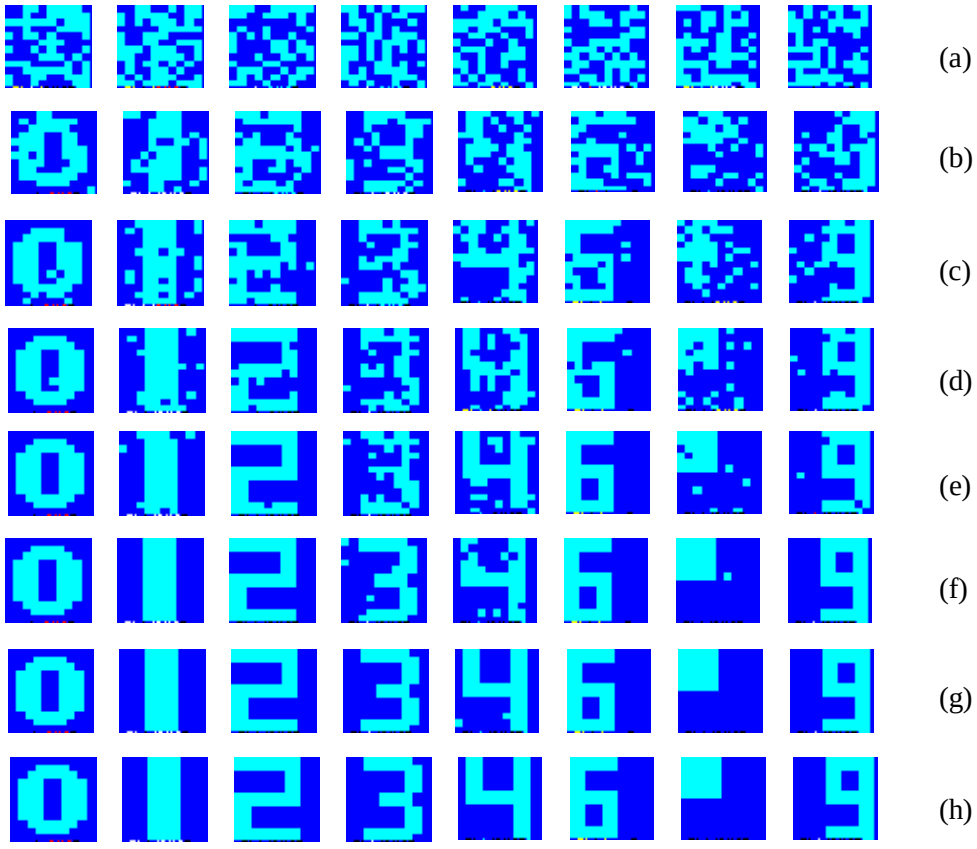


Figura 6.1. Aplicação do algoritmo CLONALG ao problema de reconhecimento de caracteres binários, onde (h) representa o conjunto de antígenos (padrões) a serem aprendidos. Conjunto de memória $\mathbf{Ab}_{\{m\}}$ inicial (a), após 20 (b), 50 (c), 75 (d), 100 (e), 150 (f), 200 (g), e 250 gerações (g). O algoritmo convergiu após 300 gerações.

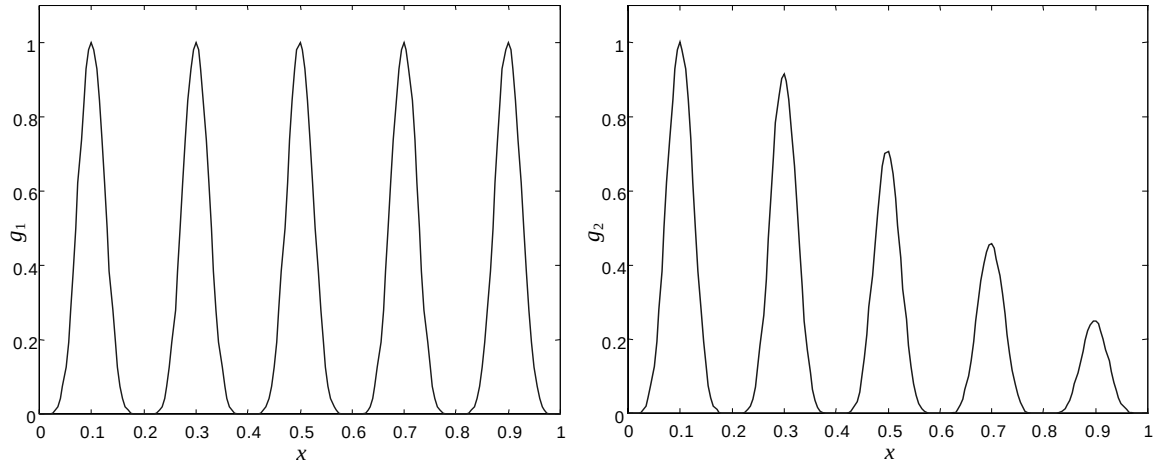
O objetivo desta seção, é demonstrar que uma variação genética aleatória e cumulativa, juntamente com uma forte pressão seletiva, é capaz de produzir indivíduos com afinidades antigênicas cada vez maiores, simulando o processo de maturação de afinidade da resposta imune adaptativa. Neste caso, assumimos a existência de uma população de antígenos, a serem reconhecidos, representada em um espaço de formas binário de Hamming composto por oito cadeias de atributos (M = 8) de comprimento L = 120. Estes caracteres já foram utilizados por Lippmann (1987) para avaliar a capacidade de armazenamento e recuperação de padrões de redes neurais discretas de Hopfield (Seção 4.2.2.3). O repertório total **Ab** de anticorpos é composto por N = 10 indivíduos, onde m = 8 deles compõem o conjunto de memória $\mathbf{Ab}_{\{m\}}$. O conjunto de antígenos a serem reconhecidos está ilustrado na Figura 6.1(h). A Figura 6.1(a) apresenta o conjunto de memória $\mathbf{Ab}_{\{m\}}$ inicial, e a Figura 6.1 de (b) a (h) representa o processo de maturação do conjunto de memória ao longo das gerações celulares. A medida de afinidade empregada foi a distância de Hamming entre os antígenos e os anticorpos, dada pela Equação 3.3.

Considerando o algoritmo de treinamento apresentado na Seção 5.3.2.1.1, a seguinte linha de comando foi utilizada: `[Abm,F]=clonalg(Ab,Ag,120,300,5,10,0)`.

6.2.2. Otimização

O algoritmo CLONALG reproduz aqueles indivíduos com altas afinidades e seleciona seus clones maturados de maior afinidade. Esta estratégia sugere um algoritmo capaz de realizar uma busca local em torno de cada indivíduo da população. Para avaliar a aplicabilidade deste método a problemas de otimização, considere inicialmente as seguintes funções multimodais unidimensionais ilustradas na Figura 6.2:

- $g_1(x) = sen^6(5\pi x)$; e
- $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$.



(a) (b)

Figura 6.2. Funções a serem maximizadas. (a) Múltiplos ótimos globais, $g_1(x) = sen^6(5\pi x)$. (b) Um único ótimo global e vários ótimos locais, $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$.

Estas funções foram utilizadas por Goldberg & Richardson (1987) para estudar a capacidade de otimização multimodal do método de compartilhamento de fitness (*fitness sharing*) apresentado na Seção 4.4.5.

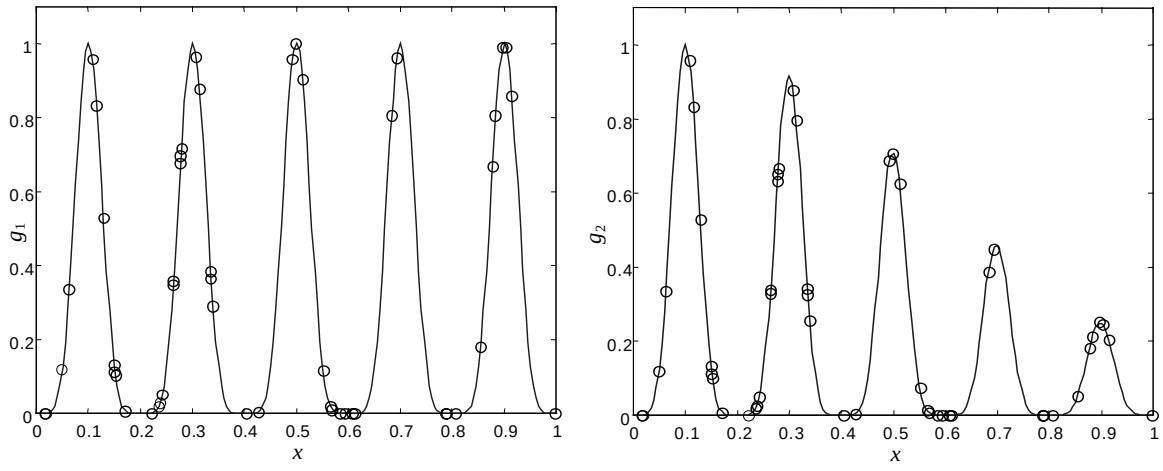
O objetivo é encontrar a maior quantidade de valores máximos destas funções, sejam eles locais ou globais. Considerando o algoritmo de treinamento apresentado na Seção 5.3.2.2.1, a linha de comando `[Ab,f]=clonalg(Ab,22,50,50,10,0)` foi utilizada. O repertório **Ab** possui $N = 50$ anticorpos, codificados por cadeias binárias de comprimento $L = 22$ em um espaço de formas binário de Hamming. O comprimento $L = 22$, corresponde a uma precisão de seis casas decimais. A variável x está definida no intervalo [0,1], e o mapeamento de uma cadeia binária $m = \langle m_L,..., m_2, m_1 \rangle$ em um número real x é executado em dois passos:

- Converta a cadeia binária $m = \langle m_L,..., m_2, m_1 \rangle$ da base 2 para a base 10:

  $$\left(\langle m_L,...,m_2,m_1\rangle\right)_2 = \left(\sum_{i=0}^{21} m_i.2^i\right)_{10} = x'$$

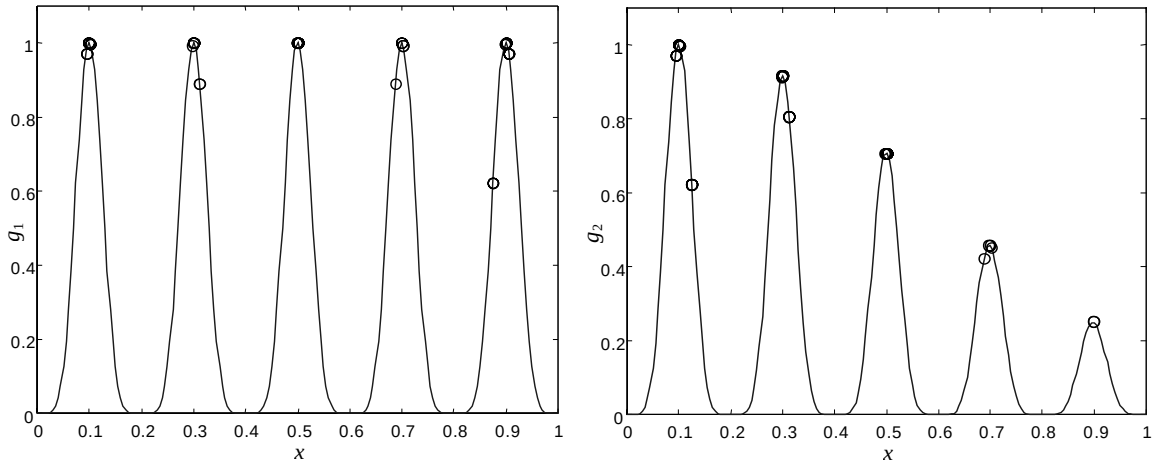
- Encontre o correspondente valor real para x: $x = x_{\min} + x'.\frac{x_{\max} - x_{\min}}{2^{22}-1}$, onde $x_{\max} = 1$ e $x_{\min} = 0$.

A medida de afinidade corresponde à avaliação da função $g_i(x)$, $i = 1, 2$, após decodificarmos x como descrito acima. A Figura 6.3 ilustra as populações iniciais e a Figura 6.4 apresenta as populações otimizadas pelo algoritmo CLONALG.



(a) (b)

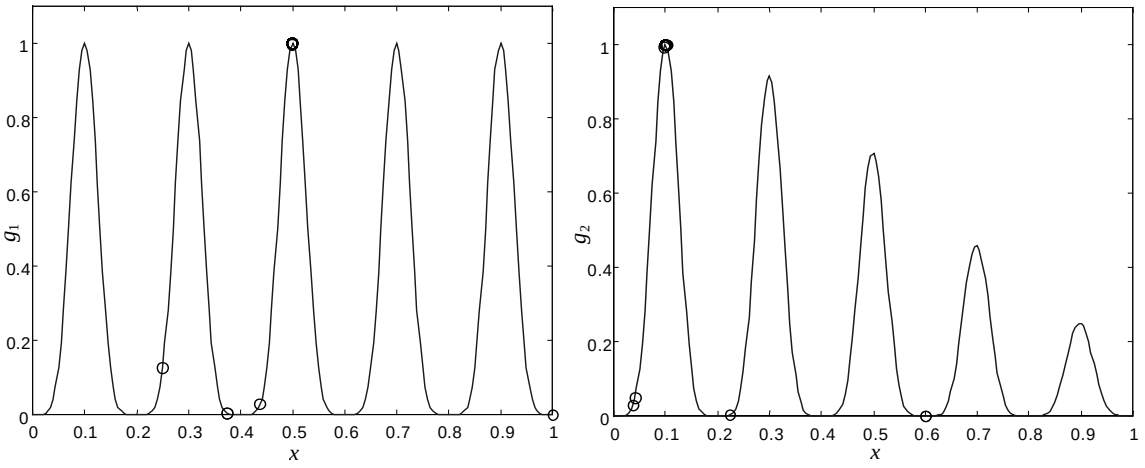
Figura 6.3. Populações iniciais compostas por $N = 50$ indivíduos para maximização das funções: (a) $g_1(x) = sen^6(5\pi x)$ e (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$.



(a) (b)

Figura 6.4. Algoritmo CLONALG aplicado ao problema de maximização das funções: (a) $g_1(x) = sen^6(5\pi x)$ e (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$.

Para comparar o desempenho do algoritmo CLONALG, aplicamos um GA (Seção 4.4.3) com seleção por torneio binário, crossover simples e mutação de múltiplos pontos. Os parâmetros escolhidos para a evolução do GA foram: $p_c = 0.6$, $p_m = 0.01$ e $k = 0.6$. A quantidade de gerações e a população inicial foram as mesmas utilizadas para executar o CLONALG, $gen = 50$, $N = 50$. A Figura 6.5 apresenta as populações finais geradas pelo GA quando aplicado às funções $g_1(x)$ e $g_2(x)$.



(a) (b)

Figura 6.5. Algoritmo genético (GA) aplicado ao problema de maximização das funções: (a) $g_1(x) = sen^6(5\pi x)$ e (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$.

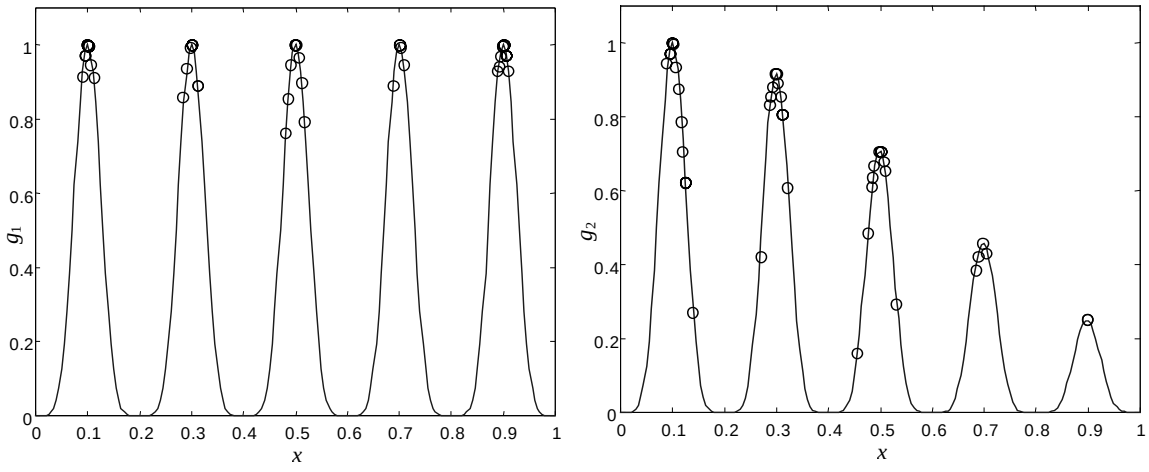
Devido a capacidade do algoritmo CLONALG desempenhar uma busca multimodal, ele também foi comparado ao método de fitness sharing (GA_{sh}) apresentado na Seção 4.4.5. Utilizou-se uma medida genotípica de sharing, ou seja, para avaliar o compartilhamento no fitness de cada indivíduo da população, calculou-se a distância de Hamming entre eles (Equação 3.3) e aplicou-se a função de compartilhamento dada pelas Equações 4.16 e 4.17. Os parâmetros escolhidos para o GA com fitness sharing e seleção por torneio binário foram: $p_c = 0.6$, $p_m = 0.0$ e $\sigma_{share} = 4$. O desempenho do algoritmo está ilustrado na Figura 6.6. Com o objetivo de manter estável a população evoluída pelo algoritmo de fitness sharing quando combinado com seleção por torneio, utilizamos o método de fitness sharing contínuo, como proposto por Oei *et al.* (1991).

Comparando-se a Figura 6.4 com a Figura 6.6, verifica-se que o algoritmo CLONALG apresenta um menor número de elementos sub-ótimos do que o GA_{sh}, ou seja, a sub-população de indivíduos ocupando cada nicho da função (ótimo local ou global) é mais uniforme, com a grande maioria dos indivíduos posicionados nos picos da curva.

Estes algoritmos também foram aplicados ao problema de maximização da função de duas variáveis $g_3(x,y) = x.sen(4\pi x) - y.sen(4\pi y + \pi) + 1$, ilustrada na Figura 6.7.

As variáveis x e y estão definidas no intervalo $[-1,2]$ e podem ser decodificadas como no caso de $g_1(x)$ e $g_2(x)$ adotando-se $x_{min} = y_{min} = -1$ e $x_{max} = y_{max} = 2$. Os parâmetros comuns utilizados para todos os algoritmos foram $N = 100$, $L = 22$, e os outros parâmetros adotados foram:

- CLONALG: `[Ab,f]=clonalg(Ab,22,100,100,10,10);`
- GA: $p_c = 0.6$, $p_m = 0.01$ e $k = 0.6$; e
- GA_{sh}: $p_c = 0.6$, $p_m = 0.0$ e $\sigma_{share} = 4$.



(a) (b)

Figura 6.6. Algoritmo genético com fitness sharing (GA_{sh}) aplicado à maximização das funções: (a) $g_1(x) = sen^6(5\pi x)$ e (b) $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$.

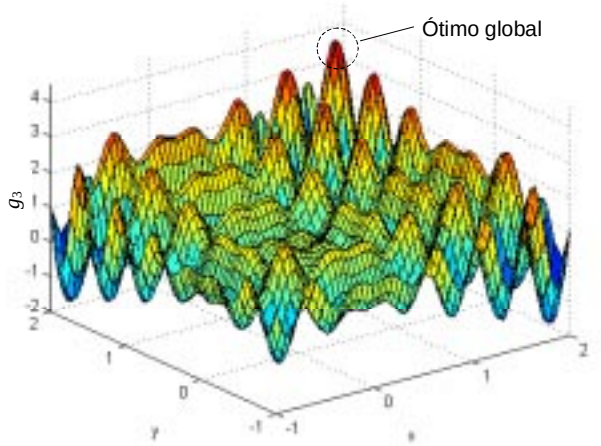
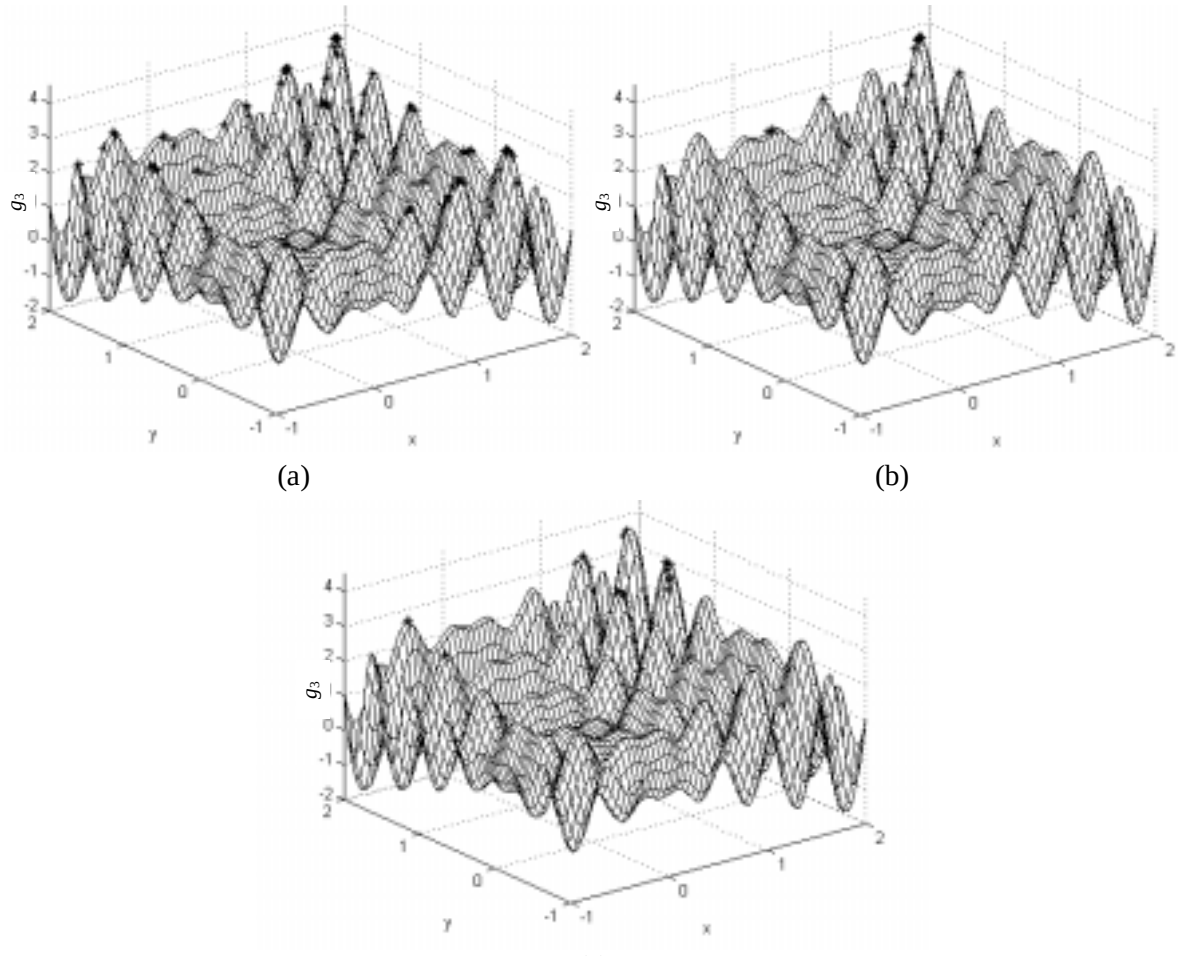


Figura 6.7. Função bidimensional $g_3(x,y) = x.sen(4\pi x) - y.sen(4\pi y + \pi) + 1$ a ser maximizada.

Analisando a Figura 6.8, verifica-se que o algoritmo CLONALG também é mais eficiente na localização dos ótimos locais e globais da função g_3 do que os outros métodos.



(a) (b)

(c)

Figura 6.8. Desempenho dos algoritmos aplicados a função $g_3(x,y) = x.sen(4\pi x) - y.sen(4\pi y + \pi) + 1$. (a) CLONALG. (b) GA. (c) GA_{sh}.

A Figura 6.9 apresenta a evolução do fitness do melhor indivíduo (traço contínuo) e o fitness médio da população (linha tracejada) dos três algoritmos (CLONALG, GA e GA_{sh}) aplicados à função $g_1(x)$. É possível verificar pela Figura 6.9(a) que o algoritmo CLONALG possui uma seleção elitista que preserva o melhor indivíduo na população, e uma evolução estável com característica assintótica em relação aos ótimos da função. O algoritmo genético simples (Figura 6.9(b)) com seleção por torneio salvacionista apresentou uma evolução assintótica em relação a apenas um ótimo da função, como também pode ser visto na Figura 6.5. A seleção por torneio aplicada ao método de fitness sharing contínuo demonstrou instabilidade da população média e do melhor indivíduo, como pode ser observado pela Figura 6.9(c).

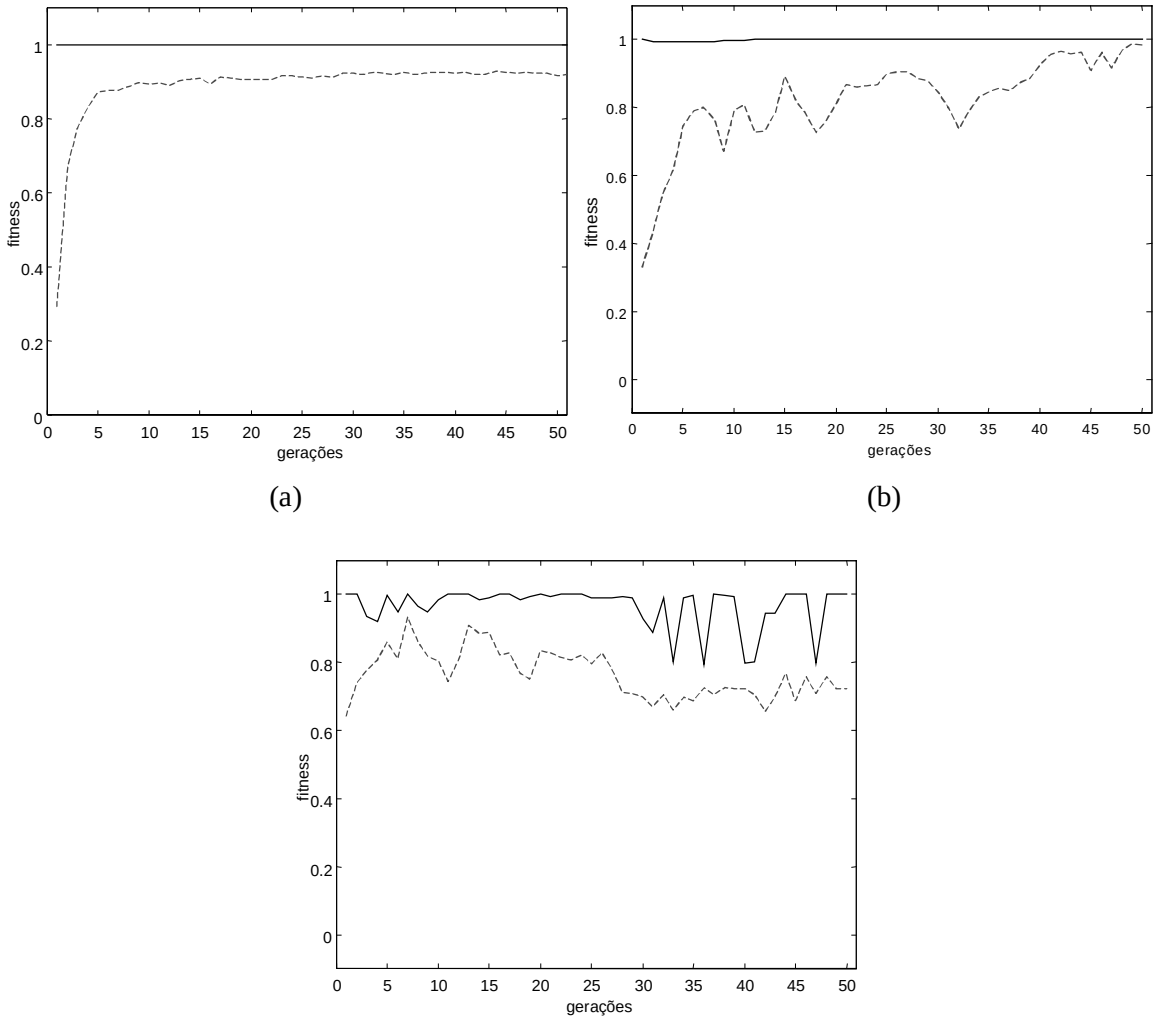


Figura 6.9. Evolução dos algoritmos CLONALG, GA e GA_{sh} aplicados ao problema de maximização da função $g_1(x) = sen^6(5\pi x)$. A curva sólida representa o fitness do melhor indivíduo da população e a curva tracejada o fitness médio da população ao longo das 50 gerações. (a) CLONALG. (b) GA. (c) GA_{sh}.

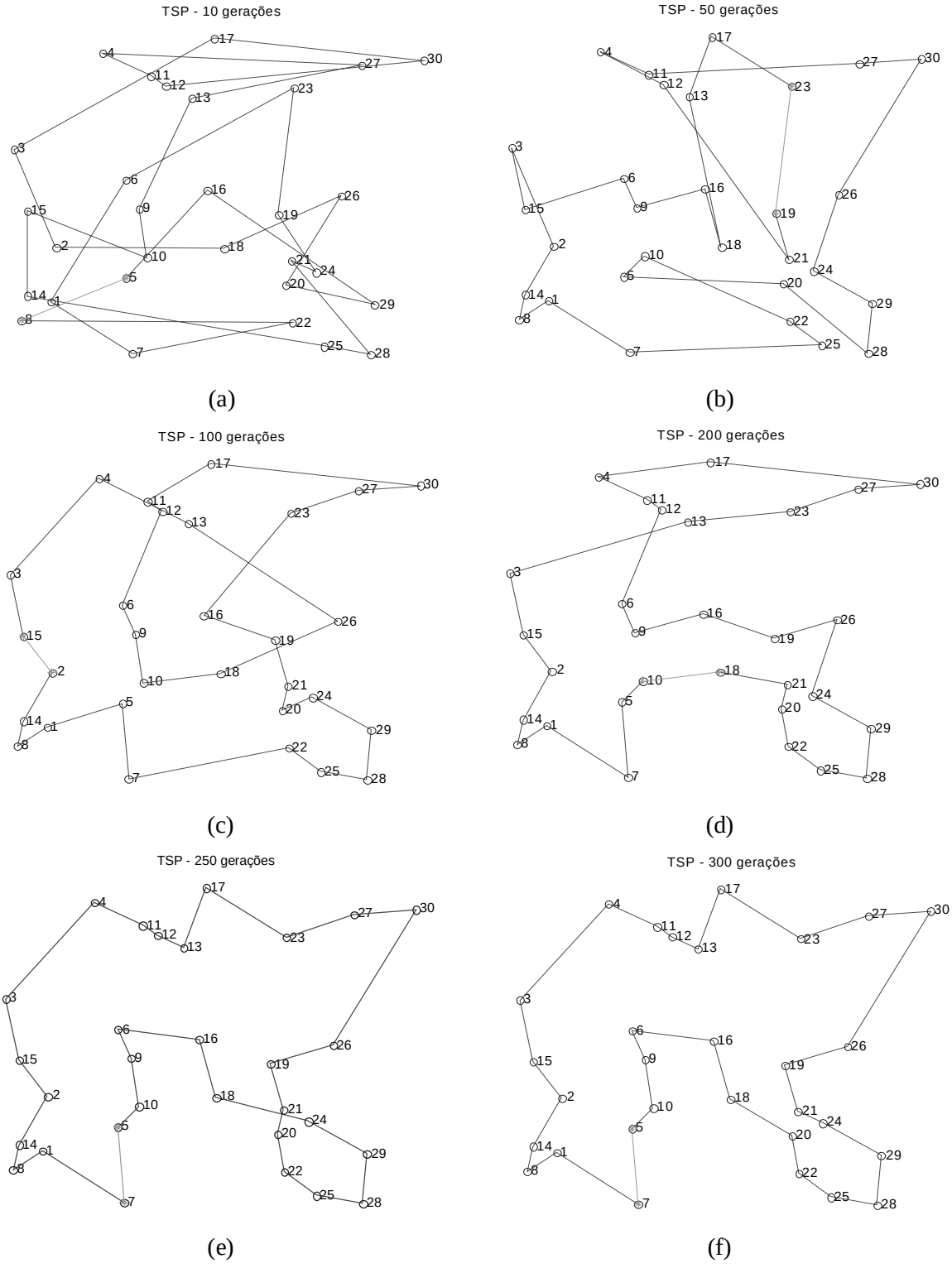


Figura 6.10. Evolução do algoritmo CLONALG (melhor indivíduo da população) aplicado ao problema TSP. Comprimento da rota (em u.m.d. – unidade de medida de distância): (a) 107932, (b) 75627, (c) 60913, (d) 54540, (e) 49433 e (f) 48872.

Para concluir a avaliação de desempenho do algoritmo CLONALG, ele foi aplicado ao *problema do caixeiro viajante* (TSP – *Travelling Salesman Problem*). De forma simples, o caixeiro viajante precisa visitar L cidades em seu território, exatamente uma única vez cada cidade, e depois retornar à cidade de partida. O problema é: dado o custo de viagem entre todos os pares de cidades, qual rota possui o menor custo? Neste trabalho, o custo de uma rota é basicamente o comprimento do caminho percorrido pelo caixeiro viajante (distância total). O problema TSP é do tipo combinatorial e aparece em diversas aplicações, de projeto de circuitos VLSI até sistemas de entregas de *fast food*. Neste caso, a utilização de um espaço de formas Inteiro é apropriada, onde vetores de comprimento L, compostos por uma permutação de elementos do conjunto $C = \{1,2,...L\}$, representam as possíveis rotas a serem percorridas. Cada componente do vetor inteiro indexa uma cidade. O inverso do comprimento total de cada rota fornece a medida de afinidade do vetor correspondente ao anticorpo.

A Figura 6.10 apresenta a evolução do melhor indivíduo da população determinado pelo algoritmo CLONALG. Após 300 gerações (Figura 6.10(f)) foi possível determinar o ótimo global do problema (Moscato & Fontanari, 1990). O tamanho da população é $N = 300$, e os parâmetros adotados foram `[Ab,f]=clonalg(Ab,30,300,150,10,60)`.

O algoritmo CLONALG como proposto nesta tese constitui uma implementação computacional do princípio imunológico da seleção clonal. Sendo assim, este algoritmo poderia ser melhorado para solucionar diversos problemas, como o TSP, onde a utilização de heurísticas ou mecanismos de busca local são capazes de aumentar significativamente a eficiência de algoritmo similares (como o GA) em problemas específicos.

6.2.3. Análise de Sensibilidade

Nesta seção, verificaremos a sensibilidade do algoritmo CLONALG quanto à definição dos seguintes parâmetros:

- n: quantidade de anticorpos a serem selecionados para clonagem, gerando a população $\mathbf{Ab}_{\{n\}}$;
- N_c: quantidade de clones gerada a partir de $\mathbf{Ab}_{\{n\}}$; e
- d: quantidade d de anticorpos com baixa afinidade a serem substituídos.

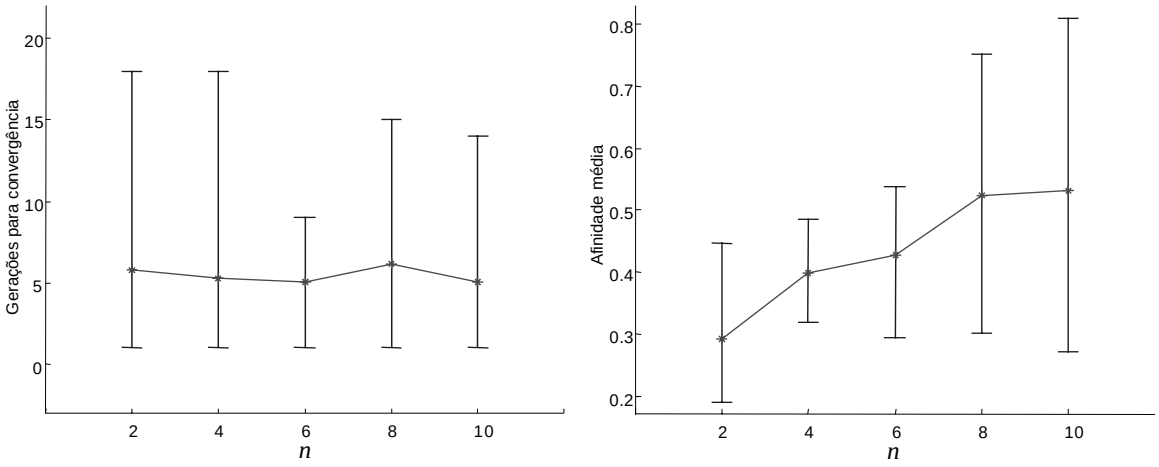
Sem perda de generalidade, analisaremos o problema de otimizar a função unidimensional $g_1(x) = sen^6(5\pi x)$ apresentada na Figura 6.2(a). A população terá um tamanho fixo $N = 10$, correspondente a duas vezes a quantidade de ótimos globais da função a ser maximizada, e será codificada como anteriormente. Nas Figuras 6.11 e 6.12, os resultados apresentados correspondem aos valores máximo, mínimo e médio obtidos a partir de 10 simulações. Desejamos verificar dois aspectos: 1) a capacidade do algoritmo determinar os cinco picos da função (ótimos globais), e o número de gerações necessárias para a determinação destes picos.

Inicialmente, fixamos o tamanho de cada clone gerado como sendo igual a 10 para cada um dos n indivíduos selecionados, portanto $N_c = 10n$. Feito isso, verificou-se a quantidade de gerações necessárias para a determinação de pelo menos um ótimo global da função.

Os resultados estão apresentados na Figura 6.11(a). Neste caso, verifica-se que não existe uma forte relação entre o parâmetro n e a velocidade de convergência do algoritmo, a não ser por um aumento no custo computacional de cada iteração. É importante lembrar, que quanto menor o valor de n em relação a N, menor a capacidade do algoritmo realizar uma busca multimodal, e mais seu comportamento se aproxima de um algoritmo evolutivo clássico. A Figura 6.11(b) mostra que a afinidade média da população sofre um aumento diretamente (e quase linearmente) proporcional a n.

Para analisar a sensibilidade do algoritmo em relação a N_c, a quantidade de clones gerada para cada um dos $n = N$ (= 10) anticorpos do repertório, foram assumidos os seguintes valores: $N_c = \{5,10,20,40,80,160\}$. É possível verificar pela Figura 6.12, que quanto maior a quantidade de clones gerada para cada anticorpo, menor a quantidade de gerações necessária para a convergência do algoritmo, sendo que a convergência foi verificada quando o fitness médio da população atingiu um valor maior ou igual a 0.92.

Por sua vez, o parâmetro d possui uma importância fundamental na manutenção da diversidade da população e potencialidade de explorar novas regiões do espaço de busca. Isso foi verificado da seguinte forma. Gerou-se uma população inicial **Ab** com $N = 10$ anticorpos iguais $\mathbf{Ab}_i = \langle 0,...,0 \rangle$, $i = 1,...,10$. A afinidade de todos estes anticorpos é igual a 0. Executou-se o algoritmo CLONALG para $d = 0$ (Figura 6.13(a)), $d = 1$ e $d = 2$ (Figura 6.13(b)). Como pode ser verificado, para um valor de $d > 0$, o algoritmo foi capaz de localizar todos os máximos globais da função. Por outro lado, é importante salientar que valores muito elevados deste parâmetro podem causar uma busca quase aleatória dentro do espaço de formas, sendo sugerido valores entre 5% e 20% do tamanho total N da população **Ab**.



(a) (b)

Figura 6.11. Análise de sensibilidade do algoritmo CLONALG em relação ao parâmetro n. (a) Número de gerações necessárias para a determinação de pelo menos um ótimo global da função. (b) Valor da afinidade média da população até a determinação de pelo menos um ótimo global da função.

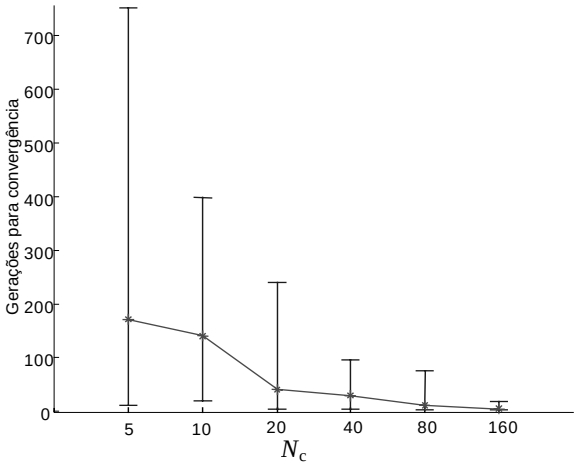
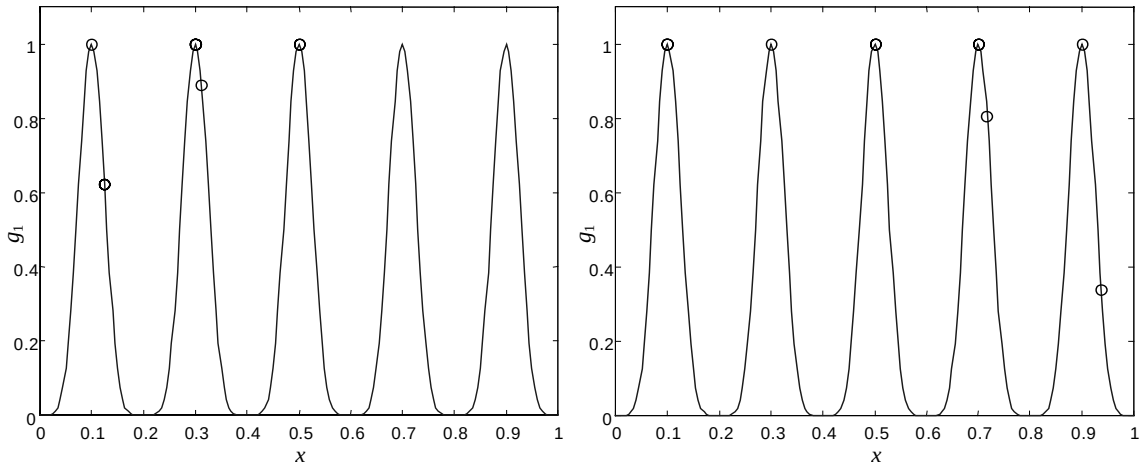


Figura 6.12. Análise de sensibilidade do algoritmo CLONALG em relação ao parâmetro N_c.



(a) (b)

Figura 6.13. Análise de sensibilidade do algoritmo CLONALG em relação ao parâmetro d, $N = 10$. (a) $d = 0$. (b) $d = 2$, correspondendo a 20% da população. Todos os indivíduos da população inicial são idênticos, correspondendo a $x = 0$.

6.2.4. Discussão

Sabemos que o reconhecimento Ag-Ab ocorre por complementaridade de formas e é geralmente medido por um critério de dissimilaridade, como discutido na Seção 3.4. Porém, sem perda de generalidade, utilizamos uma medida de similaridade para avaliar a afinidade Ag-Ab no problema de reconhecimento de caracteres (Figura 6.1), apenas para que pudéssemos representar o conjunto de memória como uma "imagem interna" dos antígenos a serem reconhecidos.

Uma outra forma de apresentar os resultados, que surtiria o mesmo efeito e estaria em maior concordância com a motivação biológica, seria utilizar uma medida de dissimilaridade, como formalmente proposto, e complementar as cadeias de atributos representativas dos anticorpos resultantes para que pudéssemos obter o efeito visual desejado. Note também, que um reconhecimento perfeito não é necessário para a correta classificação do padrão. Poderia ser empregada a idéia de reatividade cruzada introduzindo-se um limiar de afinidade ε que definiria uma região V_ε no espaço de formas de cada elemento da população, permitindo que um anticorpo pudesse reconhecer mais de um antígeno. Outra implicação direta da introdução desse limiar de afinidade seria em uma possível redução no tamanho do repertório de memória. Por exemplo, assuma que um anticorpo $\boldsymbol{Ab}_i$ possui afinidade superior a ε aos antígenos $\boldsymbol{Ag}_j$ e $\boldsymbol{Ag}_k$, ($f_{i,j}$,$f_{i,k} > \varepsilon$), então este anticorpo representa uma imagem interna dos antígenos *j* e *k*, e não seria necessário um anticorpo de memória específico para cada um deles.

Ao compararmos o algoritmo CLONALG aplicado a problemas de otimização multimodal com o método de fitness sharing (GA_{sh}), verificamos que o primeiro algoritmo é bastante eficiente na determinação de múltiplas soluções, enquanto o GA_{sh} possui algumas limitações:

- Ele executa a comparação de cada membro da população com todos os outros, requerendo um esforço computacional da ordem $O(N^2)$; e
- A definição do parâmetro σ_{share} é crítica e depende de uma distribuição uniforme dos picos no espaço de busca, o que pode ser verificado no caso da função $g_3(x,y)$, ilustrada na Figura 6.8(c), onde o desempenho desta estratégia foi baixo.

Verificamos que a velocidade de convergência do CLONALG é diretamente proporcional ao tamanho N_c do clone gerado a cada iteração do algoritmo. Porém existe um custo computacional associado a este parâmetro que não pode ser desconsiderado (ver Seção 5.3.4). Os resultados apresentados também mostram que o algoritmo CLONALG é capaz de gerar diversidade, através do parâmetro *d* (que simula a morte programada das células – anticorpos – pouco estimuladas e inserção de novos indivíduos na população) e da mutação somática, permitindo a determinação de todos os ótimos da função mesmo quando uma população inicial pequena e com baixa (ou nenhuma) diversidade é empregada.

6.3. SAND

O algoritmo de *simulated annealing* para geração de diversidade populacional (SAND), descrito na Seção 5.2, visa produzir de forma antecipatória um repertório de anticorpos que apresenta máxima cobertura do espaço de antígenos a serem reconhecidos. Esta estratégia não assume conhecimento prévio do problema a ser abordado (conjunto de antígenos) e possui aplicabilidade a problemas de reconhecimento de padrões (antígenos) e definição de condições iniciais ótimas para métodos de busca, como em algoritmos evolutivos e redes neurais artificiais.

O algoritmo SAND foi desenvolvido em duas versões: uma para espaços de formas binários de Hamming e outra para espaços Euclidianos.

Com o objetivo de avaliar o desempenho da versão para espaços de Hamming, ele será aplicado à "otimização" (maximização da diversidade) de populações de anticorpos aleatoriamente geradas. Estas populações otimizadas pelo SAND serão empregadas no reconhecimento de populações binárias de antígenos, e seus resultados comparados com a capacidade de reconhecimento apresentada pelas populações originais de anticorpos, ou seja, com as populações de anticorpos antes de serem otimizadas pelo SAND. Além disso, o algoritmo para espaços binários de Hamming também foi utilizado para inicializar os processos de busca do algoritmo CLONALG e de um algoritmo genético (GA) clássico, aplicados a problemas de otimização multimodal e global, respectivamente.

No caso de espaços de formas Euclidianos, o SAND foi aplicado ao problema de inicialização de redes neurais artificiais do tipo perceptron de múltiplas camadas (MLP) treinadas via um algoritmo de otimização não-linear irrestrita de segunda ordem. O método SAND para inicialização de redes MLP será aplicado a diversos problemas de teste e de mundo real e seu desempenho comparado com o de várias outras estratégias propostas na literatura.

6.3.1. Espaço de Formas Binário de Hamming

Serão estudados dois problemas: o reconhecimento de populações aleatórias de antígenos e a inicialização de métodos de busca exploratória.

6.3.1.1. Reconhecimento de Antígenos

O aumento na cobertura do espaço de formas binário de Hamming pode ser verificado através da aplicação do algoritmo SAND no reconhecimento de populações binárias de antígenos.

Tabela 6.1. Medida de energia (Equação 5.5) de populações aleatoriamente geradas (P_{RP}) versus a energia de populações evoluídas pelo algoritmo SAND (P_{SAND}). *N* é o tamanho da população e *L* o comprimento das cadeias binárias. Os resultados são o máximo, mínimo, média e desvio padrão obtidos a partir de 20 simulações.

			P_{RP} **(%)**				P_{SAND} **(%)**			
L	2^L	*N*	*Max*	*Min*	*Média*	δ	*Max*	*Min*	*Média*	δ
3	8	8	81.25	50.00	68.12	9.06	100	100	100	0
4	16	16	78.12	62.50	69.06	4.76	100	100	100	0
5	32	32	73.44	62.50	68.28	3.61	100	100	100	0
6	64	64	73.44	64.06	69.45	2.91	100	100	100	0
7	128	128	71.48	66.80	68.32	1.46	100	100	100	0
8	256	128	71.68	66.21	69.30	1.62	100	100	100	0
9	512	256	68.18	63.48	65.38	1.26	100	100	100	0

A Tabela 6.1 apresenta uma comparação da energia da população (Equação 5.5) considerando populações aleatoriamente geradas (P_{RP}) e populações evoluídas pelo algoritmo P_{SAND}. Os resultados apresentados são o máximo, mínimo, média e desvio padrão obtidos a partir de 20 simulações. Pode ser observado que o SAND é capaz de maximizar a energia das populações geradas aleatoriamente, ou seja, ele é capaz de maximizar a cobertura do espaço de buscas, ou a diversidade populacional.

Considerando o algoritmo apresentado na Seção 5.2.3.2, a seguinte linha de comando foi utilizada: `[Ab] = sand(N,L,100.N,0.8,3,100)`. O comprimento L das cadeias binárias assumiu os valores L = {3, 4, 5, 6, 7, 8, 9} e a quantidade de anticorpos N = {8, 16, 32, 64, 128, 128, 256}, respectivamente.

É possível concluir, pela Tabela 6.1, que se uma população **Ag** de antígenos fosse gerada aleatoriamente com cadeias binárias de comprimento L = 3, 4, 5, 6, 7, 8 e 9, as populações de anticorpos evoluídos seriam capazes de se ligar a qualquer antígeno do conjunto **Ag** para qualquer valor do limiar de afinidade ε, indicando uma ligação perfeita (Figura 6.14(a)). Neste caso, um valor de $\varepsilon > 1$ implica que haverá anticorpos na interseção dos conjuntos definidos por cada um deles, gerando anticorpos com reatividade cruzada que podem se ligar a mais do que um antígeno, iniciando uma resposta reativa cruzada (Figura 6.14(b)). Se o repertório de anticorpos tivesse sido gerado aleatoriamente, seria necessário um limiar de afinidade $\varepsilon > 0$ para que alguns dos antígenos fossem reconhecidos, indicando que a população gerada aleatoriamente não apresenta uma máxima cobertura do espaço de formas (Figura 6.14(c)).

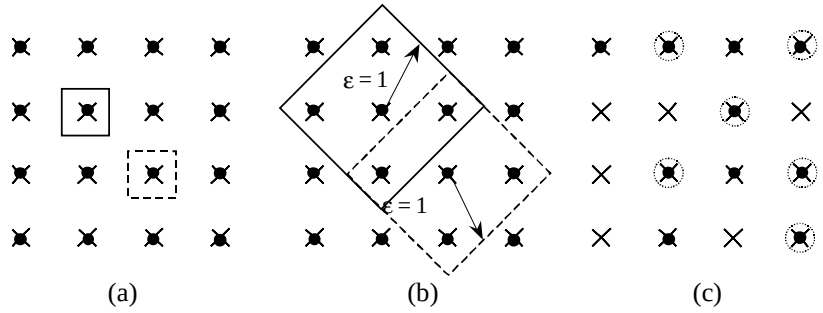


Figura 6.14. Idéia ilustrativa de cobertura do espaço de formas, onde (×) corresponde aos antígenos que representam todo o espaço de formas, e (•) são os anticorpos. (a) Existem anticorpos suficientes para cobrir todo o universo de antígenos. O algoritmo SAND evoluiu uma população capaz de apresentar a máxima cobertura do espaço, sem permitir que um anticorpo reconheça outro anticorpo do repertório ($\varepsilon < 1$). Existe uma superposição entre todos os antígenos e os anticorpos. (b) Além da superposição Ag-Ab, para $\varepsilon > 1$, interseções dos conjuntos de anticorpos estimulados pelo antígeno ocorrerão. (c) A população aleatoriamente gerada pode conter anticorpos suficientes para cobrir todo o espaço de formas (N = 16), porém existe a superposição de alguns deles (representada pelos círculos em torno de • e ×).

Dada uma cadeia binária de comprimento L, para que seja possível medir a reatividade cruzada (capacidade de generalização ou cobertura do espaço de formas) dos repertórios de anticorpos evoluídos e aleatoriamente gerados, foi adotado o seguinte procedimento. De todos os $M = 2^L$ possíveis antígenos, uma quantidade N_{Ag} desta população foi aleatoriamente escolhida. N anticorpos também foram aleatoriamente gerados, compondo uma população denominada P_{RP}, e seguidamente evoluídos pelo algoritmo SAND, compondo uma nova população denominada P_{SAND}.

Tabela 6.2. Reatividade cruzada. Percentual de antígenos reconhecidos corretamente pelas populações P_{RP} e P_{SAND}. Valores máximo, mínimo, média e desvio padrão obtidos a partir de 10 simulações para $L = 9$ e $N_{Ag} = 50$.

N	ε	P_{RP} (%)				P_{SAND} (%)			
		Max	*Min*	*Média*	δ	*Max*	*Min*	*Média*	δ
5	0	2	0	0.4	0.84	2	0	1.2	1.03
	1	16	4	8.2	3.33	16	6	10.8	3.91
	2	46	30	37.4	5.97	52	30	40.2	6.21
	3	86	64	77.6	7.82	92	78	86.2	4.94
10	0	4	0	1.4	1.35	6	0	2.8	1.69
	1	20	8	15.8	3.33	22	6	17	5.01
	2	68	58	64	3.65	76	54	67.4	7.24
	3	100	86	94.2	4.16	100	96	98.8	1.69
25	0	12	2	4.6	3.41	10	2	5.2	2.35
	1	54	32	39.6	6.10	60	36	46.8	7.73
	2	98	86	92.6	3.78	98	86	93.6	4.50
	3	100	100	100	0	100	100	100	0
50	0	14	2	8.2	3.94	16	4	8.4	4.20
	1	72	54	62.8	5.59	74	56	64	5.08
	2	100	96	99.2	1.40	100	98	99.8	0.63
	3	100	100	100	0	100	100	100	0
75	0	22	4	12.2	4.66	24	8	14.8	4.34
	1	84	68	76.8	5.90	88	78	81.4	2.99
	2	100	98	99.8	0.63	100	100	100	0
	3	100	100	100	0	100	100	100	0
100	0	26	12	18	4.62	30	12	20.8	5.09
	1	92	76	86.4	4.50	98	80	90.6	5.42
	2	100	100	100	0	100	100	100	0
	3	100	100	100	0	100	100	100	0

Para diversos valores de ε e N, o percentual de reconhecimento correto das populações de antígenos pelos repertórios P_{RP} e P_{SAND} foi avaliado, onde o reconhecimento ocorre caso, para cada antígeno i do conjunto **Ag**, exista pelo menos um anticorpo j do conjunto **Ab** cuja distância de Hamming ($\boldsymbol{Ag}_i$-$\boldsymbol{Ab}_j$) < ε.

A Tabela 6.2 apresenta os valores máximo, mínimo, médio e o desvio padrão do percentual de antígenos reconhecidos corretamente para 10 simulações, assumindo $L = 9$ e $N_{Ag} = 50$. Como pode ser observado por esta tabela, as populações evoluídas pelo SAND apresentam um percentual de reconhecimento maior do que aquelas aleatoriamente geradas. Neste caso, um antígeno é considerado reconhecido corretamente quando a distância de Hamming (Equação 3.3) entre este antígeno e pelo menos um anticorpo do repertório imunológico for menor ou igual a ε, ou seja, $D \leq \varepsilon$. A linha de comando utilizada foi igual à anterior para $L = 9$.

6.3.1.2. Inicialização de Algoritmos Evolutivos

A inicialização da população a ser empregada nos algoritmos evolutivos pode influenciar o resultado obtido ao final do processo de busca. Por exemplo, se considerarmos problemas de busca multimodal, a população inicial poderá determinar a quantidade de ótimos locais e globais obtidos. Para os casos em que apenas o ótimo global da função é desejado, a população inicial poderá aumentar ou não a probabilidade de determinação deste ótimo.

Na Seção 6.2.3, foi feita uma análise de sensibilidade do algoritmo CLONALG em relação à diversos parâmetros definidos pelo usuário. Foi verificado que, para $d = 0$, a capacidade deste algoritmo executar uma busca multimodal fica reduzida, principalmente quando as populações iniciais não são inicializadas apropriadamente (Figura 6.13). Sendo assim, faz-se necessário que a população inicial de candidatos à solução esteja bem distribuída no espaço de formas, de modo que o algoritmo possa determinar a maior quantidade de ótimos locais da função a ser otimizada. Esta característica também é importante quando algoritmos de otimização local como métodos de gradiente são empregados, pois estes métodos geralmente assumem, a priori, que a função a ser otimizada é unimodal.

Para avaliar a influência da condição inicial na determinação da maior quantidade de ótimos de uma função, aplicamos o algoritmo CLONALG na otimização da função $g_1(x) = \text{sen}^6(5\pi x)$, apresentada na Figura 6.2(a). Para populações com cardinalidades distintas, verificou-se a quantidade de ótimos globais obtidos pelo CLONALG quando inicializado com uma população aleatória P_{RP} e quando inicializado pelo algoritmo SAND (P_{SAND}). Sabendo que esta função possui 5 ótimos, todos globais, a Tabela 6.3 apresenta a quantidade de máximos, tomados a partir de 10 simulações, determinados pelo algoritmo CLONALG para os 2 métodos de inicialização. É possível perceber que, mesmo para populações de dimensões reduzidas, como $N = 5$, o algoritmo proposto é capaz de proporcionar um aumento significativo na quantidade de ótimos determinados. Foi utilizada a mesma linha de comando anterior para cadeias de comprimento $L = 22$, como descrito na Seção 6.2.2: `[Ab,f]=clonalg(Ab,22,50,50,10,0)`.

Tabela 6.3. Quantidade de ótimos globais da função $g_1(x) = sen^6(5\pi x)$ determinados pelo algoritmo CLONALG quando a população inicial é gerada aleatoriamente (P_{RP}) e quando ela é gerada pelo algoritmo SAND (P_{SAND}).

N	P_{RP}				P_{SAND}			
	Max	*Min*	*Média*	δ	*Max*	*Min*	*Média*	δ
5	4	1	2.9	0.88	5	3	3.8	0.63
6	5	2	3.5	0.85	5	2	3.8	0.92
7	5	3	4.1	0.57	5	3	4.4	0.70
8	5	3	3.8	0.63	5	3	4.5	0.71
9	5	3	4.2	0.79	5	4	4.6	0.52
10	5	3	4.2	0.63	5	4	4.7	0.48
11	5	4	4.5	0.53	5	4	4.8	0.42
12	5	3	4.3	0.67	5	5	5	0

Como último exemplo de aplicação do SAND para inicialização de algoritmos evolutivos, seja um algoritmo genético (GA) com seleção bi-classista, crossover simples e mutação de múltiplos pontos, aplicado à otimização da função $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$, ilustrada na Figura 6.2(b). Os parâmetros escolhidos para a evolução do GA foram $gen = 100$, $p_c = 0.6$ e $p_m = 0.01$. Neste caso, deseja-se verificar quantas vezes, tomadas a partir de 10 simulações, o GA é capaz de determinar o ótimo global desta função correspondente a um valor de fitness do melhor indivíduo maior do que 0.97, sabendo que o ótimo global desta função corresponde a 1.0. A Figura 6.15 faz uma comparação do percentual de determinação do ótimo global da função $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$ para populações inicializadas aleatoriamente (P_{RP}) e via o algoritmo SAND (P_{SAND}).

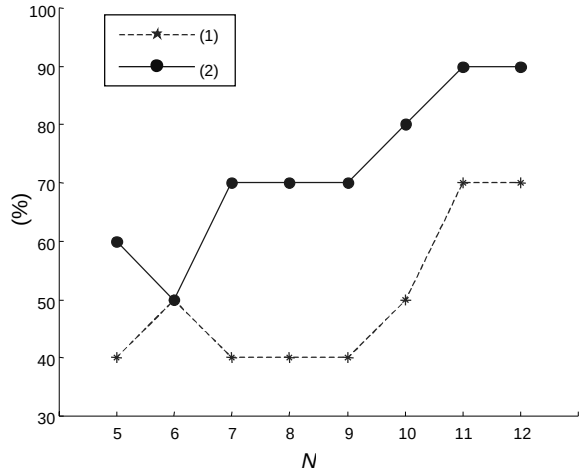


Figura 6.15. Percentual de convergência, a partir de 10 simulações, de um GA aplicado à maximização da função $g_2(x) = 2^{(-2((x-0.1)/0.9)^2} sen^6(5\pi x)$ para populações de candidatos à solução de diferentes tamanhos *N*. P_{RP} (1) e P_{SAND} (2).

6.3.2. Espaço de Formas Euclidianos

Para avaliar o desempenho da versão projetada para espaços de formas Euclidianos do algoritmo SAND, ele será aplicado ao problema de inicialização dos vetores de pesos a serem utilizados no treinamento de redes neurais artificiais do tipo perceptron de múltiplas camadas (MLP) com treinamento via algoritmos de backpropagation. Este conjunto inicial de pesos possui grande influência na velocidade de treinamento e qualidade da solução obtida após a convergência da rede. Uma escolha inadequada deste conjunto, a ser expresso na forma de um vetor pode fazer com que o algoritmo de treinamento fique preso em um mínimo local insatisfatório, ou que o algoritmo apresente problemas de instabilidade numérica.

Cada anticorpo evoluído pelo SAND corresponde a um vetor contendo os pesos de um determinado neurônio em uma das camadas da rede MLP. Assim, gerar uma população de anticorpos com máxima diversidade em S^L corresponde a produzir um conjunto de neurônios com vetores de pesos bem distribuídos em $\Re^L$. O algoritmo SAND para espaços de forma Euclidianos (Seção 5.2.3.3) será aplicado separadamente a cada camada da rede. Um aspecto importante desta estratégia é o fato de que, como estamos gerando vetores com normas unitárias, estes vetores podem ser escalonados (normalizados) de forma que a ativação de cada neurônio ocorra dentro da região aproximadamente linear de suas funções de ativação, evitando a saturação durante o processo de inicialização e treinamento da rede.

O método SAND para inicialização de redes MLP foi comparado com 5 outras estratégias e aplicado a 7 problemas de teste. Os métodos cujos desempenhos foram comparados são: BOERS (Boers & Kuiper, 1992), WIDROW (Nguyen & Widrow, 1990), KIM (Kim & Ra, 1991), OLS (Lehtokangas *et al*., 1995), INIT (de Castro & Von Zuben, 1998a,b), e SAND (de Castro & Von Zuben, 2001c).

Para especificar os problemas de teste utilizados, seja M a quantidade de amostras (antígenos), SSE a soma desejada dos erros quadráticos (critério de parada da rede MLP) e *net* a arquitetura da rede representada por $[n_i\text{-}n_h\text{-}n_o]$, onde n_i é o número de entradas da rede, n_h a quantidade de unidades intermediárias e n_o o número de saídas. Os problemas tomados para comparação foram:

- problema de paridade 2 (XOR): $M = 4$, *net*: [2-2-1], SSE = 0.01;
- problema de paridade 3: $M = 8$, *net*: [3-3-1], SSE = 0.01;
- aproximação da função $sin(x).cos(2x)$: $M = 25$, *net*: [1-10-1], SSE = 0.01;
- ESP: problema de mundo real utilizado por Barreiros *et. al.* (1997); $M = 75$, *net*: [3-10-5], SSE = 0.1;
- SOYA: problema de mundo real utilizado por de Castro *et al.* (1998b), $M = 116$, *net*: [36-10-1], SSE = 0.1;
- IRIS: classificação de tipos de plantas do gênero Iris disponível em (URL 3); $M = 150$, *net*: [4-10-3], SSE = 0.15; e
- ENC/DEC: problema de codificação/decodificação descrito por Fahlman (1988); $M = 10$, *net*: [10-7-10].

Tabela 6.4. Resultados de simulação para inicialização de redes neurais. Os valores apresentados são o máximo, mínimo, média e desvio padrão (δ) do número de épocas para convergência dos algoritmos para os valores do SSE pré-definidos, a partir de 10 simulações.

Problema	Método	*Max*	*Min*	*Média*	δ
XOR	BOERS	129	6	42.90	48.32
	WIDROW	93	8	20.50	25.92
	KIM	84	6	32.50	25.91
	OLS	166	5	46.70	46.50
	SAND	76	7	27.11	28.61
	INIT	47	8	18.60	12.76
Paridade 3	BOERS	62	10	25.10	16.01
	WIDROW	25	10	19.60	5.17
	KIM	46	10	18.90	10.41
	OLS	148	29	65.70	36.16
	SAND	19	14	16.00	2.00
	INIT	24	8	16.00	4.97
$sin(x).cos(2x)$	BOERS	187	79	135.10	35.39
	WIDROW	243	123	178.60	41.27
	KIM	231	95	164.50	44.89
	OLS	133	39	91.40	33.06
	SAND	165	84	127.30	30.43
	INIT	449	181	254.80	80.69
ESP	BOERS	1618	340	883.40	467.52
	WIDROW	2280	236	825.40	639.74
	KIM	1763	425	715.40	397.82
	OLS	2052	35	545.42	586.62
	SAND	766	385	551.50	136.12
	INIT	762	383	479.20	118.19
SOYA	BOERS	174	136	158.60	13.14
	WIDROW	464	176	266.30	79.27
	KIM	280	177	219.40	28.34
	OLS	311	155	220.44	52.53
	SAND	185	125	147.20	17.50
	INIT	236	136	188.00	29.30
IRIS	BOERS	1568	735	1102.40	281.33
	WIDROW	1240	676	918.60	183.72
	KIM	2063	767	1294.80	438.34
	OLS	5000	2075	4140.20	1384.74
	SAND	1169	688	852.80	169.71
	INIT	1407	662	869.80	216.73

Tabela 6.5. Capacidade de convergência dos algoritmos para o problema ENC/DEC. Os valores apresentados incluem o máximo, mínimo, média e desvio padrão (δ) do número de épocas para convergência dos algoritmos para os valores do SSE pré-definidos, a partir de 20 simulações

Problema	Método	*Max*	*Min*	*Mean*	δ	*Convergência*
ENC/DEC	BOERS	40	12	22.21	7.94	14/20
	WIDROW	32	7	12.36	6.81	14/20
	KIM	298	10	38.17	69.07	18/20
	OLS	276	7	97.78	102.03	9/20
	SAND	14	7	10.00	2.24	19/20
	INIT	16	6	9.44	2.53	16/20

O algoritmo de treinamento empregado foi o gradiente conjugado escalonado de Moller (1993), com cálculo exato da informação de segunda ordem, como proposto por Pearlmutter (1994). Para cada algoritmo e cada problema testados, foram efetuadas 10 simulações. Os resultados apresentados na Tabela 6.4 são o máximo, mínimo, média e desvio padrão (δ) da quantidade de épocas para convergência dos algoritmos para os valores do SSE pré-definidos.

Para os espaços Euclidianos, é suposto que as matrizes **Ab**, contendo os vetores de pesos de cada camada, são fornecidas ao algoritmo e, portanto, a linha de comando do SAND passa a ser `[Ab]=sand(Ab,gen,β,δ,mE)`. Os parâmetros gen = 5000, β = 0.9, δ = 5 e mE = 0.995 foram adotados em todas as simulações.

Para o problema ENC/DEC, os algoritmos não foram capazes de convergir em todos os casos. Sendo assim, para apresentar os resultados foram executadas 20 simulações e listada a quantidade de sucessos (convergência) obtida por cada algoritmo (Tabela 6.5), juntamente com os valores máximo, mínimo, médio e desvio padrão dos casos em que as redes convergiram.

Pode ser observado, nas Tabelas 6.4 e 6.5, que o SAND apresentou bons resultados quando comparado às outras abordagens avaliadas. É importante salientar que todos os métodos devem apresentar desempenhos superiores a uma inicialização aleatória, cujos resultados não foram mostrados. A Figura 6.16 faz uma comparação de desempenho dos algoritmos avaliados. Um método é considerado superior a outro quando ele apresenta o menor valor máximo, mínimo, média e desvio padrão do número de épocas para convergência. Os algoritmos OLS, SAND e INIT foram superiores aos outros em praticamente todos os casos.

Resultados empíricos demonstraram a necessidade de re-escalonar os vetores de pesos gerados pelo algoritmo SAND quando aplicado a problemas de mundo real (ESP, SOYA e IRIS). O procedimento de re-escalonamento tende a evitar que os neurônios saturem durante o treinamento. Mesmo assim, a complexidade computacional associada à implementação do SAND é inferior àquela produzida, por exemplo, pelos algoritmos OLS e INIT.

Durante o projeto de uma rede neural, geralmente desejamos que seu desempenho seja o melhor possível para dados não utilizados no seu treinamento, ou seja, tem-se por objetivo que a RNA treinada apresente um boa capacidade de generalização. Entretanto, todas as arquiteturas neurais básicas como o MLP estão sujeitas a um sobre-treinamento (Prechelt, 1998): enquanto a rede parece estar aprendendo a representar o problema cada vez melhor, ou seja, o erro para o conjunto de treinamento decresce, em algum ponto do processo de treinamento o seu desempenho de teste está se deteriorando, ou seja, o erro para amostras não utilizadas no treinamento está aumentando. Existem duas abordagens básicas para evitar, ou aliviar, o sobre-treinamento (Sarle, 1995; Prechelt, 1998): 1) reduzir a dimensão do espaço de parâmetros (dimensão da rede) ou 2) reduzir o número efetivo de parâmetros. Métodos aplicados à redução do espaço de parâmetros são as estratégias construtivas e de poda (Kwok & Yeung, 1997; Reed, 1993), enquanto os métodos utilizados para reduzir o número efetivo de parâmetros são as técnicas de regularização e validação cruzada (Prechelt, 1998).

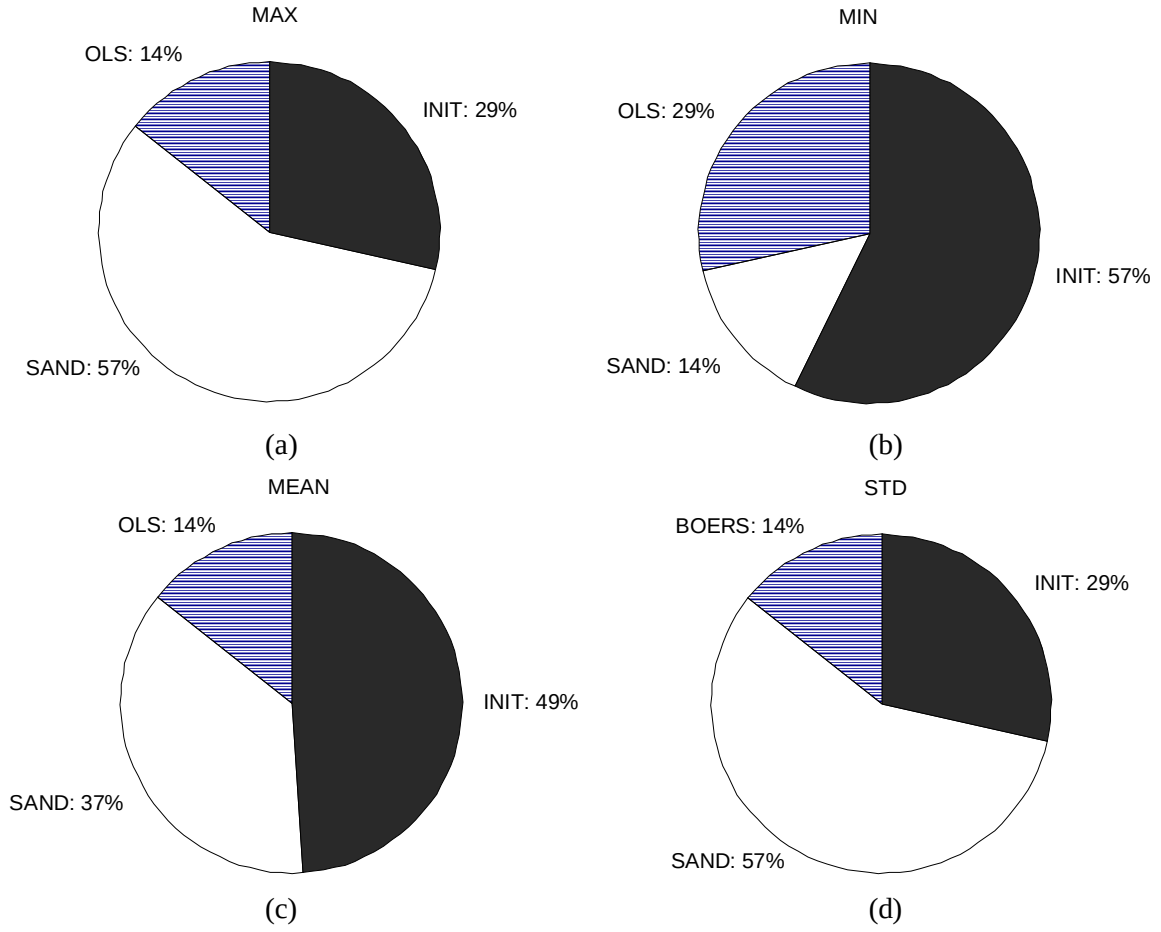


Figura 6.16. Comparação de desempenho de todos os métodos. (a,d) Em 57% dos casos, o algoritmo SAND necessitou do menor número máximo de épocas para convergência e apresentou o menor desvio padrão. (b) Em 14% dos casos, o SAND apresentou o número mínimo de épocas para convergência. (c) Em 37% dos casos, o SAND apresentou o menor número médio de épocas para convergência.

O objetivo não é o de aplicar nenhuma destas estratégias para melhorar a capacidade de generalização das redes, mas apenas o de avaliar a influência de cada método de inicialização na capacidade de generalização das redes neurais treinadas. Para que isso seja feito, todo o conjunto amostral será particionado em dois subconjuntos independentes: um conjunto de treinamento (P) e um conjunto de teste (X). A rede será treinada utilizando o conjunto P e seu desempenho avaliado para o conjunto X. Para cada problema, 10 partições distintas serão feitas considerando 90% das *M* amostras para treinamento da rede, e os 10% restantes para teste. Os valores máximo, mínimo e a média do SSE para o conjunto de teste estão apresentados na Figura 6.17. Os problemas avaliados foram apenas os problemas de mundo real, ou seja, ESP, SOYA e IRIS.

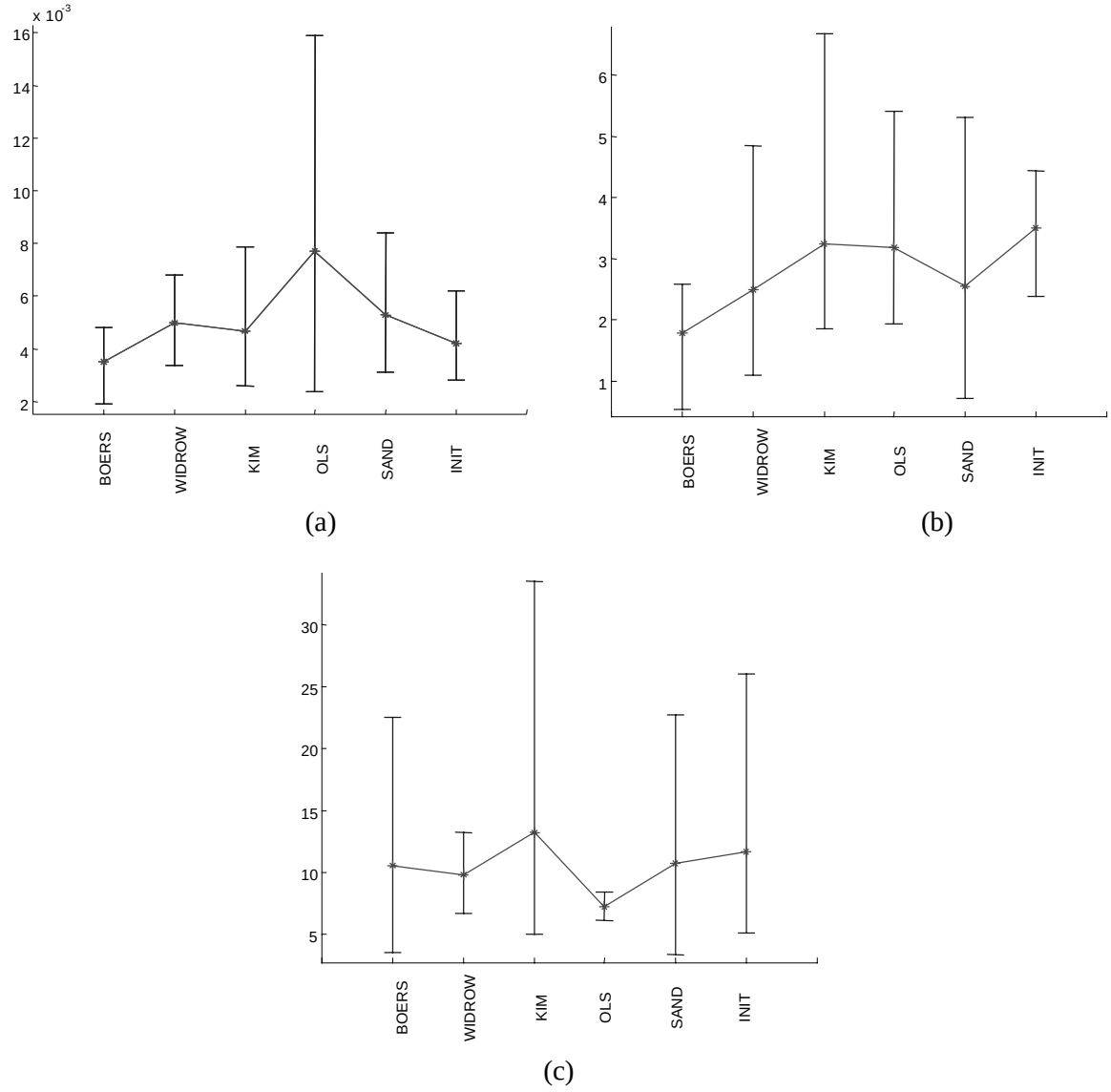


Figura 6.17. Capacidade de generalização dos métodos avaliada para os 3 problemas de mundo real. São apresentados os valores máximo, mínimo e médio (de 10 simulações) do SSE para os conjuntos de teste. (a) ESP. (b) SOYA. (c) IRIS.

Dos resultados apresentados na Figura 6.17, pode-se concluir que nenhum algoritmo se mostrou superior aos demais em termos de capacidade de generalização.

6.3.3. Análise de Sensibilidade

A linha de comando proposta para o algoritmo SAND recebe como entradas os seguintes parâmetros: *N*, *L*, *gen*, β, δ e mE. *N* e *L* irão depender do problema a ser tratado, ou de uma escolha do usuário. Como a função de energia proposta, tanto para o espaço de formas de Hamming quanto para o Euclidiano, assume valores de no máximo 100%, o critério de parada mE está bem definido (qualquer valor próximo ou igual a 100), fazendo com que a quantidade *gen* de gerações possa ser arbitrada de forma a permitir que o algoritmo atinja a diversidade desejada. Portanto, os únicos parâmetros que precisam de uma análise mais criteriosa são:

- β: taxa de decrescimento geométrico da temperatura; e
- δ: limiar de estado estacionário para redução da temperatura.

Estes parâmetros influenciam principalmente na velocidade de convergência do algoritmo. Para avaliar esta influência, foi feito o seguinte experimento: dada uma população binária de $N = 16$ anticorpos de comprimento $L = 5$, verificou-se a quantidade de gerações necessárias para a convergência do algoritmo para uma diversidade máxima da população (mE = 100). A Figura 6.18 apresenta a quantidade média de gerações para convergência do SAND, tomadas a partir de 10 simulações. É possível verificar que, para valores pequenos de δ e β, o algoritmo requer um menor número de iterações para convergência.

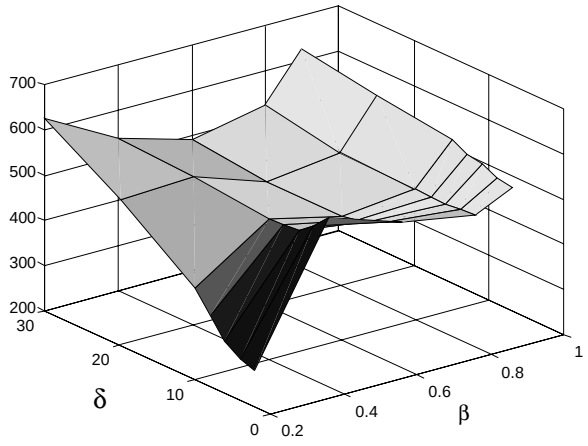


Figura 6.18. Sensibilidade do algoritmo SAND aos parâmetros β e δ. Valores médios da quantidade de gerações para convergência: mE = 100%.

6.3.4. Discussão

Ao comparar a capacidade de reconhecimento antigênico por populações de anticorpos aleatoriamente geradas e populações otimizadas pelo algoritmo SAND, é possível perceber que o método proposto é capaz de gerar repertórios de anticorpos com uma maior cobertura do espaço de formas. O mesmo ocorre quando analisamos a reatividade cruzada (capacidade de generalização) das populações evoluídas.

Esta estratégia demonstrou ser interessante não apenas para o problema de reconhecimento de padrões, mas também para a inicialização de estratégias de busca, incluindo o CLONALG proposto na Seção 5.3, e redes neurais artificiais. Verificou-se que o método permite que um algoritmo de busca multimodal (CLONALG) encontre uma maior quantidade de ótimos locais da função, e também pode aumentar o percentual do número de vezes que um GA converge para o ótimo global da função, principalmente quando populações de tamanho reduzido estão sendo utilizadas. Considerando a inicialização de redes neurais do tipo MLP, o SAND permitiu que, em alguns casos, o algoritmo de treinamento convergisse em um menor número de iterações, mas não apresentou influência significativa na capacidade de generalização da rede treinada.

6.4. ABNET

O algoritmo para a geração de uma rede neural Booleana, ABNET, descrito na Seção 5.4, foi desenvolvido com o objetivo de ilustrar como diversos princípios imunológicos podem ser utilizados para o desenvolvimento e/ou aperfeiçoamento de algoritmos de treinamento de redes neurais artificiais (RNA). Para avaliar o desempenho da ABNET, ela foi aplicada a 3 problemas distintos e seus resultados comparados com outros algoritmos de treinamento de RNAs, como as redes competitivas e os mapas auto-organizáveis de Kohonen (Seção 4.2.2.2) e as redes de Hopfield discretas (Seção 4.2.2.3). Todos os resultados apresentados constituem a média de 10 simulações. Os parâmetros empregados no treinamento da ABNET serão discutidos na Seção 6.4.4 que trata da análise de sensibilidade da rede em relação aos parâmetros de entrada definidos pelo usuário.

6.4.1. Decisão de Lógica Majoritária

Existem diversos problemas na área de telecomunicações que podem ser descritos através de mapeamentos binários, incluindo o problema de Decisão de Lógica Majoritária (MLD – *major logic decision*), que é a regra de decisão que minimiza a probabilidade de erro de mensagem no receptor de uma linha de comunicação. Para avaliar o desempenho da ABNET quando aplicada a problemas de mundo real, seu algoritmo de treinamento foi empregado na obtenção da solução do problema MLD.

6.4.1.1. Descrição do Problema

Uma possível aplicação do serviço de comunicação digital é a investigação da modulação do espectro de dispersão utilizando um detector não coerente FH-FSK (*frequency hopping-*

frequency shift keying), como proposto pelos laboratórios Bell (Goodman *et al.*, 1980; Einarson, 1980).

O diagrama de blocos de um transmissor FH-CDMA de múltiplos níveis é apresentado na Figura 6.19(a). Todas as operações binárias consideradas foram tomadas sobre o *campo de Galois GF*(*Q*). A palavra código resultante enviada através do canal por uma seqüência de *L* sinais FSK é dada por

$$\bar{y} = \bar{a} + \bar{x}, \tag{6.1}$$

onde $\bar{x}$ representa uma mensagem *Q*-ária gerada pelo *m*-ésimo usuário, e $\bar{a}$ seu endereço, ambos de comprimento *L* ($L < Q$). $\bar{a}$ é adicionado componente a componente à mensagem do usuário.

Estas seqüências FSK podem ser representadas como entradas para uma matriz de *Q* linhas e *L* colunas (Einarson, 1980). A duração destas seqüências é denominada de *tempo de chip* (*chip time*). Como exemplo, considere o sistema com $Q = 7$ e $L = 5$, ilustrado na Figura 6.19(b). A matriz de transmissão corresponde à mensagem do usuário, tomada arbitrariamente como sendo $\bar{x} = (1,1,1,1,1)$, adicionada a seu endereço $\bar{a} = (3,0,6,1,4)$, resultando em uma palavra código $\bar{y} = (4,1,0,2,5)$. Note que, neste caso, o endereço da palavra código foi movido ciclicamente (adição de módulo 7).

A seqüência FSK pode ser afetada pelas seqüências de outros usuários e, também, por ruído, que foi desconsiderado nesta aplicação. Assim, apenas outros usuários poderiam interferir na transmissão e recepção dos dados.

O receptor é composto por um desespalhador de freqüência, que extrai o endereço da seqüência FSK recebida, e *Q* detectores de energia. Durante os *L* intervalos de chip, as saídas dos detectores são comparadas à um limiar e é preciso decidir se a freqüência correspondente está presente ou não.

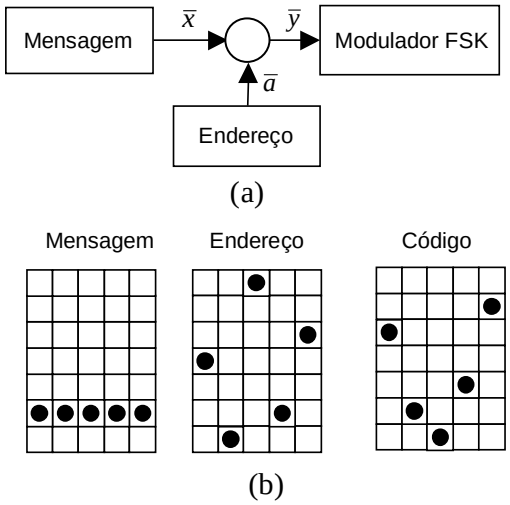


Figura 6.19. Receptor FH-CDMA de múltiplos níveis. (a) Diagrama de blocos. (b) Exemplo para $Q = 7$ e $L = 5$: $\bar{x} = (1,1,1,1,1)$, $\bar{a} = (3,0,6,1,4)$ e $\bar{y} = (4,1,0,2,5)$.

Um receptor de lógica majoritária com regra de máxima verosimilhança, para a mensagem *m*, é dado pela seguinte regra:

$$\max_{m} p(\bar{r} \mid \bar{y}_m), \qquad m = 0, 1, 2, \cdots, Q-1 \tag{6.2}$$

onde $\bar{r}$ é a matriz do receptor, $\bar{y}_m$ é uma das possíveis matrizes (transmitidas pelo usuário *m*), que está sendo codificada, e $p(\cdot \mid \cdot)$ é a probabilidade condicional.

Assim, um receptor de máxima verosimilhança, que minimiza a probabilidade de erro para mensagens de mesma probabilidade a priori, pode ser implementado por um receptor de regra majoritária que seleciona a mensagem $\bar{y}$ como a palavra código transmitida (matriz) com um maior número de entradas na matriz recebida.

6.4.1.2. Receptores Neurais para o Problema MLD

Diversas abordagens utilizando redes neurais artificiais foram propostas e testadas para a solução do problema de decisão de lógica majoritária. De Deus Jr. *et al.* (1999a) propuseram uma nova arquitetura neural e algoritmo de treinamento, especialmente desenvolvidos para resolver o problema de decisão de lógica majoritária. Esta rede, denominada, NTNET (*non-trainable neural network*), é composta por 2 módulos, onde o primeiro executa a soma das linhas da matriz, e o segundo módulo é competitivo, sendo responsável pela determinação da mensagem decodificada. A NTNET permite o projeto do receptor de lógica majoritária com um conjunto de neurônios do tipo threshold. Outras arquiteturas de RNAs também foram empregadas para comparação (de Deus Jr. *et al.* (1999a,b)): o perceptron de múltiplas camadas (MLP), treinado com o algoritmo de backpropagation, e um método de poda para os mapas auto-organizáveis de Kohonen com grid de saída uni-dimensional (PSOM) proposto por de Castro & Von Zuben (1999b). Note que estas arquiteturas de rede neural operam com conexões assumindo valores reais, enquanto a ABNET e a NTNET possuem pesos binários para as conexões. Para aplicar o MLP a este problema, os 7 padrões de entrada foram rotulados, de forma que o MLP pudesse mapeá-los em diferentes unidades de saída.

Para comparar o desempenho dos 4 algoritmos de treinamento de RNA: MLP, NTNET, PSOM e ABNET, considere um sistema com $Q = 7$ e $L = 3$. Neste caso, resolver o problema de decisão de lógica majoritária implica em projetar um receptor com um desempenho, em termos de probabilidade de erro de palavra, como dado pela Figura 6.20.

Todas as redes foram capazes de resolver o problema, ou seja, atingir a probabilidade de erro de palavra desejada. Entretanto, seus algoritmos de treinamento são completamente distintos, tornando difícil uma comparação de desempenho em termos de custo computacional e acurácia dos resultados obtidos. Assim, as diferentes redes neurais foram comparadas em relação à complexidade de suas arquiteturas resultantes, ou seja, quantidade de conexões (pesos e limiares) ao final do processo de treinamento.

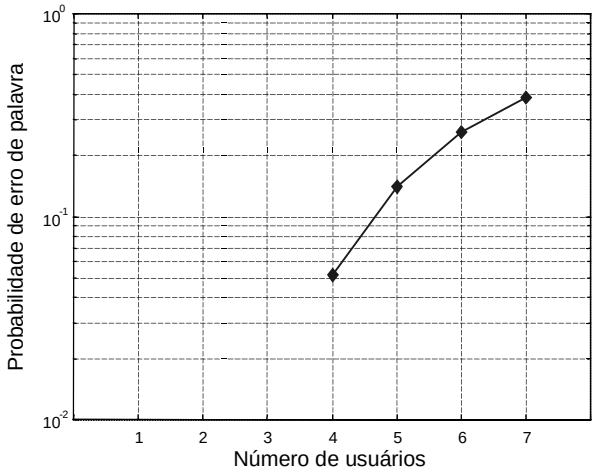


Figura 6.20. Desempenho do problema MLD para $Q = 7$ e $L = 3$. A probabilidade de que uma mensagem enviada seja decodificada corretamente, denominada probabilidade de erro de palavra, é igual a 0 para um número de usuários inferior a 4.

A rede MLP com arquitetura mais simples capaz de resolver o problema continha 181 conexões, com 21 entradas, 6 unidades intermediárias e 7 saídas. A NTNET precisou de 21 conexões. Se desconsiderarmos as conexões cujos pesos associados assumem o valor 0, a arquitetura resultante da ABNET também apresenta apenas 21 conexões, de acordo com a Figura 6.21. O algoritmo de treinamento do PSOM resultou em uma rede com 147 conexões de valores reais (21×7), entretanto, os pesos do PSOM podem ser quantizados (decodificados) para assumirem os valores {0,1} e, caso as conexões com pesos associados 0 sejam descartadas, a rede resultante também apresentará 21 conexões.

A Tabela 6.6 resume a comparação de desempenho dos algoritmos em relação à complexidade de suas arquiteturas resultantes. Esta tabela também mostra que a representação binária intrínseca dos pesos, faz com que a NTNET e a ABNET sejam particularmente adequadas para a solução do problema MLD.

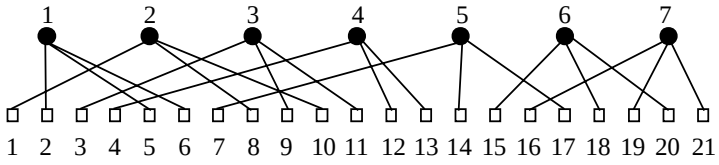


Figura 6.21. Exemplo de ABNET resultante para o problema MLD, onde apenas as conexões com pesos associados correspondentes a 1 foram apresentadas. A rede possui 21 entradas e 7 saídas, resultando em um vetor de pesos de comprimento 21.

Tabela 6.6. Complexidade das redes (MLP, NTNET, PSOM e ABNET) para resolver o problema MLD. A quantidade final de conexões apresentada para a ABNET e PSOM são tomadas após a eliminação dos pesos com valores nulos.

Rede Neural	Número de Conexões	Complexidade da Arquitetura	Representação Binária
MLP	181	Alta	Não
NTNET	21	Baixa	Sim
PSOM	21	Baixa	Não
ABNET	21	Baixa	Sim

6.4.2. Classificação de Lentes de Contato

Este problema de reconhecimento (classificação) de padrões foi utilizado por Witten & McDonald (1988) para estudar a capacidade de aquisição de conhecimentos. O banco de dados está disponível para download no repositório de banco de dados para aprendizagem de máquina da Universidade da Califórnia (URL 3). Nove regras cobrem o conjunto de treinamento e os exemplos são completos, sem ruídos e constituem uma simplificação do problema. Estão disponíveis $M = 24$ instâncias com $L = 4$ atributos, mapeadas em 3 classes.

Trata-se de um problema simples de classificação e nosso objetivo é testar a capacidade da ABNET em aprender os padrões de entrada. É importante salientar que este problema não é binário, requerendo uma modelagem binária (quantização) dos dados para que a ABNET possa ser aplicada. As entradas originais assumem apenas os valores {1,2,3}, que serão representados por 1: [1 1 1]; 2: [0 1 0] e 3: [0 0 1], correspondentes a uma quantidade total de 12 bits para representar cada amostra. Os números 1, 2 e 3 poderiam ter sido modelados com apenas 2 bits, mas isso resultaria em números com diferentes distâncias de Hamming (HD) entre eles, por exemplo, se 1: [1 1]; 2: [0 1] e 3: [0 0], a HD entre os números 1 e 2 seria HD = 1, enquanto entre os números 1 e 3 seria HD = 2. A quantização escolhida mantém uma distância de Hamming simétrica entre os três valores.

A ABNET foi comparada com uma rede competitiva (Seção 4.2.2.2) denominada RC, com o mapa auto-organizável de Kohonen com grid de saída uni-dimensional (Seção 4.2.2.2.1) denominado SOM, e com o PSOM, proposto por de Castro & Von Zuben (1999b), para a rede de Kohonen. A RC apresenta um número fixo de saídas N, pré-especificado antes do treinamento. Os resultados apresentados na Tabela 6.7 são a média e o desvio padrão, obtidos a partir de 10 simulações, do percentual de classificação correta (PCC) das diferentes RNAs avaliadas. Os resultados da ABNET levam em consideração o limiar de afinidade ε e arquiteturas com dimensão final distinta. O algoritmo de treinamento do PSOM requer a especificação do limiar de poda ξ e a quantidade final de neurônios na rede também é variável. Como as redes RC, SOM e PSOM possuem pesos ou conexões com valores reais, as entradas foram tomadas assumindo seus valores originais, porém normalizadas no intervalo [-1,1].

Tabela 6.7. Aplicação das redes RC, SOM, ABNET e PSOM ao problema de classificação de lentes de contato. PCC é o percentual de classificação correta, *N* a quantidade de unidades de saídas da rede, ε o limiar de afinidade e ξ o limiar de poda do PSOM. Os resultados apresentados são a média e desvio padrão obtidos a partir de 10 simulações. No casos das redes RC e SOM, foram escolhidos arbitrariamente diferentes valores para *N*.

RC		**SOM**		**ABNET**			**PSOM**		
N	PCC (%)	*N*	PCC (%)	ε	*N*	PCC (%)	ξ	*N*	PCC (%)
24	92.50 ± 1.75	24	97.08 ± 3.43	0	24 ± 0.00	100 ± 0.00	0.001	13.7 ± 1.30	95.00 ± 4.30
19	92.50 ± 1.75	19	96.67 ± 2.64	1	24 ± 0.00	100 ± 0.00	0.005	12.7 ± 1.06	92.08 ± 4.99
14	92.92 ± 2.01	14	96.25 ± 3.07	2	21.9 ± 0.88	97.08 ± 3.43	0.01	13.2 ± 1.81	92.92 ± 4.83
9	92.09 ± 1.32	9	92.09 ± 1.32	3	21 ± 1.41	95.83 ± 3.93	0.05	8 ± 0.00	91.67 ± 0.00
4	75.00 ± 4.81	4	77.50 ± 8.83	4	18.2 ± 1.23	93.75 ± 4.05	0.1	8 ± 0.00	91.67 ± 0.00

Pode ser observado que, devido à quantização dos dados de entrada, a ABNET apresentou as arquiteturas finais menos parcimoniosas, quando comparadas ao algoritmo PSOM, por exemplo. Por outro lado, a ABNET demonstrou ser muito poderosa na aquisição de conhecimentos, sendo capaz de classificar os padrões de entrada corretamente com 100% de acerto para um limiar de afinidade de 0 ou 1. Nestes casos, a rede evoluída contém anticorpos altamente específicos aos antígenos a serem reconhecidos.

6.4.3. Reconhecimento de Caracteres Binários

Este problema objetiva avaliar a tolerância a ruídos e capacidade de generalização da ABNET, quando caracteres binários são apresentados à rede. Neste caso, a ABNET foi comparada com a rede de Hopfield discreta, descrita na Seção 4.2.2.3 e utilizando a regra de aprendizagem da pseudo-inversa.

Os padrões de entrada utilizados para testar a tolerância a ruídos da rede foram artificialmente gerados e estão apresentados na Figura 6.22. Um ruído aleatório foi inserido nas amostras simplesmente revertendo-se bits em 0 em bits em 1, e vice-versa. Foram testados 4 níveis de ruído distintos: 5.7%, 11.4%, 20% e 40%, correspondendo à troca de 2, 4, 7 e 14 bits, respectivamente. A ABNET foi testada para diferentes valores do limiar de afinidade ε (ε = 0, 1, 2, 3, 4 e 5) e uma classificação correta é considerada quando a rede mapeia o padrão corrompido (com ruído) no mesmo neurônio de saída em que foi mapeado o padrão corresponde na ausência de ruído. No caso da rede de Hopfield, uma classificação correta é considerada quando a distância de Hamming entre o padrão restaurado e o armazenado é menor do que δ, onde δ = 0, 1, 2, 3, 4 e 5.

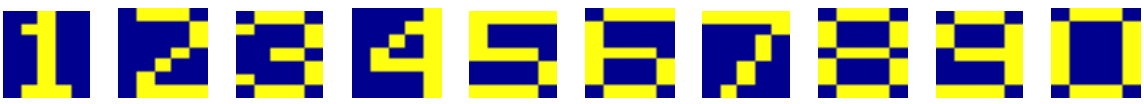
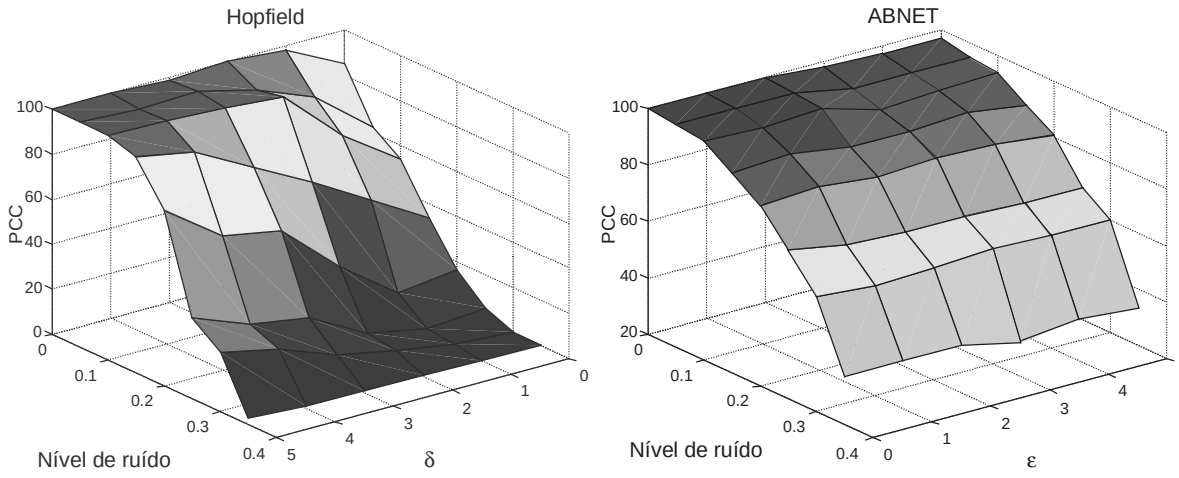


Figura 6.22. 10 caracteres com resolução 7×5 empregados na avaliação da tolerância a ruídos da ABNET.

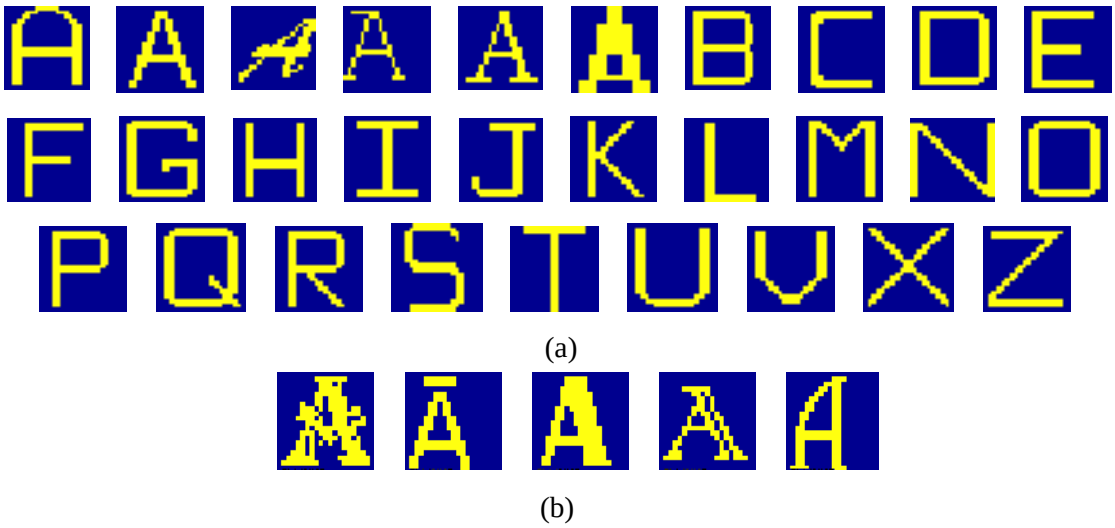


(a) (b)

Figura 6.23. Tolerância a ruídos (percentual de classificação correta – PCC). (a) Rede de Hopfield, onde δ é a distância de Hamming permitida entre o padrão recuperado e o padrão armazenado. (b) ABNET, onde ε é o limiar de afinidade da rede.

A Figura 6.23 apresenta os resultados da rede de Hopfield e ABNET quando aplicadas ao problema de reconhecimento de caracteres binários. É possível observar que, caso seja requerido um reconhecimento perfeito pela rede de Hopfield ($\delta = 0$) entre o padrão armazenado e o corrompido, seu desempenho apresenta uma forte degradação com o nível de ruído. A ABNET demonstrou ser menos sensível ao nível de ruído do que a rede de Hopfield. É importante ressaltar que os parâmetros δ e ε não são diretamente comparáveis. Enquanto δ representa a acurácia do estado estacionário alcançado pela rede de Hopfield ao final do processo iterativo de treinamento (medindo a existência de estados espúrios), ε indica o limiar de afinidade necessário para a aprendizagem da ABNET. Entretanto, para efeitos práticos, o papel desempenhado por estes parâmetros nos problemas de classificação é equivalente.

Para avaliar a capacidade de generalização da ABNET, foi empregado um conjunto de caracteres distintos do anterior, como ilustrado na Figura 6.24. Este conjunto amostral é composto por $M = 29$ caracteres de comprimento $L = 400$ (resolução 20×20), onde 6 deles correspondem a diferentes fontes da letra A, e os outros representam outras letras do alfabeto. A idéia é treinar as redes ABNET e Hopfield com os dados da Figura 6.24(a) e verificar se elas são capazes de reconhecer os caracteres da Figura 6.24(b), que são diferentes fontes para a letra A, como pertencentes a grupos representativos dos caracteres A. Para isso, é necessário que estas redes sejam capazes de criar estes grupos representativos da letra A.



(a) (b)

Figura 6.24. Caracteres para testar a capacidade de generalização da ABNET. (a) Amostras do alfabeto com 6 letras A distintas. (b) Conjunto de teste independente.

A Figura 6.25(a) ilustra os padrões recuperados pela rede de Hopfield e a associação de padrões realizada pela ABNET. Pode ser visto que a rede de Hopfield não foi capaz de recuperar um dos caracteres, resultando em um estado espúrio. A ABNET agrupou as diferentes fontes de A em dois grupos representativos do caractere A, resultando em uma classificação correta.

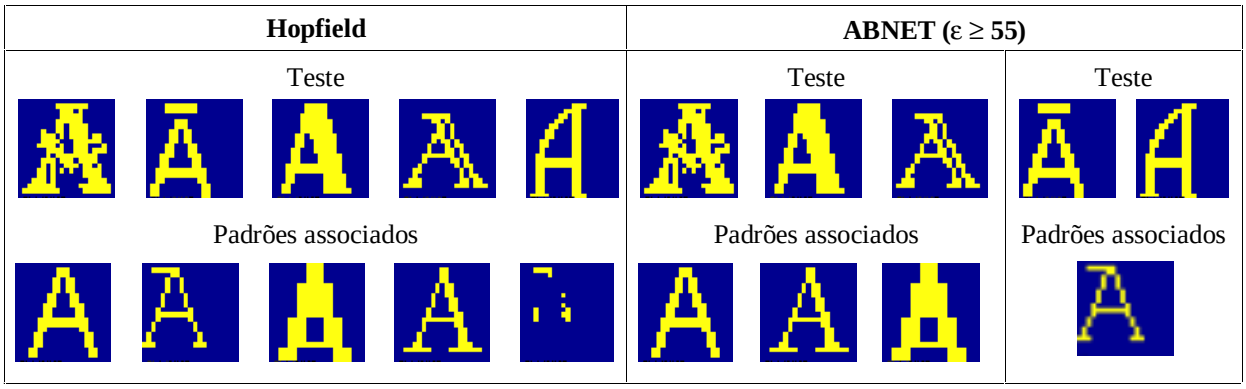


Figura 6.25. Resultados obtidos para as redes de Hopfield e ABNET aplicadas ao problema de capacidade de generalização para caracteres binários. (a) Hopfield. (b) ABNET.

6.4.4. Análise de Sensibilidade

O pseudocódigo para o algoritmo de treinamento proposto para a ABNET na Seção 5.4.3 apresenta a seguinte linha de comando `[Ab,F]=abnet(Ag,L,gen,ε,α0,β)`. Assim como para o algoritmo CLONALG discutido anteriormente, os dados de treinamento correspondem ao conjunto antigênico **Ag** de comprimento L e, portanto, são determinados pelo problema a ser estudado. Para todos os problemas abordados, adotou-se $\beta = 3$ e $\alpha_0 = 3$.

No caso da ABNET, a quantidade mínima de gerações para o treinamento da rede merece uma análise um pouco mais cuidadosa do que no caso do algoritmo CLONALG. Por se tratar de uma rede construtiva cujos neurônios são inseridos a cada β iterações (ou gerações), é preciso que a quantidade `gen` de gerações de treinamento seja suficiente para que o processo construtivo da rede permita a inserção de todos os neurônios necessários à aprendizagem e classificação correta do conjunto de treinamento. Na Seção 3.4.2.1 foi demonstrado analiticamente que a cobertura de cada anticorpo em um espaço de formas binário de Hamming pode ser determinada pela seguinte expressão

$$C = \sum_{i=0}^{\varepsilon} \binom{L}{i} = \sum_{i=0}^{\varepsilon} \frac{L!}{i!(L-i)!}, \qquad \textbf{(6.3)}$$

onde C é a cobertura de cada anticorpo e L o comprimento das cadeias.

Para cadeias de comprimento L, é possível gerar um conjunto total de 2^L cadeias distintas e, portanto, a quantidade máxima N_{max} de anticorpos que precisam ser inseridos na ABNET para que ela seja capaz de apresentar a cobertura máxima do espaço de formas pode ser dada pela Equação (6.4), que é função do limiar de afinidade ε.

$$N_{max} = \frac{2^L}{C}. \qquad \textbf{(6.4)}$$

Ou seja, N_{max} corresponde ao número total de cadeias que podem ser geradas dividido pela cobertura de cada anticorpo. Se N_{max} for maior do que a quantidade M de antígenos a serem reconhecidos ($N_{max} > M$), então $N_{max} = M$, já que não é necessário mais do que um anticorpo para reconhecer cada antígeno do conjunto amostral. Dado N_{max}, e considerando que a cada β gerações um anticorpo é verificado para poder entrar na rede, a seguinte regra pode ser derivada para a determinação da quantidade mínima `gen` de gerações necessária para que a ABNET convirja para uma rede de dimensão apropriada

$$\texttt{gen} = (\beta + 1)\, N_{max}. \qquad \textbf{(6.5)}$$

Para o primeiro problema testado na Seção 6.4.1, o limiar de afinidade foi assumido nulo, indicando que um reconhecimento perfeito entre um antígeno e um anticorpo faz-se necessário para que a ABNET classifique os dados de treinamento. No caso dos dois outros problemas abordados, Seções 6.4.2 e 6.4.3, o desempenho da rede foi avaliado para diferentes valores de ε. No problema de classificação de lentes de contato, os resultados apresentados na Tabela 6.7 indicam que ε influencia na dimensão final da rede e também no seu percentual de classificação correta.

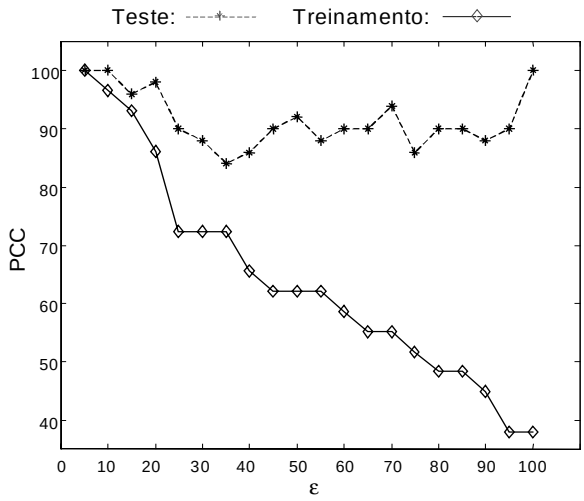


Figura 6.26. Capacidade de generalização da ABNET (percentual de classificação correta – PCC) como função do limiar de afinidade ε para os padrões de treinamento e teste apresentados na Figura 6.24.

A Figura 6.26 mostra como a capacidade de generalização da ABNET varia em relação ao limiar de afinidade ε. Esta figura mostra que a classificação para os padrões de treinamento sofre uma degradação com ε, mas a capacidade de generalização para a classe A não sofre uma degradação significativa, mantendo uma classificação correta em torno de 92%. Este fenômeno permite concluir que a ABNET está mantendo pelo menos uma classe separada para os padrões A, e está misturando os outros caracteres. Isto pode ser esperado se considerarmos que a classe A é predominante no conjunto amostral.

Por último, a taxa inicial de hipermutação α_0 exerce influência na velocidade de convergência do algoritmo de treinamento da ABNET. Para avaliar como se dá esta influência, foi feito o seguinte experimento. Uma ABNET foi inicializada com $N = 10$ anticorpos de comprimento $L = 35$ com o objetivo de aprender os caracteres da Figura 6.22 para $\varepsilon = 0$. Como a ABNET possui a quantidade mínima de anticorpos necessária para reproduzir os dados de treinamento para $\varepsilon = 0$, resta-lhe a tarefa de aprender o conjunto amostral. A Figura 6.27 mostra a sensibilidade do algoritmo em relação ao parâmetro α_0 para os dados de treinamento da Figura 6.22, $\varepsilon = 0$ e $N = 10$. É possível verificar que para valores de $\alpha_0 < 6$ a velocidade de convergência da rede é menor. Também é possível perceber que valores elevados de α_0 não reduzem significativamente o tempo de treinamento da rede.

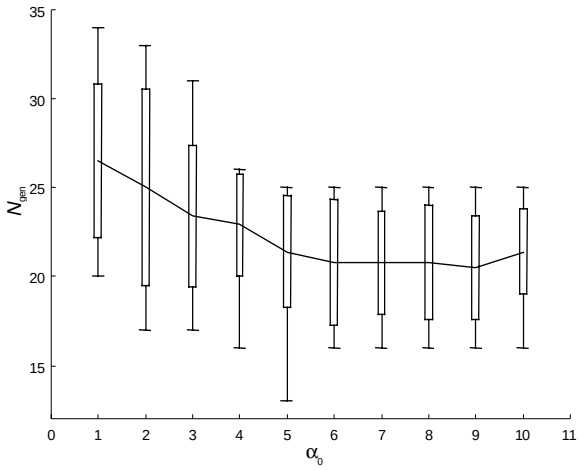


Figura 6.27. Sensibilidade da ABNET em relação ao parâmetro α_0 para os dados de treinamento da Figura 6.22, $\varepsilon = 0$ e $N = 10$. Valores máximo, mínimo, média e desvio padrão da quantidade de épocas necessária para convergência (N_{gen}).

6.4.5. Discussão

As redes neurais testadas não apresentaram dificuldade para solucionar o problema de decisão de lógica majoritária. Entretanto, a principal vantagem da ABNET sobre as outras redes neurais testadas, é que ela representa uma arquitetura genérica, com estratégia de aprendizagem inerentemente Booleana, que pode ser aplicada a diversos problemas. A outra rede que apresentou bom desempenho para este problema foi a NTNET, especialmente projetada para resolver o problema MLD.

No exemplo descrito na Seção 6.4.2, a ABNET foi capaz de realizar a aquisição de conhecimentos com altas taxas de reconhecimento de padrões. Também foi possível observar que, como proposto na teoria (Seção 5.4) e verificado na análise de sensibilidade (Seção 6.4.4), valores pequenos de ε (0 ou 1) implicam em uma rede resultante com anticorpos altamente específicos aos estímulos antigênicos e muitas unidades são inseridas para que a rede possa representar apropriadamente os dados de treinamento. Quanto mais generalistas os anticorpos, menos unidades são inseridas e mais parcimoniosa é a rede.

A ABNET apresenta algumas vantagens sobre a rede de Hopfield discreta na solução de problemas de reconhecimento de caracteres binários. Ela implementa uma espécie de classificador ótimo de erro mínimo quando ruído é inserido de forma aleatória e independente nos padrões de treinamento. A ABNET também necessitou de menos conexões do que a rede de Hopfield para o problema testado: 400×400 (160.000) conexões na rede de Hopfield, contra 400×29 (11.600) conexões da ABNET para $\varepsilon = 0$. A ABNET também pode ser modificada para tornar-se um classificador ótimo de erro mínimo quando os padrões de entrada são bipolares. Além disso, ela não apresenta estados espúrios.

Durante a apresentação do algoritmo de backpropagation, Rumelhart *et al.* (1986) discutiram 7 problemas, dos quais podemos citar, por conveniência, 6: XOR, paridade, codificação, simetria, adição e negação. Para todos estes exemplos, a rede a ser treinada

pode ser classificada como Boolena, onde a informação a ser adquirida está codificada sob a forma de cadeias binárias. Devido ao padrão de conexões de ABNET e ao seu algoritmo de treinamento, ela é capaz de resolver problemas como estes de forma direta e eficiente, enquanto algumas redes que operam com pesos reais, como o MLP treinado com o algoritmo de backpropagation, teriam dificuldades de convergência e, além disso, demandariam um esforço computacional muito superior.

Finalmente, a implementação em hardware da ABNET é simplificada, pois ela não estaria sujeita aos efeitos de quantização do conjunto de pesos e seu algoritmo de treinamento opera diretamente com valores binários.

6.5. aiNet

Para avaliar o desempenho da rede imunológica artificial proposta, denominada aiNet, ela foi aplicada a diversos problemas de teste e seu desempenho comparado com os mapas auto-organizáveis de Kohonen (SOM – *self-organizing map*) com grid unidimensional de saída (ver Seção 4.2.2.2). Os parâmetros de treinamento do SOM são: valor inicial da taxa de aprendizagem $\alpha_0 = 0.9$, valor final da taxa de aprendizagem $\alpha_f = 10^{-3}$, com um decrescimento geométrico de 0.9 a cada 5 iterações. Os pesos foram inicializados empregando-se uma distribuição uniforme no intervalo [–0.1,0.1]. A quantidade de neurônios *m* do grid de saída irá depender do problema a ser abordado. Ao final do processo de treinamento do SOM, aquelas unidades de saída que não forem responsáveis pela classificação de nenhum dado de entrada serão removidas. Dentre as características avaliadas, destacam-se a capacidade de compressão de informação (amostras de treinamento) medida pelo índice *CR* (*compression rate*), e de separação e representação de clusters. Uma versão deste algoritmo também foi empregada na determinação da quantidade e localização de centros de redes neurais com funções de ativação de base radial (RBF), discutidas na Seção 4.2.2.1.

6.5.1. Análise de Dados

Foram avaliados quatro problemas de teste, como ilustrados na Figura 6.28, CLASSLS, CLASSNLS, SPIR e CHAINLINK.

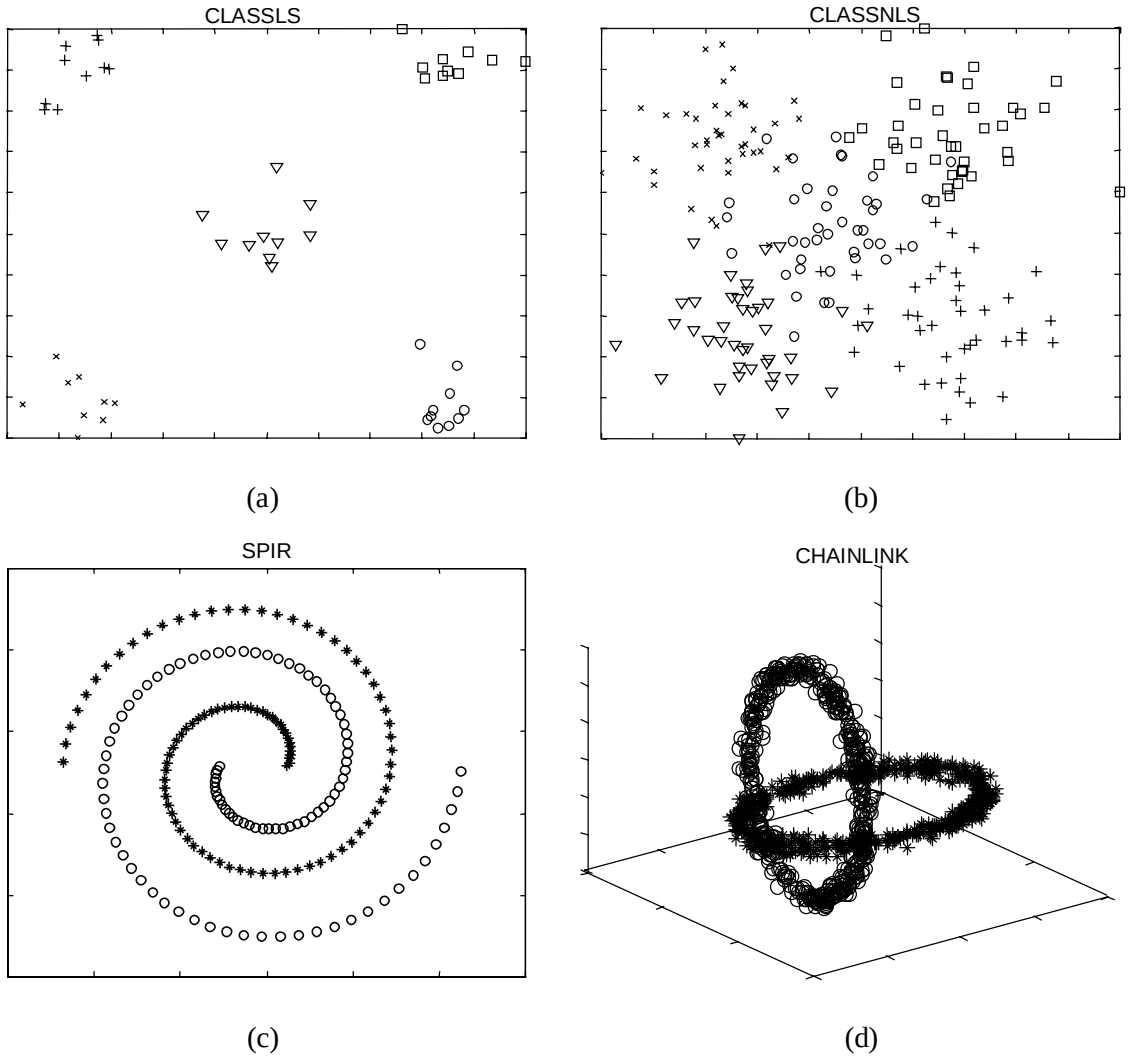


Figura 6.28. Antígenos (dados de treinamento) para avaliação de desempenho do algoritmo aiNet. (a) CLASSLS, $M = 50$. (b) CLASSNLS, $M = 200$. (c) SPIR, $M = 190$. (d) CHAINLINK, $M = 500$.

6.5.1.1. CLASSLS

Este problema é composto por cinco classes linearmente separáveis, cada qual contendo 10 amostras e está ilustrado na Figura 6.28(a). As amostras foram geradas utilizando-se distribuições gaussianas com diferentes médias e variâncias fixas. O objetivo é ilustrar a capacidade da aiNet em separar, definir e representar clusters, construir a arquitetura final da rede e reduzir a complexidade do problema, ou seja, comprimir o conjunto amostral, o qual será apresentado à aiNet sem os rótulos, ou seja, sem o conhecimento prévio da classe a que cada dado pertence .

Considerando o algoritmo de treinamento apresentado na Seção 5.5.3, a seguinte linha de comando foi utilizada: `[Ab`$_m$`,S]=aiNet(Ag,2,10,4,0.2,1,0.14,10)`.

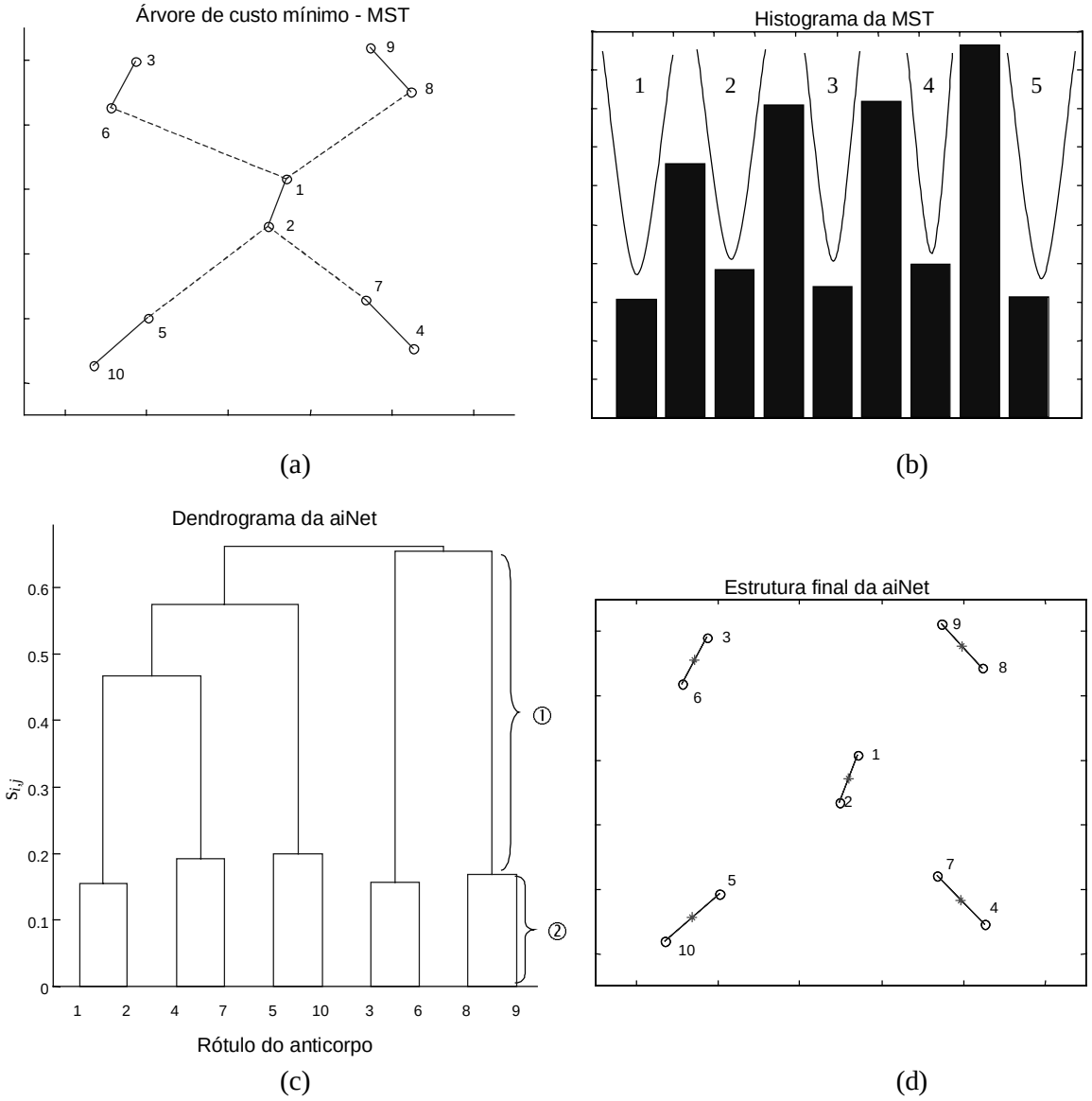


Figura 6.29. Resultados apresentados pela aiNet para o problema CLASSLS. (a) Árvore de custo mínimo (MST), onde as linhas tracejadas indicam conexões a serem removidas. (b) Histograma da MST. (c) Dendrograma da rede indicando grandes diferenças entre os valores de fusão (① em relação a ②). (d) Arquitetura final da aiNet com os cinco clusters representados pelos seus respectivos centros de massa.

A Figura 6.29(a) apresenta a árvore de custo mínimo (MST) resultante, representando uma rede composta por $m = 10$ anticorpos de memória, o que corresponde a uma compressão $CR = 80\%$ do conjunto amostral. As linhas tracejadas indicam conexões inconsistentes, a serem detectadas pelo histograma da MST e removidas (podadas). A quantidade de vales na Figura 6.29(b) permite identificar 5 clusters distintos no conjunto de dados. Também é possível utilizar o critério do tamanho do passo (*stepsize*) para empregar o dendrograma da aiNet na definição dos clusters resultantes da rede.

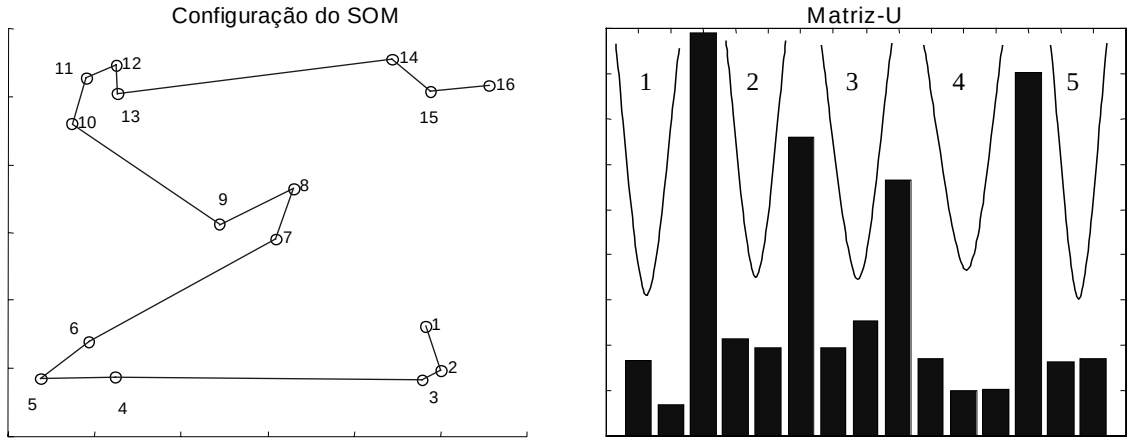
Tabela 6.8. Valores de pertinência de cada anticorpo c_i, $i = 1,...,10$, da aiNet em relação ao centróide v_j, $j = 1,...,5$, de cada cluster.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
v_1	1.00	1.00	0.67	0.71	0.76	0.69	0.84	0.75	0.71	0.66
v_2	0.58	0.63	0.50	1.00	0.68	0.50	1.00	0.60	0.56	0.64
v_3	0.67	0.50	0.63	0.56	0.50	0.58	0.57	1.00	1.00	0.50
v_4	0.50	0.55	0.54	0.62	1.00	0.57	0.64	0.50	0.50	1.00
v_5	0.60	0.50	1.00	0.50	0.59	1.00	0.50	0.60	0.56	0.64

Um passo de tamanho igual a 1.5 seria adequado para classificar corretamente os anticorpos da rede, como pode ser visto na Figura 6.29(c). A Figura 6.29(d) apresenta a arquitetura final da aiNet com cinco clusters distintos, onde cada um deles está representado pelo seu respectivo centro de massa, ou centróide (estrelas).

Uma vez definida a arquitetura final da rede (Figura 6.29(d)) e os clusters representados pelos seus respectivos centros de massa, é possível aplicar os conceitos de clusterização gradual discutidos na Seção 5.5.4.2 para a aiNet resultante. A matriz de proximidades **U** especificando o grau de pertinência de cada anticorpo da rede em relação aos centróides determinados é apresentada na Tabela 6.8. A partir desta tabela percebe-se que os anticorpos c_1 e c_2 pertencem ao cluster v_1 com pertinência 1.00, ao cluster v_2 com pertinência 0.58 e 0.63 ($u_{2,1}$ e $u_{2,2}$) respectivamente, e assim por diante.

Considere agora o caso de aplicar um SOM unidimensional ao mesmo problema. Os resultados estão apresentados na Figura 6.30 para uma quantidade final de saídas $m = 16$ ($m_0 = 20$). É possível verificar que esta rede também é capaz de solucionar este problema.



(a) (b)

Figura 6.30. Aplicação de um SOM para o problema CLASSLS. (a) Posição final dos vetores de pesos e ligação com seus vizinhos mais próximos. (b) Matriz-U com cinco vales, cada qual representando um cluster diferente.

6.5.1.2. CLASSNLS

O problema ilustrado na Figura 6.28(b) é composto por 5 classes não-linearmente separáveis, onde cada classe possui 40 amostras de treinamento. Como pode ser observado, se os dados não estivessem rotulados nesta figura, como no caso dos modelos avaliados, seria difícil identificar diferenças entre os grupos de dados, até mesmo para um observador humano. Este problema foi utilizado por de Castro & Von Zuben (1999b) para avaliar o desempenho de um método de poda, denominado PSOM, para os mapas auto-organizáveis de Kohonen.

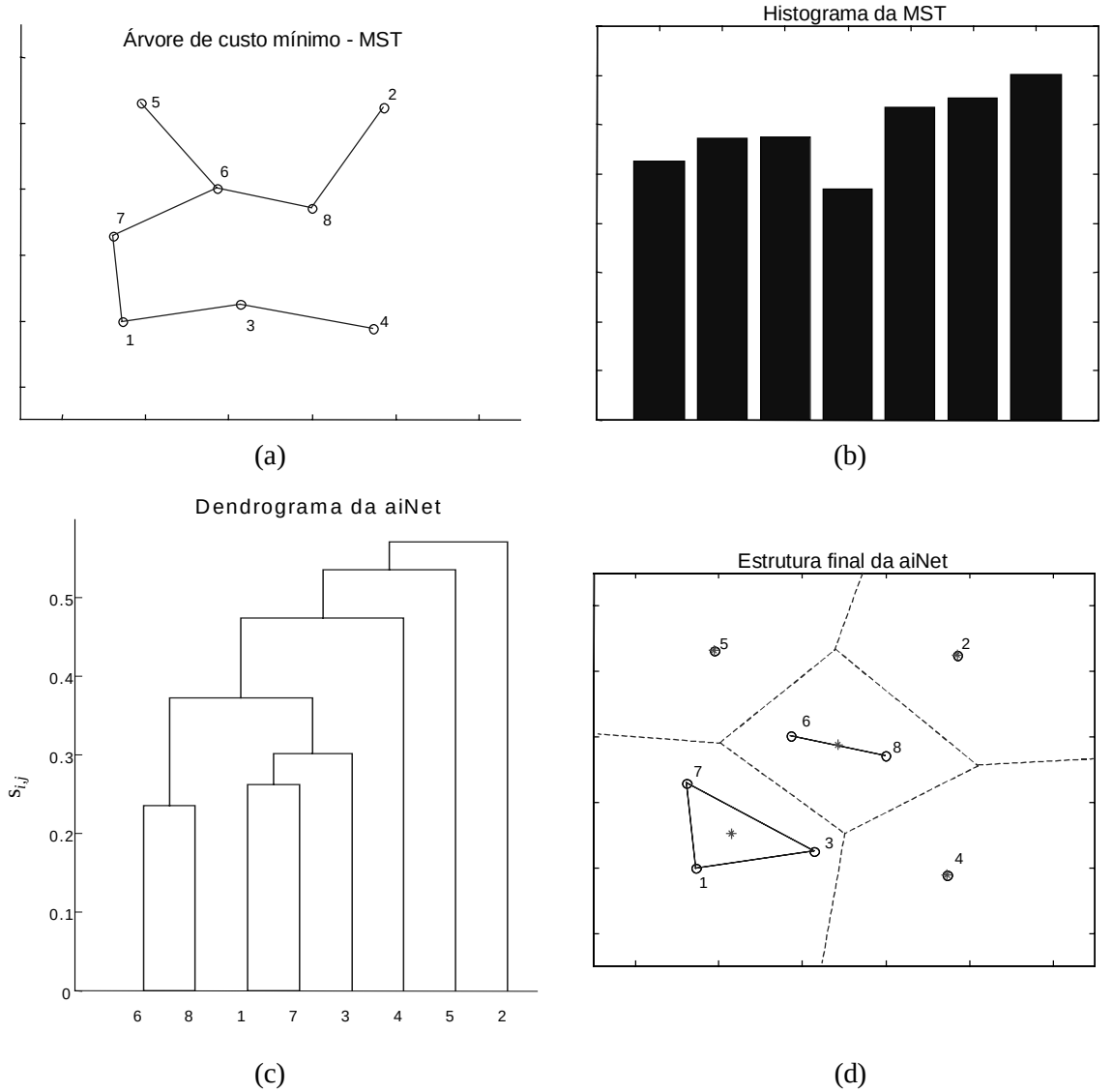


Figura 6.31. aiNet aplicada ao problema CLASSNLS. (a) Árvore de custo mínimo (MST). (b) Histograma da MST. (c) Dendrograma da rede indicando os cinco clusters de anticorpos. (d) Arquitetura final da aiNet com os cinco clusters representados pelos seus respectivos centros de massa e o diagrama de Voronoi particionando o espaço de formas em relação aos centros de massa de cada cluster.

Foram apresentados resultados referentes ao percentual de classificação dos dados de treinamento para diferentes dimensões do PSOM. Estes resultados serão reproduzidos objetivando comparar a classificação apresentada pela aiNet com a classificação obtida pelo PSOM.

Considerando o algoritmo de treinamento da Seção 5.5.3, a seguinte linha de comando foi utilizada: `[`$\mathbf{Ab}_m$`,S]=aiNet(Ag,2,10,4,0.2,1,0.2,10)`. A aiNet apresentou uma taxa de compressão CR = 96%, correspondendo a m = 8 anticorpos de memória na rede.

Como discutido na Seção 5.5.4, quando a estrutura final da rede está próxima da quantidade de clusters existente no conjunto antigênico (5 para este problema), o método da árvore de custo mínimo (MST) apresenta dificuldades para detectar e separar os clusters da aiNet. Neste caso, a utilização do dendrograma como critério de decisão se apresenta como uma boa alternativa. Como pode ser observado na Figura 6.31(b), nada pode ser concluído quanto à quantidade de clusters da aiNet partindo-se do histograma da MST. Entretanto, analisando a Figura 6.31(c), é possível determinar quantos clusters resultaram na aiNet e quais anticorpos pertencem a cada um deles: basta particionar o dendrograma tomando $s_{i,j} \approx 0.3$. A Figura 6.31(d) apresenta a arquitetura final da rede e as superfícies de decisão determinadas pelo diagrama de Voronoi dos anticorpos resultantes. Os clusters estão representados pelos seus respectivos centros de massa. A Figura 6.32 apresenta o diagrama de Voronoi da Figura 6.31(d) plotado sobre os dados de entrada.

A Tabela 6.9 apresenta os valores de pertinência de cada anticorpo da rede (c_i, i = 1,...,8) em relação aos cinco clusters (v_i, i = 1,...,5) detectados.

A Tabela 6.10 reproduz os resultados apresentados por de Castro & Von Zuben (1999b) adicionados do percentual de classificação obtido pela aiNet. O PSOM e a aiNet possuem estruturas cuja dimensão varia dinamicamente ao longo do processo de treinamento, enquanto o SOM possui um grid de saída com tamanho definido a priori. É possível verificar que o PSOM e a aiNet não são capazes de determinar arquiteturas com todas as dimensões listadas na Tabela 6.10; o que é representado pelas linhas tracejadas.

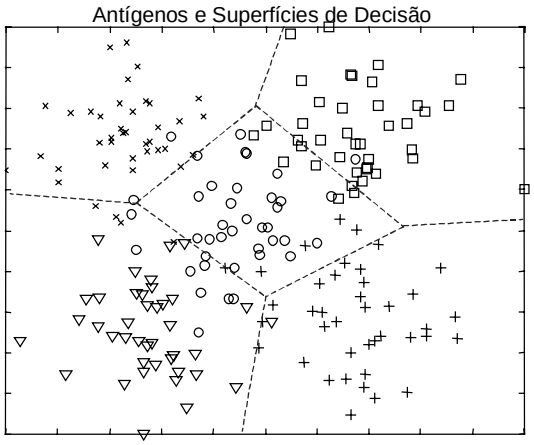


Figura 6.32. Superfícies (hiperplanos) de decisão dados pelo diagrama de Voronoi aplicado à aiNet resultante na Figura 6.31(d) e antígenos da Figura 6.28(b).

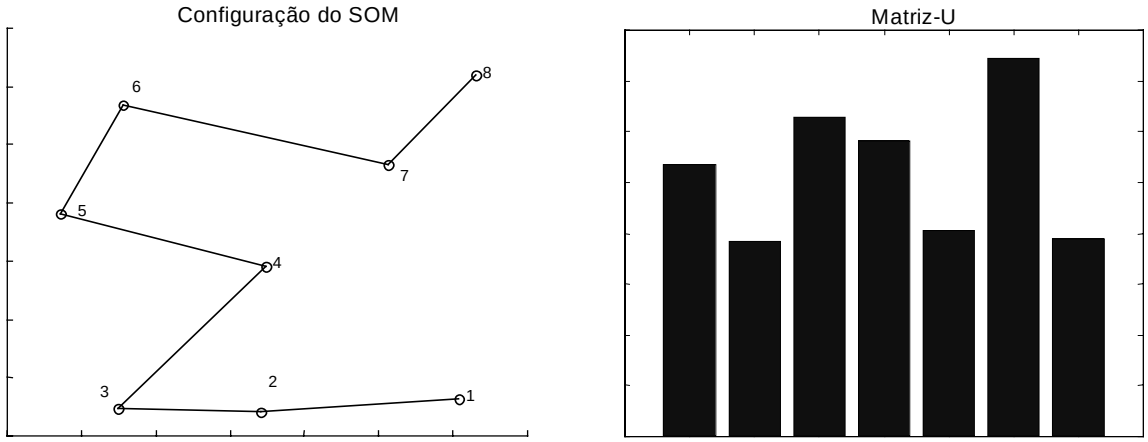
Tabela 6.9. Valores de pertinência de cada anticorpo c_i, i = 1,...,8, em relação ao centróide de cada cluster v_j, j = 1,...,5.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
v_1	1.00	0.50	1.00	0.50	0.50	0.86	1.00	0.63
v_2	0.80	0.50	1.00	0.50	0.50	1.00	0.97	1.00
v_3	0.65	0.50	0.55	0.50	1.00	0.89	0.94	0.50
v_4	0.71	0.50	1.00	1.00	0.50	0.50	0.59	0.72
v_5	0.50	1.00	0.50	0.50	0.50	0.63	0.50	0.80

Tabela 6.10. Comparação de desempenho entre os algoritmos SOM, PSOM e aiNet quanto à capacidade de classificação correta do dados de treinamento do problema CLASSNLS, onde *m* é a dimensão final da rede (de Castro & Von Zuben, 1999b).

	Classificação Correta (%)				
m	6	7	8	9	10
SOM	79.0	74.0	80.5	76.5	82.5
PSOM	--------	79.5	---------	---------	83.0
aiNet	--------	--------	83.5	67.0	76.5

Para os mesmos dados de treinamento adotados no problema anterior, os resultados da aplicação do SOM a este problema estão ilustrados na Figura 6.33, assumindo m = 8. Note que não é possível inferir a quantidade de clusters determinados partindo da matriz-U. Sendo assim, para avaliar o desempenho do SOM treinado, aplicou-se os dados de treinamento à rede e verificou-se qual unidade de saída era a vencedora para cada padrão. As unidades que venceram mais para os padrões de determinada classe eram consideradas as unidades representativas daquela classe, permitindo avaliar o percentual de classificação correta do SOM.



(a) (b)

Figura 6.33. Resultado da aplicação do SOM ao problema CLASSNLS. (a) Configuração final dos vetores de pesos e ligação com seus vizinhos mais próximos. (b) Matriz-U.

6.5.1.3. SPIR

O problema das duas espirais, denominado SPIR, está ilustrado na Figura 6.28(c). O conjunto de antígenos é composto por $M = 190$ elementos divididos em duas classes de mesma cardinalidade em um espaço de formas Euclidiano de dimensão $L = 3$. O objetivo deste problema é avaliar a capacidade da aiNet em distinguir clusters que não são linearmente separáveis, mas que apresentam uma distância relativa entre as classes. Os parâmetros de treinamento foram `[Ab`$_m$`,S]=aiNet(Ag,3,50,4,0.1,1,0.08,10)`.

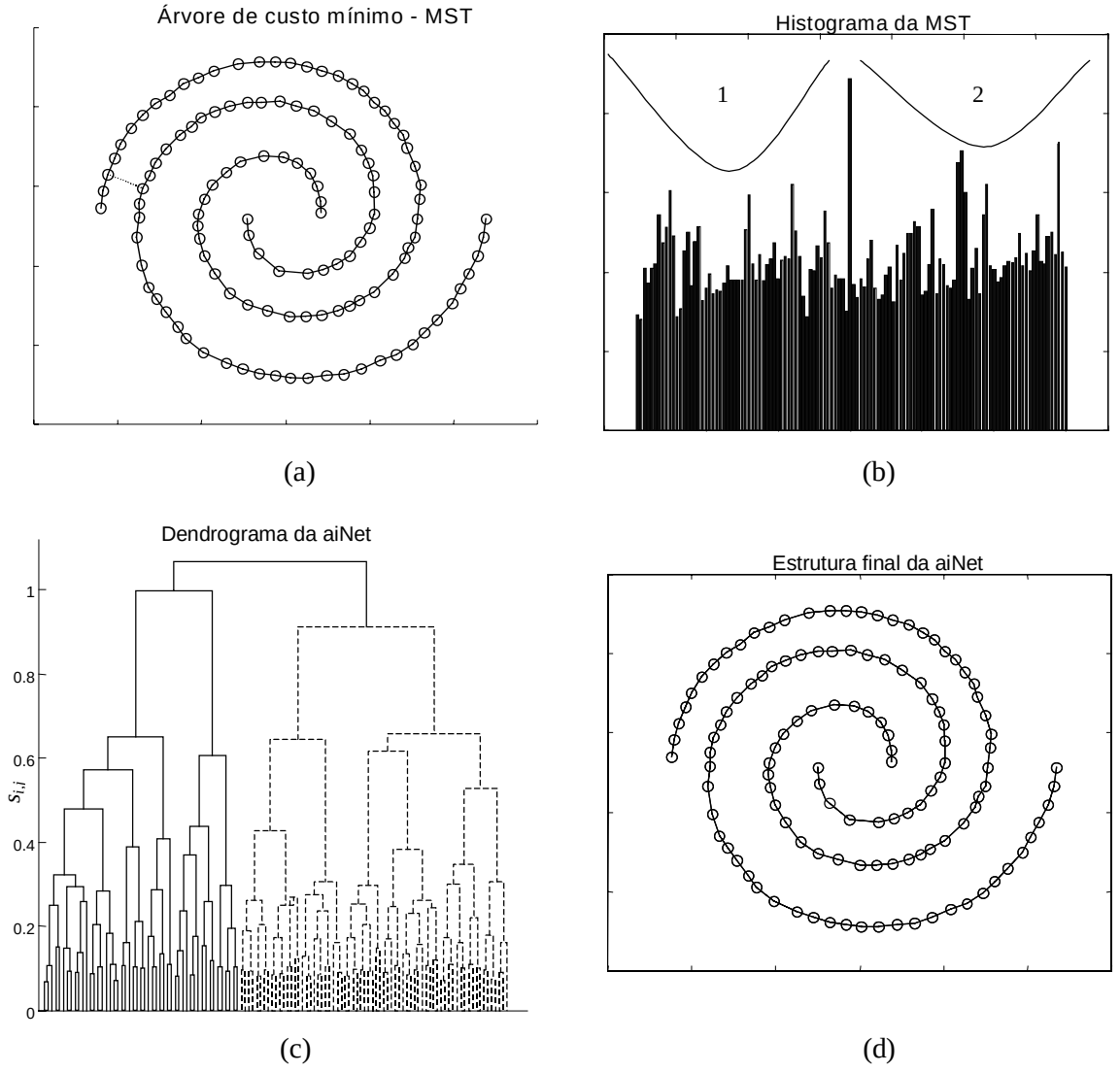
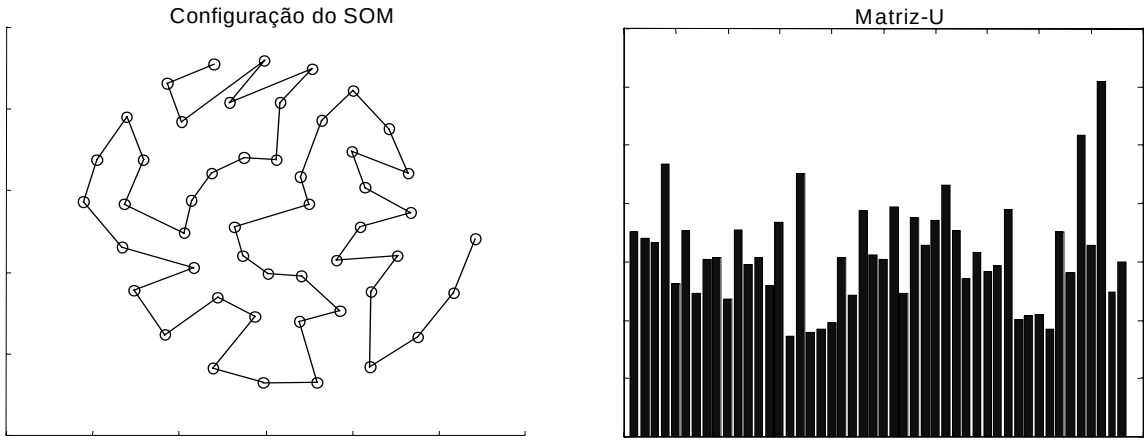


Figura 6.34. Resultados da aplicação da aiNet ao problema SPIR. (a) Árvore de custo mínimo (MST), onde a linha tracejada indica a conexão a ser removida. (b) Histograma da MST. (c) Dendrograma da rede indicando grandes diferenças entre os valores de fusão. (d) Arquitetura final da aiNet com dois clusters.



(a) (b)

Figura 6.35. Resultado da aplicação do SOM ao problema SPIR. (a) Configuração final dos vetores de pesos e ligação com seus vizinhos mais próximos. (b) Matriz-U.

A Figura 6.34(a) e (b) apresenta a árvore de custo mínimo e seu respectivo histograma. Partindo do histograma da MST é possível identificar a existência de dois clusters de anticorpos na aiNet empregando-se um fator $f = 2$. Neste caso, a rede resultante foi composta por $m = 121$ anticorpos, indicando uma taxa de compressão $CR = 36.32\%$. Note que a compressão foi maior em regiões do espaço onde a quantidade de antígenos similares é maior, ou seja, onde existe uma maior quantidade de informações redundantes. O dendrograma da aiNet também permite identificar dois grandes grupos (clusters) de anticorpos, como ilustrado na Figura 6.34(c).

A Figura 6.35 apresenta o resultado da aplicação do SOM com $m = 46$ ($m_0 = 50$) neurônios na camada de saída ao problema SPIR. A configuração final do SOM e sua respectiva matriz-U não nos permitem concluir nada a respeito da quantidade de clusters existente no conjunto amostral.

6.5.1.4. CHAINLINK

1000 dados amostrais em um espaço $\Re^3$ foram arranjados sob a forma de dois anéis entrelaçados, dos quais um deles segue na direção *x-y* e o outro na direção *x-z*. Os dois anéis podem ser interpretados como pertencentes a uma corrente, cada qual composta por 500 amostras. Os dados são produzidos por um gerador de números aleatórios dentro de dois toróides de raio R = 1.0 e r = 0.1, de acordo com a Figura 6.28(d).

A análise estatística dos dados mostra que eles possuem as seguintes propriedades: seus diferentes componentes (*x*,*y*,*z*) não estão correlacionados (utilizando o coeficiente de correlação de Pearson) e a distribuição de cada componente pode ser vista como normal. Uma análise de componentes principais mostra que não existe nenhuma dimensão mais representativa do que as demais. Além disso, não existe nenhum hiperplano que possa

separar os dados corretamente em dois subconjuntos, ou seja, eles são não-linearmente separáveis. Por outro lado, da forma pela qual os dados foram criados, é possível identificar claramente duas classes distintas, por simples inspeção visual. A distância entre os dados amostrais de cada classe é uma ordem de magnitude menor do que a distância entre as classes (Ultsch, 1995).

Os parâmetros de treinamento foram `[Ab`$_m$`,S]=aiNet(Ag,3,5,4,0.1,1,0.15,10)`. A Figura 6.36(a) e (b) apresenta a árvore de custo mínimo resultante e seu histograma correspondente. Partindo do histograma da MST é possível determinar de forma contundente os dois clusters da rede, utilizando-se um fator $f = 2$.

Note que, neste caso, a avaliação dos valores de fusão (tamanho do passo) do dendrograma da aiNet (Figura 6.36(c)) torna-se difícil, podendo levar a interpretações incorretas. A taxa de compressão foi de $CR = 94.5\%$ ($m = 55$).

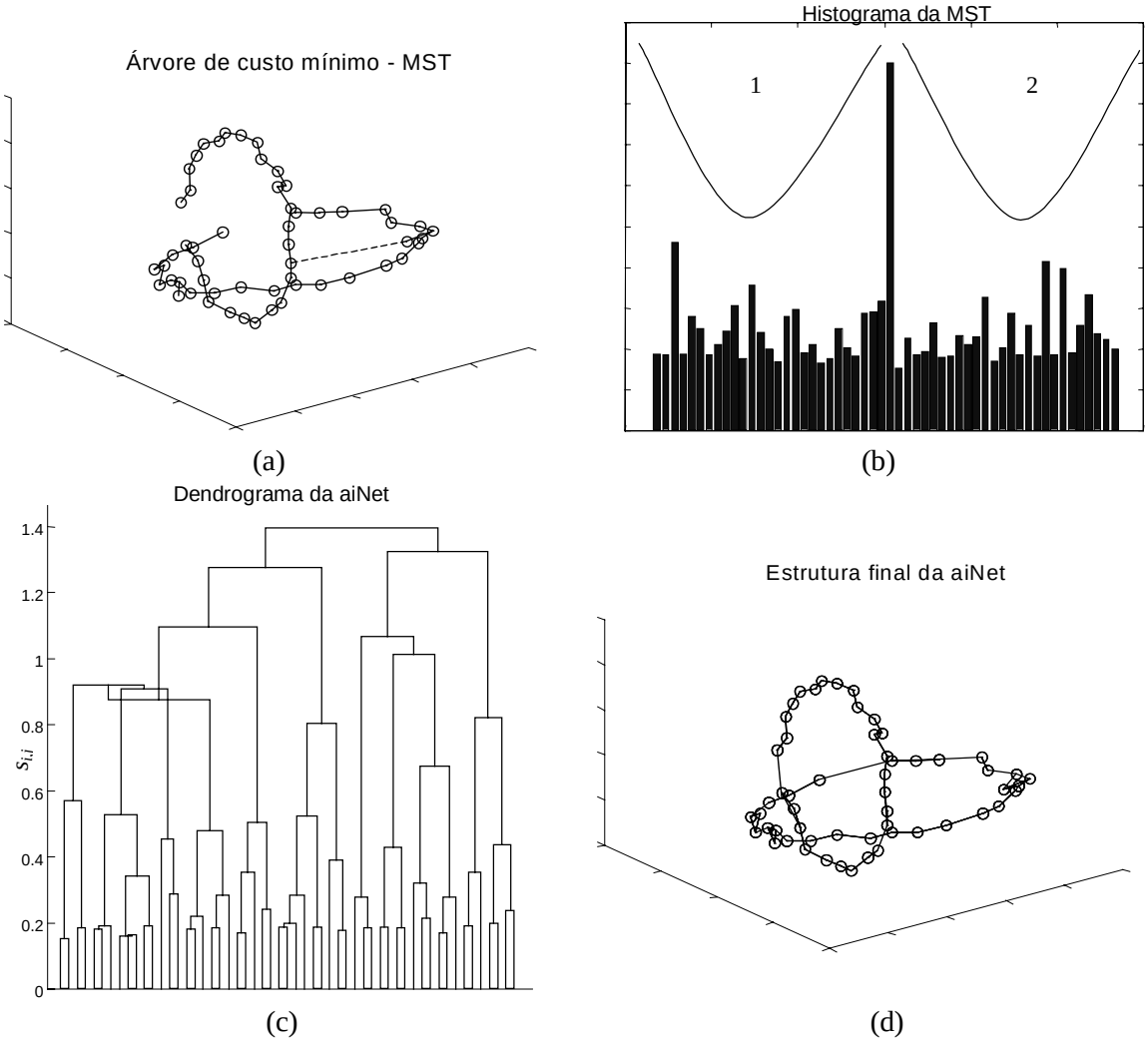
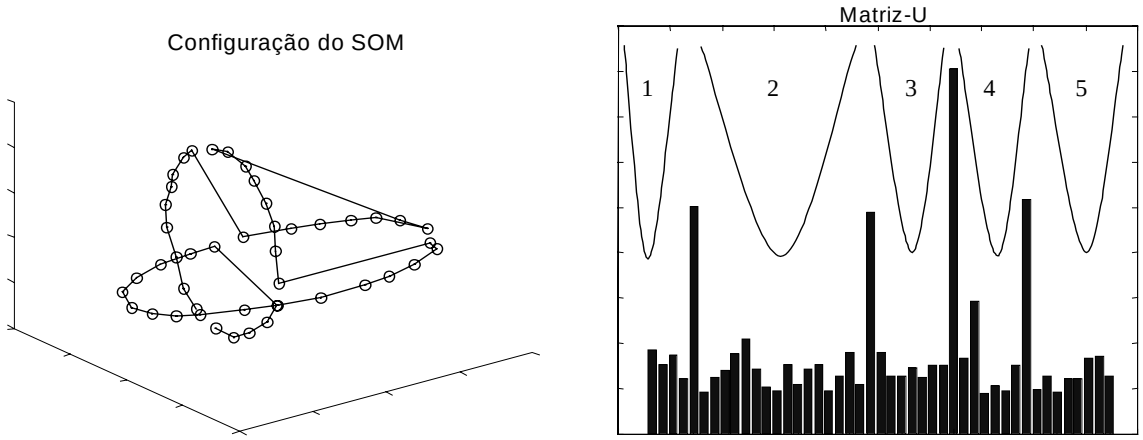


Figura 6.36. Aplicação da aiNet ao problema CHAINLINK. (a) Árvore de custo mínimo (MST), onde a linha tracejada indica a conexão a ser removida. (b) Histograma da MST. (c) Dendrograma da rede. (d) Arquitetura final da aiNet com dois clusters.



(a) (b)

Figura 6.37. Resultado da aplicação do SOM ao problema CHAINLINK. (a) Configuração final dos vetores de pesos e ligação com seus vizinhos mais próximos. (b) Matriz-U.

O SOM foi aplicado na solução deste problema com um grid de saída composto por $m = 46$ neurônios ($m_0 = 50$). Os resultados estão apresentados na Figura 6.37. Para este problema, a matriz-U foi composta por 5 vales, indicando 5 clusters distintos. Note que estes 5 clusters podem ser obtidos desenhando-se 5 hiperplanos separando as partes dos anéis nos neurônios correspondentes aos picos da matriz-U.

6.5.2. Projeto Automático de Redes Neurais de Funções de Base Radial

Como discutido na Seção 4.2.2.1, uma das principais limitações para o funcionamento apropriado de uma rede neural com funções de ativação de base radial, ou simplesmente RBF, está na determinação dos parâmetros das RBFs, inclusive a quantidade de neurônios da rede. Juntamente com um algoritmo para a determinação automática da dispersão das funções de base radial, o algoritmo de treinamento da aiNet foi empregado na determinação da quantidade e posicionamento da base destas funções, dando origem ao algoritmo denominado ICS (*immunity-based center selection*), cujos resultados foram apresentados em de Castro & Von Zuben (2001b,e) e serão reproduzidos nesta tese.

Cada anticorpo de memória resultante do treinamento da aiNet corresponde ao centro de uma RBF da rede neural, portanto, resta-nos apenas descrever o algoritmo empregado na determinação da dispersão destes centros definidos pela aiNet. Trata-se de uma abordagem não-supervisionada que pode ser descrita como a seguir.

Seja ρ_j a dispersão associada ao centro $\boldsymbol{c}_j$, $j = 1,...,m_1$, localizado pelo algoritmo de treinamento da aiNet. Neste caso, assumimos que a RBF é composta por funções Gaussianas descritas como na Equação (6.6):

$$g(\| \boldsymbol{x} - \boldsymbol{c}_j \|^2) = \exp\left(-\frac{1}{2} \frac{\| \boldsymbol{x} - \boldsymbol{c}_j \|^2}{\rho_j^2} \right), \quad j = 1,...,m_1 . \qquad \textbf{(6.6)}$$

onde $\boldsymbol{x}$ é o vetor de entrada (antígeno).

Para definir o processo iterativo de atualização dos centros determinados, foi empregado um algoritmo não-supervisionado semelhante ao descrito na Seção 4.2.2.1.2:

1. *Inicialização*: Inicialize as dispersões ρ_j, $j = 1,..., m_1$, com o valor das distâncias dos centros posicionados $\boldsymbol{c}_j$ em relação ao seu padrão de entrada de maior afinidade $\boldsymbol{x}_i$, $i = 1,...,M$.
2. *Amostragem*: Selecione uma amostra de treinamento $\boldsymbol{x}$ aleatoriamente.
3. *Afinidade*: Determine o índice k do centro com maior afinidade em relação ao vetor de entrada $\boldsymbol{x}$, de acordo com a medida de distância

$$d_k = \|\boldsymbol{x}(t) - \boldsymbol{c}_k(t)\|, \qquad k(\boldsymbol{x}) = \arg\min_k d_k,\ k = 1,...,m_1, \tag{6.7}$$

   onde $\boldsymbol{c}_k(t)$ é o centro da k-ésima RBF e d_k sua distância ao padrão de entrada $\boldsymbol{x}(t)$, ambos na iteração t.
4. *Atualização*: Ajuste os centros das RBFs através das seguintes expressões

$$\text{Se } d_k < \rho_k, \text{ então } \rho_k(t+1) = \rho_k(t) - \gamma(t)\,\|\mathbf{x}(t) - \mathbf{c}_k(t)\|,\ k = k(\mathbf{x}),$$
$$\text{Senão, se } d_k > \rho_k, \text{ então } \rho_k(t+1) = \rho_k(t) + \lambda(t)\,\|\mathbf{x}(t) - \mathbf{c}_k(t)\|,\ k = k(\mathbf{x}), \tag{6.8}$$

onde $\gamma(t)$ e $\lambda(t)$ são as taxas de aprendizagem para as dispersões das RBFs na iteração t. A taxa γ é inicializada com um valor próximo da unidade e decresce geometricamente ao longo das iterações até que um valor mínimo γ_{min} seja atingido.

5. *Ciclo*: repita os Passos 2 a 5 até que $\gamma(t) \leq \gamma_{min}$.

Nas nossas implementações, $\lambda(t) = \gamma_{min}/\gamma(t)$, $\gamma(0) = 0.9$, $\gamma_{min} = 10^{-3}$ e, a cada 5 iterações, o valor de γ é decrescido geometricamente por um fator 0.95.

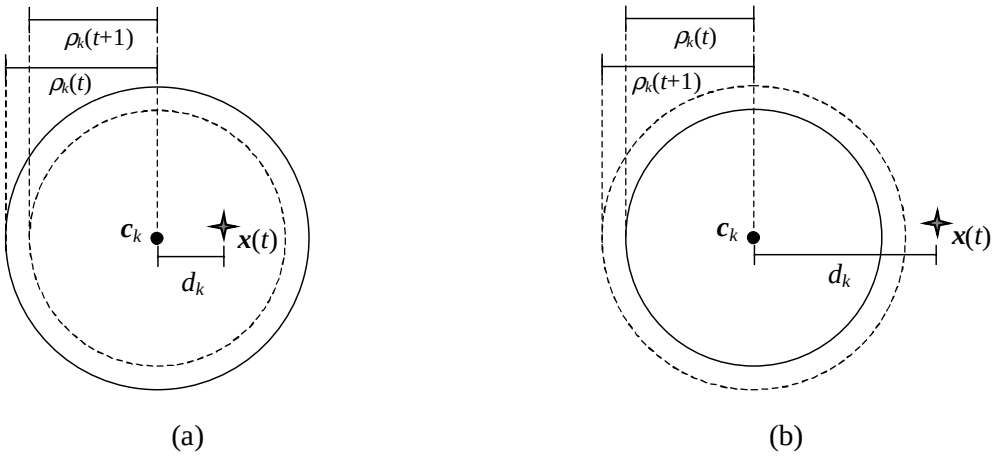


Figura 6.38. Determinação iterativa das dispersões das RBFs. (a) Redução da dispersão. (b) Aumento da dispersão.

Tabela 6.11. Parâmetros de treinamento da aiNet para a determinação de centros de RBFs.

	σ_s	σ_d	n	β	$\zeta(\%)$	*gen*
***sin*(*x*)**	0.50	1.0	4	10	20	5
CLASS	0.15				10	10
SPIR	0.80				10	5
IRIS	0.20				20	10

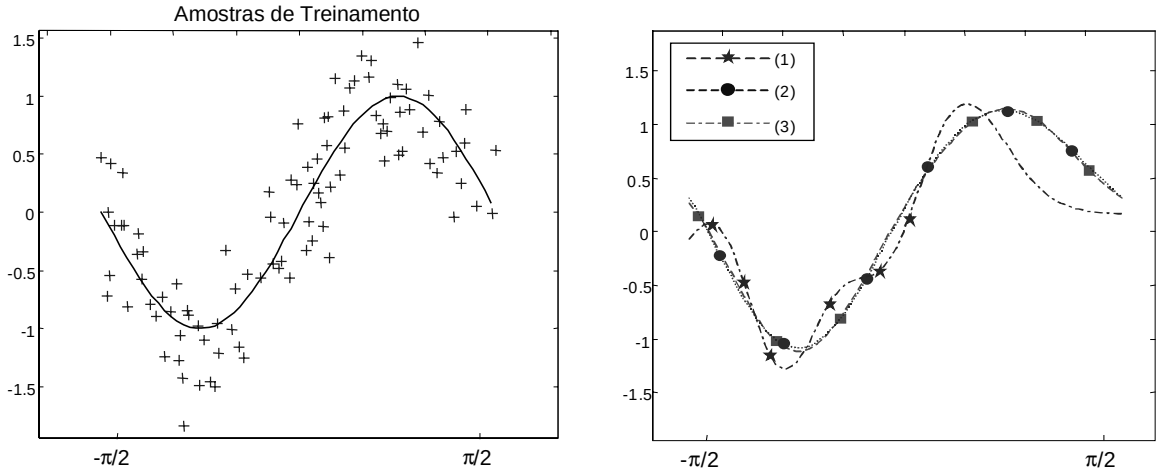
Esta regra de atualização para o vetor ρ de dispersões pode ser explicada da seguinte forma. Se a dispersão ρ_k do centro $\boldsymbol{c}_k$ de maior afinidade ao padrão $\boldsymbol{x}(t)$ é menor do que a distância d_k, Equação (6.8), então sua dispersão ρ_k pode ser reduzida, senão sua dispersão ρ_k deve ser aumentada para que o padrão de treinamento seja apropriadamente classificado. Este processo está ilustrado na Figura 6.38.

Com o objetivo de avaliar o desempenho do algoritmo, ele foi aplicado a diversos problemas de regressão e classificação: *sin*(*x*), CLASSNLS, SPIR e IRIS. Seus resultados foram comparados com o método de inicialização aleatória, apresentado na Seção 4.2.2.1.1, com o método *k*-means, apresentado na Seção 4.2.2.1.2 e com alguns resultados da literatura. Os parâmetros de entrada da aiNet estão resumidos na Tabela 6.11. Os resultados a serem apresentados são os valores médios, obtidos a partir de 10 simulações computacionais, e o desvio padrão ($\pm\delta$) em relação a esta média.

6.5.2.1. Função *sen*(*x*)

Trata-se de um problema de aproximação de funções, com $M = 100$ amostras de treinamento para um período da função *sen*(*x*). Estas amostras foram geradas com ruído uniformemente distribuído no intervalo [–0.7,0.7]. A função *sen*(*x*) e os dados utilizados para a avaliação do algoritmo ICS estão apresentados na Figura 6.39(a). A Figura 6.39(b) apresenta o desempenho (aproximação e posição dos centros) dos três algoritmos aplicados a este problema. A quantidade média de centros determinados pela aiNet foi $m_1 = 6.8$ ($\pm$ 0.8), correspondente a uma taxa média de compressão $CR = 97.84\%$. As aproximações apresentadas na Figura 6.39(b) foram obtidas para $m_1 = 6$ centros.

Se os centros determinados pelo método de inicialização aleatória estiverem aproximadamente uniformemente distribuídos no espaço de entradas, então esta estratégia pode resultar em uma boa aproximação, caso contrário, a aproximação pode ser polarizada em relação a alguns dados de entrada, como pode ser visto na Figura 6.39(b). Observando esta figura, é possível verificar que o método *k*-means foi capaz de realizar uma aproximação semelhante à do algoritmo ICS, porém com a necessidade de se definir a priori a quantidade m_1 de centros e assumir que todos as funções de base radial possuem a mesma dispersão ρ, dada pela Equação 4.13.



(a) (b)

Figura 6.39. Aproximação da função *sen*(*x*). (a) 100 amostras de treinamento (+) e função *sen*(*x*) definida no intervalo [–π,+π] (linha sólida). (b) Desempenho dos algoritmos: (1) seleção aleatória; (2) *k*-means e (3) ICS.

O algoritmo proposto permitiu determinar $\rho = [0.88,1.38,0.92,1.14,1.02,1.04]$ como sendo o vetor de dispersões das RBFs, enquanto a Equação 4.13 produziu $\rho = 1.63$ e $\rho = 1.52$ para a inicialização aleatória e via *k*-means, respectivamente. Orr (1995) obteve uma boa regularização para $m_1 = 3$ RBFs na rede, enquanto o algoritmo proposto por Chen *et al.* (1995) selecionou $m_1 = 15$ RBFs. No caso do método ICS, a quantidade de RBFs é controlada por σ_s, porém existe uma relação entre o valor escolhido para σ_s e a qualidade da solução. Por exemplo, com $\sigma_s = 1.0$, uma rede com $m_1 = 3$ pode ser obtida, porém com um baixo nível de aproximação. Resultados satisfatórios para o algoritmo proposto ocorrem para $m_1 \geq 5$.

6.5.2.2. CLASSNLS

Este problema foi descrito na Seção 6.5.1.2, Figura 6.28(b). A diferença é que, enquanto lá os rótulos dos dados não foram usados em nenhum momento, aqui eles não serão usados no posicionamento dos centros e na definição das dispersões mas serão usados no ajuste dos pesos de saída da rede neural. A Figura 6.40 apresenta a posição dos centros e as superfícies de decisão determinadas pelos três algoritmos aplicados ao problema CLASSNLS. A quantidade média de centros determinados pela aiNet foi $m_1 = 6.6$ ($\pm$ 1.4), correspondente a uma taxa média de compressão $CR = 96.67\%$. Foram utilizados $m_1 = 7$ RBFs para os métodos de inicialização aleatória e via *k*-means. Note que a inicialização aleatória pode resultar em uma quantidade de clusters menor do que o número real de classes existente no conjunto amostral, como pode ser observado na Figura 6.40(b).

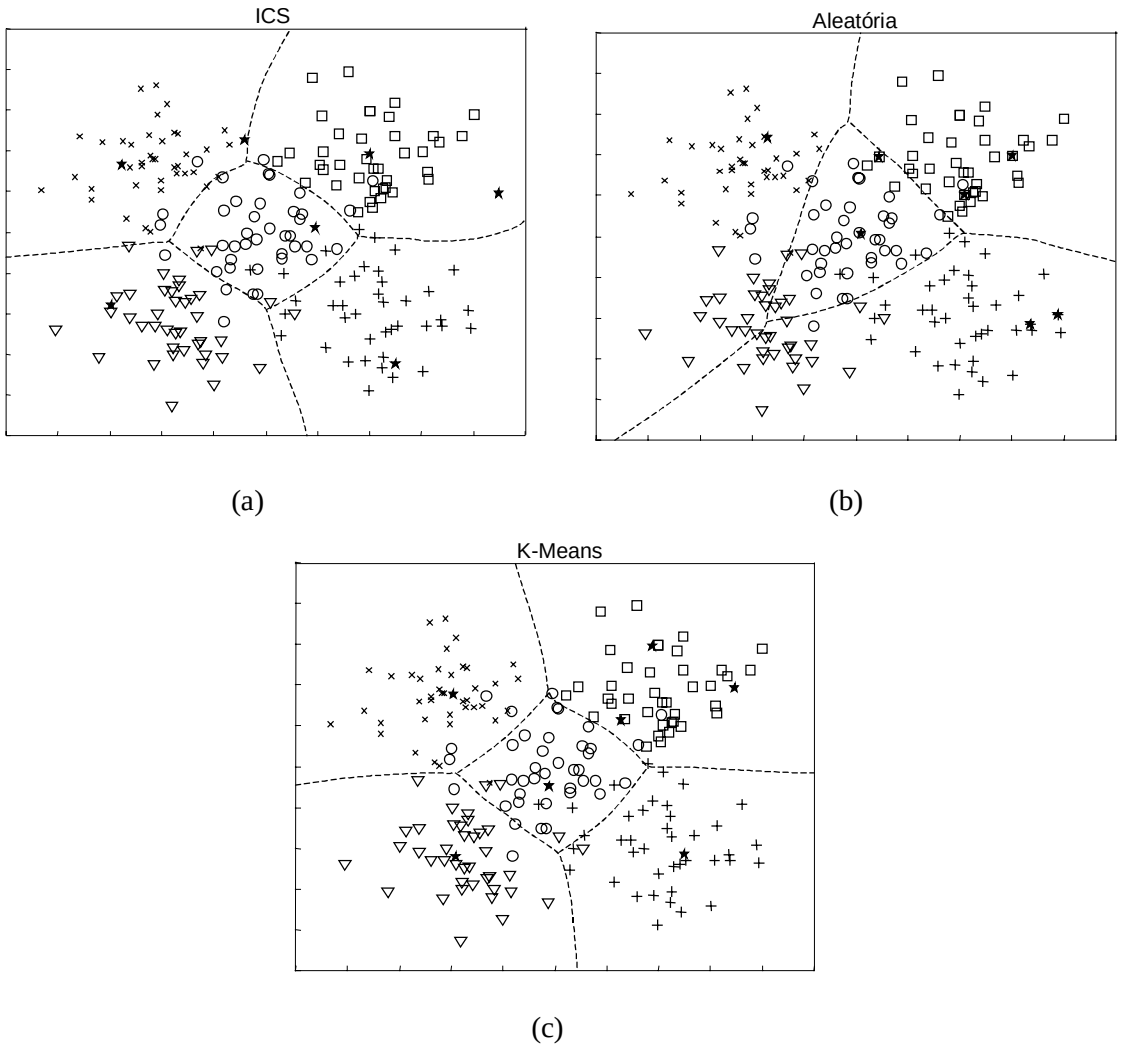


Figura 6.40. Posição dos centros (estrelas em preto) e superfícies de decisão para o problema CLASSNLS para $N = 7$. (a) ICS. (b) Inicialização aleatória. (c) *k*-means.

Este problema foi utilizado na literatura para avaliar a capacidade de classificação de algoritmos não-supervisionados (de Castro & Von Zuben, 1999b), onde um erro de classificação da ordem de 17% foi verificado, enquanto o método ICS apresenta um erro médio de 9.5%. Entretanto, é importante salientar que a RBF resultante opera de forma supervisionada, resultando, geralmente, em erros de classificação inferiores aos obtidos pelos métodos não-supervisionados.

6.5.2.3. SPIR

Este problema foi descrito na Seção 6.5.1.3, Figura 6.28(c). A diferença é que, enquanto lá os rótulos dos dados não foram usados em nenhum momento, aqui eles não serão usados no posicionamento dos centros e na definição das dispersões mas serão usados no ajuste dos pesos de saída da rede neural. A Figura 6.41 apresenta a posição dos centros e superfícies

de decisão determinadas pelos três algoritmos aplicados ao problema SPIR. A quantidade média de centros determinados pela aiNet foi $m_1 = 52.7$ ($\pm$ 3.1), correspondente a uma taxa média de compressão $CR = 72.26\%$. Para efeito de comparação, foram utilizados $m_1 = 52$ RBFs para os métodos de inicialização aleatória e via *k*-means. Novamente o método de inicialização aleatória pode resultar em uma classificação incorreta (Figura 6.41(b)), enquanto o algoritmo ICS sempre resulta em uma classificação correta para os parâmetros de treinamento adotados. Neste caso, foi possível verificar que o algoritmo *k*-means pode resultar em centros posicionados em regiões do espaço sem nenhum dado, como ilustrado na Figura 6.41(c).

O problema SPIR já foi utilizado na literatura para avaliar a capacidade de novos algoritmos de determinação automática de arquiteturas de redes do tipo RBF. Uma versão supervisionada da rede construtiva proposta por Fritzke (1994a) foi aplicada na solução do problema SPIR (Fritzke, 1994b). Nesta aplicação, o autor utilizou $M = 194$ amostras de treinamento o que resultou em uma rede RBF com $m_1 = 145$ unidades intermediárias, uma quantidade muito superior à média $m_1 = 52.7$ ($\pm$ 3.1) obtida pelo algoritmo ICS.

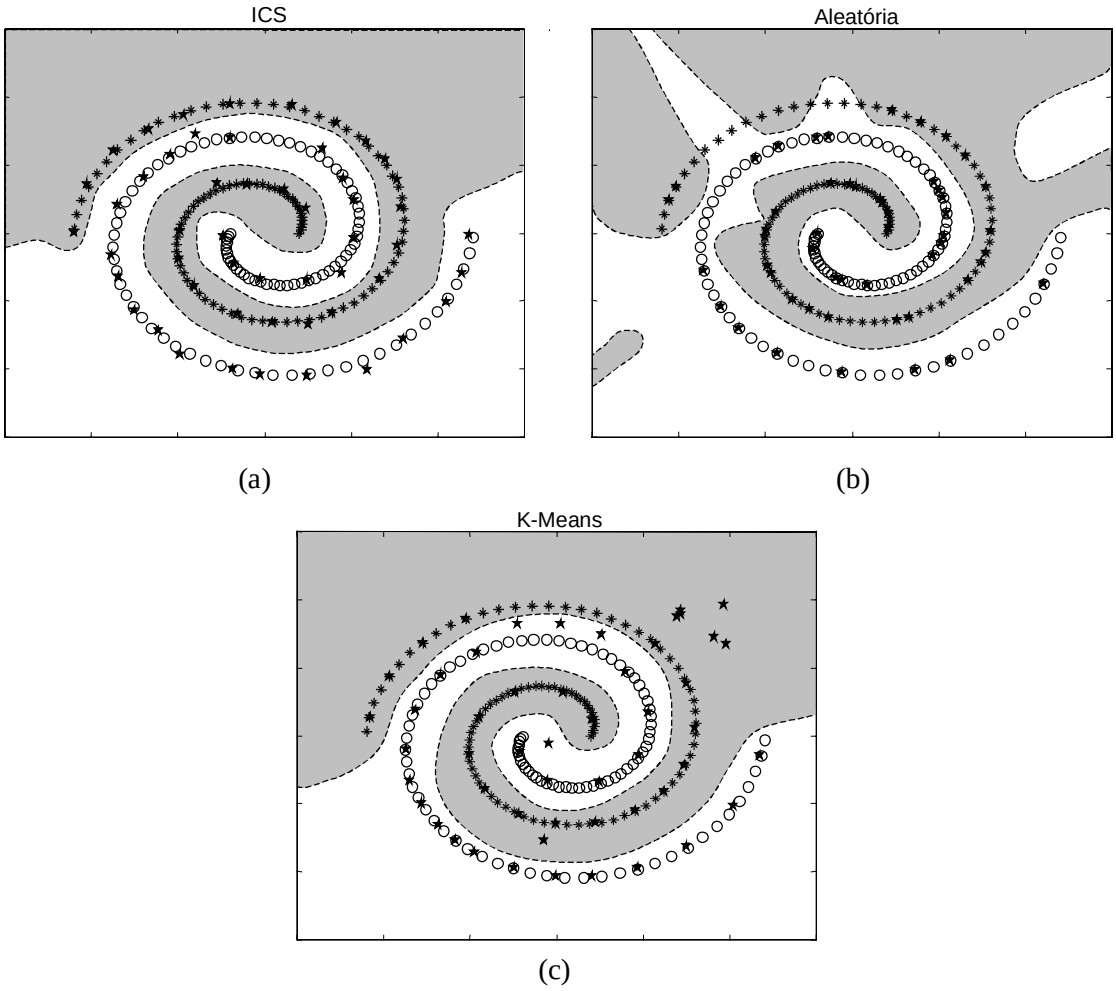


Figura 6.41. Posição dos centros e superfícies de decisão para o problema SPIR, $N = 52$. (a) ICS. (b) Inicialização aleatória. (c) *k*-means.

6.5.2.4. IRIS

Para testar a capacidade de generalização do algoritmo ICS, foi utilizado o problema IRIS (URL 3). Este conjunto de dados é composto por 150 amostras distribuídas em três classes distintas, onde apenas uma delas é linearmente separável das demais. Todo o conjunto amostral foi particionado para gerar conjuntos de treinamento e teste.

As simulações foram executadas considerando-se 10 partições distintas para cada tamanho do conjunto de treinamento, com 10% a 50% dos dados empregados exclusivamente para treinamento, e o restante para a avaliação da capacidade de generalização da rede. A Tabela 6.12 apresenta o erro médio de classificação obtido considerando cada uma das três estratégias: ICS, inicialização aleatória e via *k*-means. A quantidade média de RBFs determinada pelo algoritmo ICS também é apresentada. Os outros dois métodos requerem este parâmetro como dado de entrada.

Existem vários resultados na literatura referentes à aplicação de algoritmos de treinamento modificados ao problema da IRIS (Karayiannis & Mi, 1997; Jayasuriya & Halgamuge, 1998). A grande maioria destes algoritmos são construtivos, ou seja, a dimensão da rede RBF é determinada durante o processo de aprendizagem. A Tabela 6.13 resume os melhores desempenhos dos melhores resultados apresentados na literatura (maiores detalhes podem ser encontrados na literatura citada) contrastando-os com o melhor desempenho obtido pelo método ICS proposto.

Tabela 6.12. Classificação correta (valores médios e desvio padrão) dos algoritmos (ICS, Inicialização Aleatória e *k*-means) para o problema IRIS, onde Etr é o erro médio ($\pm\delta$) para o conjunto de treinamento e Ete é o erro médio ($\pm\delta$) para o conjunto de teste. Ntr é a quantidade de amostras de treinamento.

	ICS		**Aleatória**		***k*-means**		m_1
Ntr	Etr (%)	Ete (%)	Etr (%)	Ete (%)	Etr (%)	Ete (%)	
75	3.33 ± 2.61	4.67 ± 2.20	5.07 ± 1.97	4.27 ± 1.51	3.60 ± 2.81	5.87 ± 3.68	9.40 ± 1.90
90	3.00 ± 1.29	3.83 ± 2.61	6.11 ± 2.11	4.83 ± 4.12	4.22 ±3.26	5.17 ± 2.77	9.11 ± 1.62
105	3.71 ± 1.58	4.44 ± 4.06	4.76 ± 1.74	5.77 ± 5.05	4.38 ± 2.25	5.56 ± 3.19	8.70 ± 1.64
120	3.00 ± 0.98	3.33 ± 1.57	7.58 ± 9.72	9.33 ± 11.3	4.58 ± 1.58	4.66 ± 5.71	8.80 ± 1.48
135	3.04 ± 1.46	1.33 ± 2.81	4.66 ± 4.54	6.00 ± 12.8	3.48 ± 1.40	2.67 ± 3.44	11.4 ± 2.99

Tabela 6.13. Melhor desempenho do algoritmo ICS aplicado ao problema IRIS e comparado com os melhores resultados dos melhores métodos da literatura (Karayiannis & Mi, 1997 – Tabela II; Jayasuriya & Halgamuge, 1998 – Tabela V), para Ntr = 75.

Metódo	Etr (%)	Ete (%)	m_1
ICS	0.0	0.0	7
MSBFN	-----	2.9	5
ELEANNE 2	0.0	1.3	11
Gradient descent	1.3	1.3	7

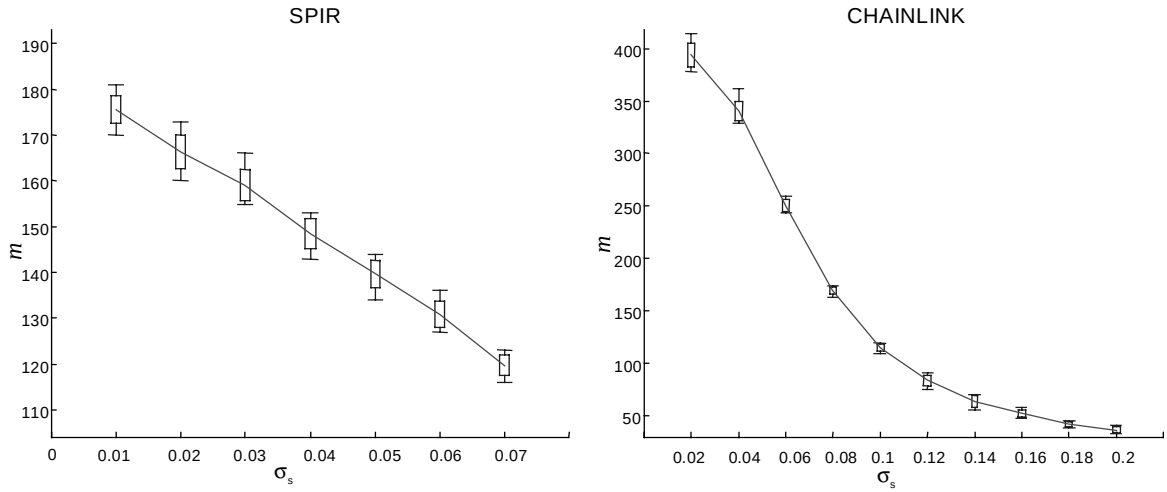
6.5.3. Análise de Sensibilidade

Do Passo 1.1.1 ao Passo 1.1.7 do algoritmo de treinamento da aiNet, descrito na Seção 5.5.2, o algoritmo CLONALG é reproduzido, portanto a análise de sensibilidade do CLONALG em relação aos parâmetros n, N_c e d também se aplica à aiNet. Resumidamente (ver Seção 6.2.3), foi verificado que n é diretamente proporcional ao valor médio da afinidade dos anticorpos em relação aos antígenos, enquanto N_c implica uma maior velocidade de convergência do algoritmo e d aumenta a capacidade de exploração do espaço de formas. Sendo assim, será feita uma análise de sensibilidade da aiNet em relação aos parâmetros da rede imunológica, como σ_s, σ_d e ζ. Além disso, a relação entre n e a quantidade final m de anticorpos de memória na aiNet também será verificada.

Na Seção 6.5.1 a aiNet foi aplicada a diversos problemas de teste objetivando ilustrar sua adequação e potencialidade. Para que a aiNet possa ser aplicada a qualquer problema, vários parâmetros devem ser definidos pelo usuário, destacando:

- σ_s: limiar de supressão;
- σ_d: limiar de mortalidade natural;
- n: quantidade de anticorpos a serem selecionados para clonagem; e
- ζ: percentual de anticorpos maduros a serem re-selecionados.

A Figura 6.42 apresenta a relação entre o limiar de supressão σ_s e a quantidade final m de anticorpos de memória na aiNet para os problemas SPIR e CHAINLINK. Como discutido na Seção 5.5.2, σ_s controla a quantidade de anticorpos na rede sendo responsável por sua plasticidade. Valores elevados para σ_s indicam anticorpos mais generalistas, enquanto valores pequenos para σ_s resultam em anticorpos altamente específicos. Este parâmetro é crítico, pois uma escolha inadequada pode resultar em classificações incorretas.



(a) (b)

Figura 6.42. Relação entre a quantidade final de anticorpos na aiNet (N) e o limiar de supressão (σ_s). Os resultados são os valores máximo, mínimo, médio e o desvio padrão de 10 simulações para os problemas SPIR e CHAINLINK.

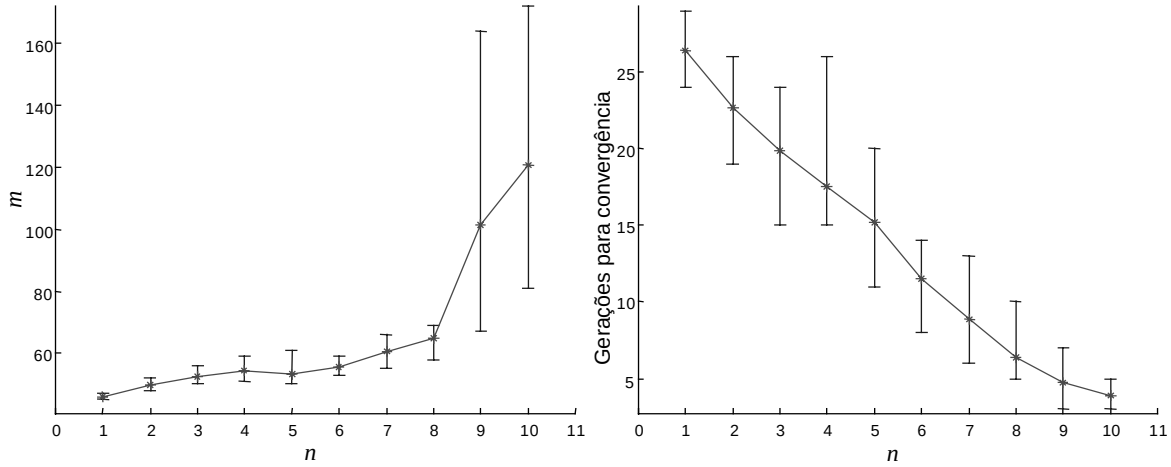
Tabela 6.14. Quantidade de anticorpos N_{Ab} que sofreram mortalidade natural (poda) na primeira geração do algoritmo de treinamento da aiNet para o problema CLASSLS em 10 simulações.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Estatística**
N_{Ab}	74	87	73	60	81	54	55	58	62	77	68.1±11.7

Se algum conhecimento sobre a distribuição dos dados (ou classes) do problema é conhecido, ele pode ser empregado na determinação de σ_s. Para os problemas testados, $\sigma_s = 0.2$, $\sigma_s = 0.2$, $\sigma_s = 0.08$ e $\sigma_s = 0.2$, respectivamente, resultaram em bons desempenhos do algoritmo. Valores acima destes causaram classificações incorretas em algumas simulações.

O limiar de morte natural (σ_d) é responsável pela eliminação de anticorpos cuja afinidade antigênica é baixa. Sem perda de generalidade, considerando o primeiro problema, CLASSLS (Seção 6.5.1.1), é possível avaliar a relevância deste parâmetro para a aprendizagem da rede. A Tabela 6.14 apresenta a quantidade de anticorpos que sofreram mortalidade natural durante a primeira geração do algoritmo de treinamento. Os resultados são apresentados para 10 simulações distintas. Em todos os casos, a mortalidade natural ocorreu apenas durante a primeira geração. Nas gerações subsequentes o parâmetro σ_d não foi responsável pela morte de nenhum anticorpo. Este resultado permite-nos concluir que os processos de maturação de afinidade e forte pressão seletiva fazem com que o algoritmo seja capaz de absorver rapidamente (em apenas uma geração) as características mais relevantes do conjunto antigênico.

Para estudar a sensibilidade da aiNet em relação aos parâmetros n e ζ, foi escolhido um valor para o limiar de supressão tal que a rede resultante apresentasse aproximadamente $m = 50$ anticorpos para o problema CLASSLS. O objetivo é verificar a capacidade da aiNet em reproduzir (aprender) o conjunto antigênico através da definição de um critério de parada que avalie a distância média (ou erro médio) entre os anticorpos da rede e os antígenos, cujo valor foi escolhido igual a 10^{-2}. Sendo assim, adotou-se $\sigma_s = 0.01$.
A Figura 6.43(a) apresenta a relação entre n e m e a Figura 6.43(b) ilustra a relação entre n e a quantidade de gerações necessária para convergência (N_{gen}). É possível verificar que quanto maior n, maior a dimensão final da aiNet (maior m), indicando que n tem uma influência direta na plasticidade da rede. Por outro lado, da Figura 6.43(b), é possível verificar que quanto maior n, menor a quantidade de gerações necessária para convergência (aprendizagem do conjunto antigênico). Os resultados apresentados são os valores máximo, mínimo e médio obtidos a partir de 10 simulações.



(a) (b)

Figura 6.43. Sensibilidade da aiNet em relação à quantidade *n* de anticorpos a serem selecionados para clonagem. Valores máximo, mínimo e média tomadas de 10 simulações. (a) Relação $n \times m$. (b) Relação $n \times N_{gen}$.

Para estudar a sensibilidade da aiNet em relação ao parâmetro ζ, foi utilizado o mesmo critério de parada e limiar de supressão anteriores, variando-se ζ de 2% a 24% em intervalos de 2%. A Figura 6.44 apresenta a relação entre ζ, *m* e a quantidade de gerações para convergência (média de 10 simulações). Desta figura, é possível perceber que ζ não tem grande influência na dimensão final da rede, mas valores altos de ζ implicam em convergência mais lenta.

Intuitivamente poderíamos imaginar que quanto maior o valor de ζ mais rápida deveria ser a convergência da aiNet. Entretanto, diversos fatores devem ser considerados. Primeiramente, o critério de parada empregado é a diferença (distância) entre os anticorpos da rede e os antígenos a serem reconhecidos. Neste caso, dado um limiar de supressão $\sigma_s = 0.01$, a quantidade *m* de anticorpos de memória gerados é aproximadamente igual à quantidade de antígenos a serem reconhecidos ($m \approx M$). Portanto, existe aproximadamente um anticorpo de memória servindo como imagem interna para cada antígeno. Nesta situação, um valor de $\zeta > 2\%$ implica que, para cada antígeno, mais de um clone maduro está sendo substituído do conjunto de memória, portanto, enquanto o antígeno apresentado na iteração atual está sendo melhor aprendido, um elemento de memória com alta afinidade a algum outro antígeno está sendo "perdido", reduzindo a velocidade de convergência do algoritmo para o critério de parada adotado.

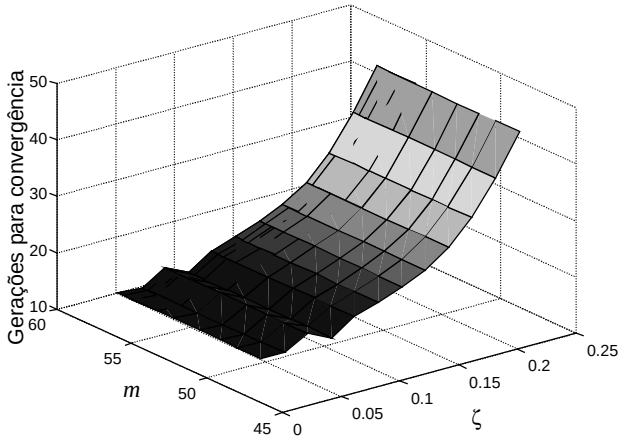


Figura 6.44. Relação entre ζ, m e a quantidade de gerações para convergência (média de 10 simulações).

6.5.4. Discussão

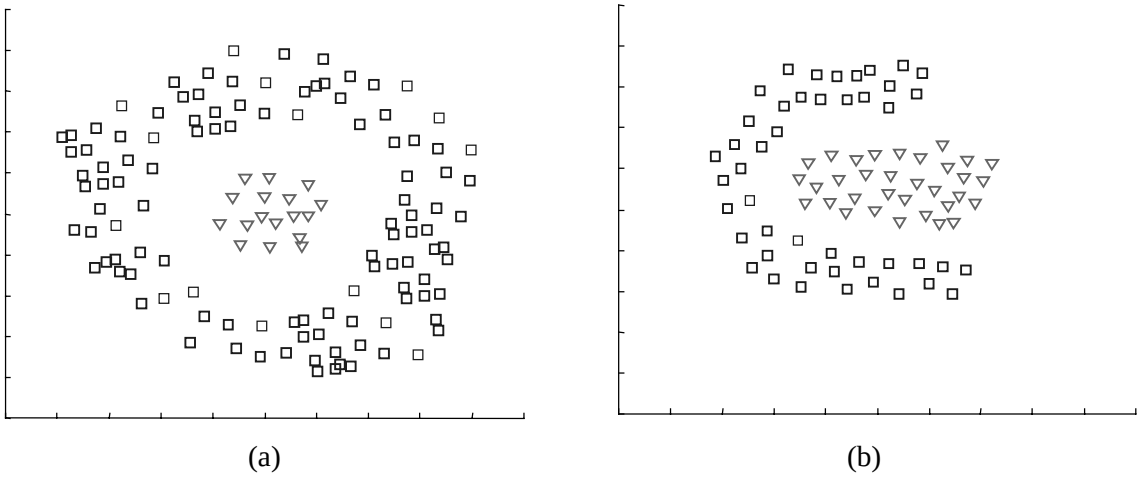
Os objetivos principais da aiNet foram definidos como sendo primeiramente a capacidade automática de extrair informações de um conjunto de dados (antígenos) através da construção de imagens internas, e reduzir a redundância do conjunto de anticorpos da rede dada a interação dos elementos do modelo de rede imunológica. A utilização de anticorpos de mesma dimensão que os antígenos permite manter a topologia original das classes, que é geralmente perdida quando alguma técnica de redução de dimensionalidade (como o SOM) é empregada. A aiNet permite a manutenção de relações de vizinhança em um espaço L-dimensional.

Os principais objetivos do SOM são reduzir a dimensionalidade dos dados preservando uma métrica e as relações de vizinhança entre os padrões de entrada (Kohonen, 1982). Com uma abordagem diferente, a aiNet reduz a redundância dos dados de treinamento permitindo a reconstrução métrica e das relações de vizinhança sem reduzir a dimensionalidade dos dados. Devido à reprodução das relações topológicas, informações similares (baseadas em uma métrica de distância) são mapeadas em anticorpos vizinhos (próximos), eventualmente o mesmo anticorpo, caracterizando uma quantização e clusterização do espaço de entrada.

A definição do limiar de supressão (σ_s) é crucial para a determinação da estrutura final da rede e, consequentemente, da quantidade e formas dos clusters definidos pela árvore de custo mínimo. A quantidade n de anticorpos com alta afinidade a serem selecionados para reprodução também demonstrou ser importante para o tamanho final da aiNet. Entretanto, este parâmetro, assim como vários outros, foi mantido fixo para a grande maioria das aplicações apresentadas. No caso dos problemas com uma quantidade maior de dados de treinamento, empregou-se $\zeta = 10\%$ para reduzir o tempo de processamento do algoritmo. A Tabela 6.11 apresenta valores default para alguns parâmetros de treinamento da aiNet como

σ_d, n, β e ζ. É importante mencionar também que em nenhum caso a aiNet demonstrou ser sensível à condição inicial, ou seja, ao repertório inicial de anticorpos, assumido de tamanho $N_0 = 10$ em todos os casos.

No caso da rede neural do tipo SOM, a definição a priori do tamanho do grid de saída impõe uma arquitetura de rede que nem sempre é capaz de mapear todas as interelações importantes, presentes nos dados de entrada, nos nós de saída. Mesmo para o caso dos modelos com arquitetura dinâmica (Fritzke, 1994a; de Castro & Von Zuben, 1999b; Cho, 1997), se os clusters dos dados de treinamento não são representáveis em espaços de menor dimensão ou então apresentarem uma conformação não-radial, estes algoritmos apresentam dificuldades de clusterização. No caso da aiNet, desde que exista uma distância relativa entre as classes, uma classificação correta pode ser obtida em qualquer situação. Exemplos claros desta afirmação foram apresentados nas Seções 6.5.1.3 e 0. Outros exemplos de problemas que poderiam ser resolvidos pela aiNet, mas para os quais o SOM e suas variantes provavelmente falhariam estão apresentados na Figura 6.45.



(a) (b)

Figura 6.45. Exemplos de problemas que poderiam ser resolvidos pela aiNet, mas para os quais uma rede neural artificial competitiva como o SOM falharia na determinação correta dos clusters.

6.5.4.1. Projeto Automático de RBFs

Os resultados apresentados na Seção 6.5.2 mostram a dependência do algoritmo *k*-means em relação à escolha inicial dos centros. Pode ser observado na Figura 6.41(c) que a definição a priori da quantidade de RBFs a ser utilizada na rede neural pode resultar no desperdício de recursos computacionais, com centros posicionados em locais do espaço com baixa densidade de dados. Este tipo de comportamento não ocorre para o algoritmo proposto, pois ele elimina os centros que não estão mapeando nenhum dado de treinamento. Assim como de Castro & Von Zuben (1999b) propuseram um método de poda para redes do tipo SOM, uma metodologia similar poderia ser empregada para o *k*-means no sentido de eliminar unidades pouco representativas do conjunto amostral.

O algoritmo ICS gera um conjunto de funções de base radial que satisfazem o teorema de Micchelli (1986). Este algoritmo permite naturalmente a produção de uma solução regularizada para problemas de aproximação de funções, como pôde ser visto na Figura 6.39(b). Também pôde ser verificado nesta figura, que o algoritmo ICS não necessariamente determina os centros das RBFs uniformemente distribuídos, como geralmente é feito pelo *k*-means. O ICS distribui as RBFs de acordo com a função densidade de probabilidade dos dados de treinamento e ajusta a dispersão de cada função de forma que as amostras sejam apropriadamente classificadas ou aproximadas.

O problema IRIS foi empregado para avaliar a capacidade de generalização do método ICS e seu desempenho foi comparado satisfatoriamente com os melhores resultados encontrados na literatura.