CAPÍTULO 5

FERRAMENTAS DE ENGENHARIA IMUNOLÓGICA

Este capítulo concretiza a engenharia imunológica através da proposição de quatro ferramentas computacionais desenvolvidas baseadas nos princípios imunológicos biológicos, introduzidos no Capítulo 2, e no paradigma de engenharia imunológica, proposto no Capítulo 3. Dentre os princípios fundamentais do sistema imunológico biológico utilizados, destacam-se a questão da diversidade populacional, o princípio da seleção clonal e a teoria da rede imunológica. A formalização teórica e o desenvolvimento dos algoritmos SAND, CLONALG, ABNET e aiNet são apresentados. Algumas das ferramentas propostas constituem estratégias híbridas que utilizam, além de conceitos imunológicos, noções de outras técnicas computacionais como as redes neurais artificiais competitivas e os algoritmos evolutivos revisados e discutidos no Capítulo 4.

"O acaso favorece a mente preparada" – Autor desconhecido

5.1. Introdução

Neste capítulo são propostas quatro ferramentas de engenharia imunológica:

1. SAND: algoritmo baseado na técnica clássica de *Simulated Annealing* (Kirkpatrick *et al.*, 1987) cujo objetivo principal é aumentar a diversidade de um repertório de anticorpos representado sob a forma de cadeias binárias (espaço de formas de Hamming) e vetores reais (espaço de formas Euclidiano). É utilizado o processo de mutação somática proporcional à afinidade;
2. CLONALG: implementação computacional do algoritmo de seleção clonal, cujas aplicações incluem aprendizagem de máquina e reconhecimento de padrões, otimização multimodal e combinatorial. Engloba o princípio de seleção clonal e maturação de afinidade;
3. ABNET: rede neural artificial Booleana com processos de crescimento (inserção de unidades), poda (remoção de unidades) e atualização de pesos, baseados nos princípios imunológicos da seleção clonal e maturação de afinidade;
4. aiNet: modelo de rede imunológica artificial com aplicações a problemas de análise de dados, como compressão de informações e clusterização. Utiliza o algoritmo CLONALG como parte do seu processamento e a teoria da rede imunológica para controlar a dimensão da rede.

A apresentação de todos os algoritmos propostos inclui a descrição e motivação para o seu desenvolvimento, o escopo de aplicações, a listagem de pseudocódigos, um estudo sobre o custo computacional envolvido no processamento de cada iteração e uma discussão caracterizando e relacionando as ferramentas propostas com outras abordagens da literatura. O estudo e a apresentação teórica dos algoritmos fazem parte deste capítulo, enquanto as aplicações e comparações de desempenho com outras abordagens serão apresentadas no Capítulo 6.

5.1.1. Características Gerais

Todos os quatro algoritmos propostos nesta tese e que serão descritos neste capítulo foram desenvolvidos baseados em algum princípio imunológico e serão analisados utilizando a estrutura formal para a engenharia imunológica proposta no Capítulo 3.

Como discutido na proposta de engenharia imunológica (Cap. 3), devido à característica monoclonal dos anticorpos, não é feita distinção entre uma célula B e seu receptor e, por convenção, todos os N indivíduos da população são denominados genericamente de anticorpos e representados por $\boldsymbol{Ab}_i$, $i = 1,...,N$. Considerando o problema genérico de reconhecimento de padrões, existe uma população **Ag** de antígenos a ser reconhecida por um repertório **Ab** de anticorpos. Cada anticorpo e cada antígeno é representado por um vetor (cadeia de atributos) $\boldsymbol{Ab}_i = \langle Ab_{i,1},..., Ab_{i,l_1}, Ab_{i,l_1+1},..., Ab_{i,L} \rangle$ e $\boldsymbol{Ag}_j = \langle Ag_{j,1}, Ag_{j,2}, ..., Ag_{j,L} \rangle$, $j = 1,...M$, respectivamente, considerados de mesmo comprimento L. Não é feita distinção entre o paratopo e o idiotopo dos anticorpos. Portanto, a cadeia de atributos que representa esta molécula pode ser resumida na forma $\boldsymbol{Ab}_i = \langle Ab_{i,1}, Ab_{i,2}, ..., Ab_{i,L} \rangle$ e a representação pictórica dos anticorpos componentes da EI, ilustrada na Figura 3.9(a), pode ser sintetizada como na Figura 5.1(a), que constitui um caso particular da Figura 3.9(a). Esta representação pictórica é mais interessante para a visualização da aiNet, a ser apresentada na Seção 5.5.

Existem diversas características em comum entre as quatro ferramentas de engenharia imunológica a serem propostas, destacando-se:

- São baseadas em um repertório, ou população, de anticorpos;
- Foram genericamente desenvolvidas para resolver problemas de aprendizagem de máquina e reconhecimento de padrões, mas muitas serão aplicadas a outros contextos, como otimização e classificação;
- Não fazem diferença entre o paratopo e o idiotopo dos anticorpos, ou seja, a mesma cadeia de atributos que faz o reconhecimento antigênico é também responsável pelo reconhecimento dos elementos próprios (este aspecto é relevante apenas no modelo de rede imunológica artificial proposto, aiNet);
- Todas as células possuem o mesmo limiar de afinidade ε (quando ele é considerado); e
- Células ou anticorpos individuais podem sofrer variações genéticas, mais especificamente mutação (maturação de afinidade), permitindo que eles se adaptem melhor ao ambiente.

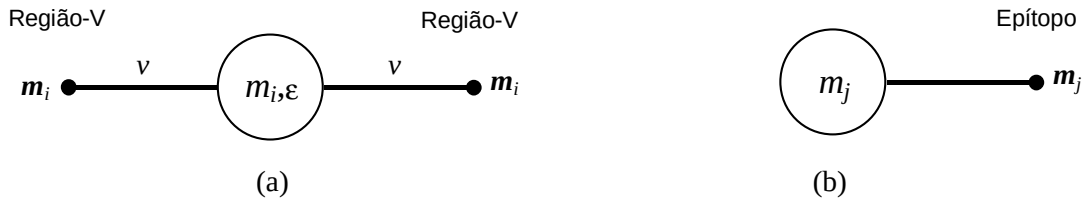


Figura 5.1. Representações pictóricas das moléculas utilizadas nos algoritmos propostos. (a) Anticorpo. A região-V é a região de ligação ao antígeno. (b) Antígeno.

Outro aspecto importante de todos os modelos propostos, é a utilização de uma taxa de mutação proporcional à afinidade, ou fitness, de cada anticorpo. O que o processo de maturação de afinidade, descrito na Seção 2.8.2, estabelece, é que quanto maior a afinidade antigênica de um receptor celular, menor a taxa de mutação a que ele estará sujeito durante o processo de reprodução e vice-versa. Esta característica pode ser simulada como proposto na Seção 3.4.3.4, Figura 3.16.

5.1.2. Complexidade Computacional

Um programa computacional, embora derivado de um algoritmo correto, pode ser inútil para algumas aplicações (dados de entrada), devido ao tempo de processamento ou à quantidade de memória necessária para armazenar os dados e variáveis envolvidas. A *análise de um algoritmo* refere-se ao processo de derivação de estimativas de tempo e espaço (memória) requeridos para executá-lo. A *complexidade de um algoritmo* refere-se à quantidade de tempo e espaço necessários para executá-lo (Johnsonbaugh, 1997). A determinação de parâmetros de desempenho de um programa computacional não é uma tarefa fácil e depende de uma série de fatores, tais como o computador sendo utilizado, a forma de representação dos dados, e como e com qual linguagem de programação o código foi implementado. Embora estimativas precisas do tempo de processamento de um algoritmo devam considerar estes aspectos, informações úteis podem ser obtidas pela análise da complexidade computacional do algoritmo.

O tempo necessário para executar um algoritmo é função dos dados de entrada. Ao invés de tratar os dados de entrada diretamente, utiliza-se parâmetros que caracterizam sua dimensão como, por exemplo, a quantidade N de dados. Neste capítulo, todos os algoritmos a serem propostos acompanharão uma análise assintótica de complexidade envolvendo o tempo de processamento e a quantidade de memória necessários para executá-los.

Sendo f e g funções reais com domínio no conjunto dos inteiros positivos, a notação usual para análise de custo computacional é (Horowitz & Sahni, 1978):

- $f(n) = O(g(n))$ se existem constantes c, $n_0 > 0$ tais que $|f(n)| \leq c\,|g(n)|$, $\forall\ n > n_0$. Diz-se que $f(n)$ é de *ordem no máximo* $g(n)$;
- $f(n) = \Omega(g(n))$ se existem constantes c, $n_0 > 0$ tais que $|g(n)| \leq c\,|f(n)|$, $\forall\ n > n_0$. Diz-se que $f(n)$ é de *ordem pelo menos* $g(n)$;
- $f(n) = \Theta(g(n))$ se $f(n) = O(g(n))$ e $f(n) = \Omega(g(n))$. Diz-se que $f(n)$ é de *ordem* $g(n)$;
- $f(n) = o(g(n))$ se $f(n) / g(n) \rightarrow 0$ quando $n \rightarrow \infty$.

Com exceção do primeiro algoritmo a ser apresentado, denominado SAND, todas as outras três ferramentas de engenharia imunológica a serem propostas (CLONALG, ABNET e aiNet) possuem duas etapas em comum: um mecanismo de seleção do(s) melhor(es) indivíduo(s) (competição), e um mecanismo de variação genética (mutação) dos indivíduos da população. Dada uma população **Ab** composta por N anticorpos, n membros desta população são selecionados, produzindo uma matriz $\mathbf{Ab}_{\{n\}}$, e reproduzidos gerando um conjunto **C** de clones. O conjunto **C** de clones irá sofrer um processo de mutação com taxas elevadas, gerando um conjunto **C***. Para simplificar o estudo de cada algoritmo individualmente, apresentaremos uma análise genérica dos operadores de seleção e mutação empregados.

5.1.2.1. Complexidade Computacional dos Operadores de Seleção

Suponha que N_c clones são gerados a partir da seleção dos n melhores indivíduos de **Ab** ($\mathbf{Ab} \in S^{N \times L}$), e que seus valores de afinidade, ou fitness, estão armazenados em um vetor $\boldsymbol{f}$. A forma mais comum de selecionar os n melhores indivíduos é ordenando o vetor de afinidades de forma que estes n melhores indivíduos possam ser extraídos facilmente. Este processo de ordenação pode ser executado em um tempo $O(N_c.\log N_c)$ utilizando-se por exemplo o algoritmo *heapsort* (Sedgewick, 1988). Note que não é necessário ordenar todo o vetor, é possível criar uma pilha com custo $O(N_c)$, selecionar o melhor elemento e aplicar um mecanismo de reparação da pilha ao restante dos elementos. Como este passo deve ser executado n vezes e o processo de reparação da pilha requer um tempo $O(\log N_c)$, toda a rotina demanda um tempo limitado a $O(N_c + n.\log N_c)$. Outros algoritmos de ordenação (*sort*) podem ser empregados, onde Fischetti & Martello (1988) apresentam um método de *quicksort* sofisticado capaz de extrair os n melhores elementos em um tempo $O(N_c)$. A quantidade de memória requerida por qualquer um destes algoritmos é $O(N_c)$.

Se desejarmos obter apenas o melhor indivíduo de uma população de tamanho N_c, ao invés dos n melhores, o custo computacional do algoritmo fica reduzido a $O(\log N_c)$.

5.1.2.2. Complexidade Computacional dos Operadores de Mutação

Primeiramente, é suposto que o operador de mutação é aplicado a uma cadeia de atributos de comprimento L (L-upla) e que esta operação é executada localmente para cada cadeia.

Seja $p_i \in (0,1]$ a probabilidade de que um componente $i = 1,...,L$ sofra uma mutação elementar. A mutação ocorre da seguinte forma: para cada componente i, sorteie um número aleatório uniformemente distribuído, $u \in (0,1)$, e mute o componente i caso $u \leq p_i$, do contrário mantenha-o inalterado. Esta etapa requer um tempo $\Theta(L)$. Entretanto, considerar que mutações elementares requerem um tempo constante nem sempre é apropriado. Por exemplo, seja $\boldsymbol{x} \in \mathbf{X} = \Re^L$ e seja $\boldsymbol{z} \sim N(0,\mathbf{C})$ um vetor aleatório com distribuição normal de média zero e matriz de covariância **C** simétrica e definida positiva. A menos que **C** seja diagonal, os componentes do vetor aleatório $\boldsymbol{z}$ estão correlacionados. Como são necessários $O(L)$ números aleatórios com distribuição normal e $O(L^2)$ operações elementares para construir o vetor aleatório $\boldsymbol{z}$, a operação de mutação completa $\boldsymbol{x} + \boldsymbol{z}$ requer um tempo $O(L^2)$. Consequentemente, uma operação elementar de mutação requer um tempo $O(L)$ (Bäck *et al.*, 2000b). A execução da operação de mutação sobre todos os N indivíduos

da população requer um tempo $O(N.L)$. De forma similar, a quantidade de memória necessária para realizar a mutação de uma população de N indivíduos de comprimento L é $O(N.L)$.

É importante mencionar que em alguns dos algoritmos a serem propostos (como ABNET e aiNet), o processo de mutação é determinístico, ou seja, a mutação é direcionada de forma a aumentar a afinidade Ag-Ab. Esse procedimento elimina o custo computacional necessário para sortear um número aleatório $u \in (0,1)$, mas requer um mecanismo de determinação das posições i apropriadas para direcionar a mutação.

5.1.3. Funções Para Pseudocódigos

Para apresentar os algoritmos de forma didática e tal que eles possam ser reproduzidos com facilidade, serão apresentados *pseudocódigos* das rotinas implementadas para cada uma das ferramentas propostas. Os algoritmos serão introduzidos sob a forma de funções genéricas, onde um conjunto de parâmetros de entrada/saída deverão ser atribuídos. Um conjunto de funções básicas e suas respectivas descrições estão apresentadas a seguir (entre parênteses encontram-se os nomes dos algoritmos que utilizam estas funções):

1. `Ab := gera(N,L);`
   Gera uma população **Ab** contendo N anticorpos de comprimento L cada um. É válida para qualquer espaço de formas. Pode ser implementada por qualquer um dos modelos de medula óssea propostos na Seção 3.4.3.1. (Todos)

2. `f := afinidade(Ab,Ag(j,:));`
   Calcula a afinidade **f** da população **Ab** em relação ao antígeno $\mathbf{Ag}_j$. Pode ser diretamente ou inversamente proporcional à distância entre eles, dependendo do algoritmo e/ou aplicação, seguindo as regras apresentadas na Seção 3.4.2. (CLONALG, ABNET, aiNet)

3. `f := decodifica(Ab);`
   Decodifica e calcula a afinidade **f** da população **Ab** (cada elemento do vetor **f** representa a afinidade de um único indivíduo da população **Ab**) em relação a uma função objetivo $g(\cdot)$ a ser otimizada. (CLONALG)

4. $\mathbf{Ab}_{\{n\}}$ `:= select(Ab,f,n);`
   Seleciona n anticorpos da população **Ab** proporcionalmente à afinidade **f**, gerando uma subpopulação $\mathbf{Ab}_{\{n\}}$. (CLONALG, ABNET, aiNet)

5. `E := energy(Ab);`
   Calcula a energia da população **Ab** de acordo com as Equações (5.5) e (5.9). (SAND)

6. `C := clona(`$\mathbf{Ab}_{\{n\}}$`,β,f);`
   Clona (reproduz) os elementos da população $\mathbf{Ab}_{\{n\}}$ proporcionalmente à afinidade **f** e a um fator multiplicativo β, gerando uma população **C**. (CLONALG, aiNet)

7. **C*** := hypermut(**C**,**f**);
   Muta os elementos da população **C** proporcionalmente à afinidade **f**, gerando uma população **C***, de acordo com os mecanismos apresentados na Seção 3.4.3.4. (SAND, CLONALG)

8. **C*** := dmut(**C**,**Ag**,**f**);
   Mutação direcionada. Os elementos da população **C** são mutados proporcionalmente à afinidade **f** visando aumentar a afinidade ao(s) antígenos(s) **Ag**, gerando uma população **C***. (ABNET, aiNet)

9. $\mathbf{Ab}_{\{m\}}$:= insere($\mathbf{Ab}_{\{m\}}$,$\mathbf{Ab}_{\{d\}}$);
   Insere a matriz $\mathbf{Ab}_{\{d\}}$ na população $\mathbf{Ab}_{\{m\}}$ ou, dito de outra forma, concatena a matriz $\mathbf{Ab}_{\{m\}}$ com a matriz $\mathbf{Ab}_{\{d\}}$. (CLONALG, aiNet)

10. $\mathbf{Ab}_{\{r\}}$:= replace($\mathbf{Ab}_{\{r\}}$,$\mathbf{Ab}_{\{d\}}$,**f**);
    Substitui *d* anticorpos de $\mathbf{Ab}_{\{r\}}$ pelos *d* elementos da população $\mathbf{Ab}_{\{d\}}$ proporcionalmente à afinidade **f**. (CLONALG, aiNet)

11. [**Ab**,α] := split(**Ab**,$\boldsymbol{\nu}$,τ,ε,α_0);
    Clona (reproduz) um elemento de **Ab** proporcionalmente aos parâmetros **v**, τ, e ε (Seção 5.4), gerando um único clone. O parâmetro α_0 é passado de forma a permitir o retorno de $\alpha := \alpha_0$. (ABNET)

12. [**Ab**,α] := poda(**Ab**,τ,t,β,α_0);
    Elimina um elemento de **Ab** se $\tau = 0$ durante β iterações consecutivas. O parâmetro α_0 é passado de forma a permitir o retorno de $\alpha := \alpha_0$. (ABNET)

13. [$\mathbf{Ab}_{\{m\}}$,**S**] := suppress($\mathbf{Ab}_{\{m\}}$,**S**,σ_s);
    Elimina elementos de $\mathbf{Ab}_{\{m\}}$ cuja afinidade $s_{i,j}$ é inferior ao parâmetro σ_s, gerando uma nova população $\mathbf{Ab}_{\{m\}}$ e uma nova matriz de afinidades **S**. (aiNet)

14. **S** := afinidade(**Ab**,**Ab**);
    Calcula a afinidade **S** da população **Ab** em relação a si própria. Neste caso, a afinidade é inversamente proporcional à distância entre os anticorpos. (aiNet)

5.2. SAND: Simulated Annealing Aplicado ao Problema de Diversidade Populacional

Nesta seção, apresentamos uma abordagem baseada no algoritmo de *Simulated Annealing* (Kirkpatrick *et al.*, 1987) cujo objetivo principal é gerar uma população de candidatos à solução que apresenta uma ampla cobertura do espaço de buscas. A estratégia não emprega conhecimento prévio do problema a ser tratado e produz uma população de candidatos à solução com cardinalidade fixa e conhecida. Este algoritmo foi inicialmente proposto por de Castro & Von Zuben (1999a). O algoritmo, denominado SAND (*Simulated ANnealing approach for Diversity*), possui duas versões: uma primeira para espaços de formas binários de Hamming e outra para espaços Euclidianos.

5.2.1. Motivação e Escopo de Aplicações

O problema de geração de condições iniciais ótimas, ou seja, de geração de candidatos à solução, possui aplicabilidade em diversas áreas, desde os métodos de otimização mais clássicos, como métodos de gradiente, até as abordagens de sistemas inteligentes, particularmente as redes neurais artificiais e os algoritmos evolutivos discutidos no Capítulo 4.

Uma das principais características do sistema imunológico é sua capacidade de gerar receptores antigênicos altamente diversificados sendo, portanto, capazes de reconhecer uma enorme quantidade de moléculas. Este mecanismo é *antecipatório*, no sentido de que nenhum conhecimento prévio sobre os antígenos precisa ser considerado para a geração dos receptores (anticorpos). Entretanto, em qualquer momento da vida de um indivíduo, o sistema imunológico mantém uma quantidade finita de elementos (células e moléculas) circulantes e, portanto, uma variedade limitada de receptores. Sendo assim, é crucial que o sistema imunológico otimize o uso (e aplicação) destes recursos limitados, mantendo receptores em regiões "estratégicas" do espaço de possíveis antígenos.

O algoritmo SAND induz diversidade em uma população de tamanho fixo através da maximização de funções de energia que consideram a afinidade entre os elementos que compõem um repertório de anticorpos. A população de antígenos a ser posteriormente reconhecida é considerada desconhecida. Portanto o método opera de forma antecipatória como no caso do sistema imunológico biológico. Para a maximização da diversidade do repertório de anticorpos, são propostas duas *funções de energia* a serem utilizadas no processo iterativo de busca do algoritmo de Simulated Annealing. O algoritmo pode ser aplicado aos processos de geração de condições iniciais para ferramentas de engenharia imunológica e de outras áreas, como conjuntos de pesos iniciais para o treinamento de RNAs e até populações de cromossomos a serem utilizadas em algoritmos de computação evolutiva.

5.2.2. Simulated Annealing

O algoritmo de *Simulated Annealing* (Recozimento Simulado) faz uma conexão entre a mecânica estatística e a otimização combinatorial (Kirkpatrick *et al*., 1987; Haykin, 1999). A origem do método está associada a propriedades agregadas de uma grande quantidade de átomos encontrados em amostras de líquidos ou em elementos sólidos. O comportamento do sistema em equilíbrio térmico, a uma dada temperatura, pode ser caracterizado experimentalmente por pequenas flutuações em torno de um comportamento médio.

A temperatura é simplesmente um parâmetro de controle que produz efeitos na função de energia (ou função de custo). O processo de simulated annealing consiste em inicialmente "aquecer" o sistema sendo otimizado a uma temperatura bastante elevada, e depois reduzir gradativamente a temperatura até que o sistema "congele" e nenhuma variação possa ser observada. Etapas de elevação de temperatura também podem ser incorporadas. A cada temperatura, a simulação deve proceder por tempo suficiente para que o sistema atinja um estado estacionário.

A cada iteração deste algoritmo, uma pequena perturbação aleatória é dada em um átomo, e a variação ΔE resultante na energia do sistema é calculada. Se $\Delta E \leq 0$, a perturbação é aceita, e a nova configuração é utilizada como condição inicial da próxima iteração. O caso $\Delta E > 0$ é tratado probabilisticamente: a probabilidade de que a configuração seja aceita é dada pela Equação (5.1):

$$P(\Delta E) = \exp(-\Delta E/T), \quad \textbf{(5.1)}$$

onde E é a energia da configuração, T a temperatura e ΔE uma pequena perturbação na energia medida.

Assim, transições para fora de ótimos locais são sempre possíveis a temperaturas não nulas. A seqüência de temperaturas e a dimensão das variações ΔE necessárias para atingir o equilíbrio a cada temperatura são denominadas de *annealing schedule*.

5.2.3. Descrição e Análise do SAND

Seja a seguinte notação:

- T: temperatura da configuração;
- **Ab**: repertório total de N anticorpos (**Ab** $\in S^{N \times L}$);
- $\mathbf{Ab}_T$: repertório temporário gerado pela perturbação de **Ab** ($\mathbf{Ab}_T \in S^{N \times L}$);
- E: energia da configuração **Ab**;
- E_T: energia da configuração temporária $\mathbf{Ab}_T$;
- α: taxa de hipermutação a ser aplicada na geração de $\mathbf{Ab}_T$; e
- β: decrescimento geométrico da temperatura (*cooling schedule*).

Dentro do escopo de engenharia imunológica, o algoritmo SAND pode ser descrito como a seguir:

1. Gere um repertório **Ab** inicial de N anticorpos;
2. Calcule a energia E deste repertório (baseado nas funções a serem descritas posteriormente);
3. Provoque uma pequena perturbação no repertório **Ab**, proporcional a α, gerando um repertório temporário $\mathbf{Ab}_T$ e reduza a taxa α de mutação;
4. Calcule a energia E_T deste repertório $\mathbf{Ab}_T$ e avalie ΔE;
   4.1. Caso $\Delta E < 0$, aceite a perturbação e retorne ao Passo 3;
   4.2. Caso $\Delta E > 0$, trate probabilisticamente, de acordo com a Equação (5.1), e retorne ao Passo 3;
   4.3. Caso $\Delta E = 0$, avalie a quantidade de iterações (*ct*) na qual o sistema permanece em estado estacionário. Caso este número de iterações ultrapasse um limiar pré-definido δ, reduza a temperatura T do sistema, restaure a taxa α de mutação e retorne ao Passo 3; caso contrário, retorne ao Passo 3, mas mantenha temperatura e mutação constantes.

O fluxograma da Figura 5.2 descreve o comportamento do algoritmo.

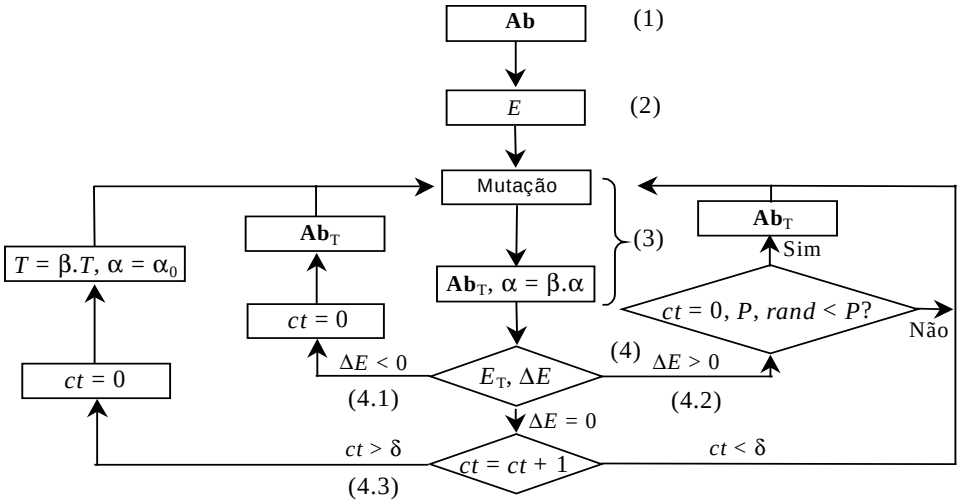


Figura 5.2. Diagrama de blocos do algoritmo SAND (etapas de elevação da temperatura *T* não foram incorporadas).

5.2.3.1. Pseudocódigo

Considere a implementação de uma função computacional para simular o algoritmo SAND. Esta função pode ter como argumentos os seguintes parâmetros:

Entrada:

- Comprimento *L* dos anticorpos;
- O tamanho *N* da população de anticorpos a ser gerada;
- Quantidade *gen* de iterações a serem executadas (possível critério de parada);
- A taxa β de decrescimento geométrico da temperatura;
- Limiar δ de iterações de estado estacionário para a redução da temperatura; e
- O valor máximo *mE* da energia do sistema (geralmente muito próximo de 100%, a ser descrito pelas Equações (5.5) e (5.9)).

Saída:

- Repertório **Ab** de anticorpos otimizados (maximamente diversificados).

Dados os parâmetros de entrada e saída, é possível descrever um pseudocódigo para a implementação do SAND como a seguir:

```
function [Ab] = sand(N,L,gen,β,δ,mE);
t  := 0;
ct := 0;
T  := 1;
α0 := 1;
Ab := gera(N,L);              % Passo 1
% A cada iteração, faça
while t < gen & E < mE,
   E  := energy(Ab);          % Passo 2
   Ab_T := hypermut(Ab,α);    % Passo 3
   E_T  := energy(Ab_T);      % Passo 4
```

```
   dE := E - E_T;                % Passo 4
   if dE < 0,                    % Passo 4.1
      Ab := Ab_T;
      E  := E_T;
      ct := 0;
   elseif dE > 0,                % Passo 4.2
      if rand < exp(-dE/T),
         Ab := Ab_T;
         E  := E_T;
         ct := 0;
      end,
   else,                         % Passo 4.3
      ct := ct + 1;
      if ct mod δ = 0,
         T  := β * T;
         ct := 0;
         α  := α_0;
      end;
      α  := β * α;
   end;
end;
```

5.2.3.2. Espaço de Formas de Hamming

Para implementação do algoritmo, é preciso propor as equações para o cálculo da energia do sistema nos espaços de formas de Hamming e Euclidiano.

No primeiro caso, ou seja, em um espaço de formas binário de Hamming, dado um repertório de N anticorpos de comprimento L, considere as seguintes equações:

$$s(i) = \begin{cases} 1, & \boldsymbol{Ab}_i \neq \boldsymbol{Ab}_j, \forall j \\ 0, & \text{outros casos} \end{cases}, \tag{5.2}$$

$$D(\%) = 100.\frac{1}{N} \times \sum_{i=1}^{N} s(i), \tag{5.3}$$

$$F(\%) = 100.\frac{4}{L.N^2} \times \sum_{i=1}^{N} \sum_{j=i+1}^{N} HD(i,j), \tag{5.4}$$

$$E(\%) = \frac{F(\%) + D(\%)}{2}. \tag{5.5}$$

Lembrando que o objetivo é gerar e manter a diversidade, cada uma destas equações contribui para a determinação da energia do sistema da seguinte forma:

- O vetor $s(i)$, $i = 1,...,N$, na Equação (5.2), apresenta valor 1 somente quando cada anticorpo $\boldsymbol{Ab}_i$ for diferente de todos os outros anticorpos do repertório, ou seja, cada anticorpo possui uma única especificidade;
- $D(\%)$, na Equação (5.3), é o percentual de indivíduos distintos em relação a todo o repertório;

- Na Equação (5.4), F(%) é a distância de Hamming (dada pela Equação 3.3) percentual entre todos os anticorpos do repertório. F será máximo (F = 100%) quando, para $N \leq 2^L$, a distância de Hamming entre todos os anticorpos do repertório for máxima. Para $N > 2^L$, F nunca atingirá seu valor máximo, pois haverá pelo menos $N - 2^L$ anticorpos repetidos no repertório. Oprea (1999) demonstrou que a distância de Hamming máxima entre os elementos de um repertório binário de anticorpos é dada por $L.N/(2.(N-1))$, critério que pode ser empregado como parte da função de energia do SAND para espaços de formas de Hamming; e
- E(%), dado pela Equação (5.5), é a energia da configuração que mede a diversidade do repertório **Ab**.

Note que se o número de indivíduos na população é menor do que a quantidade total de indivíduos diferentes que podem ser gerados para uma cadeia binária de atributos de comprimento L ($N = 2^L$), então diferentes populações podem surgir para a mesma energia máxima (100%), e o repertório ótimo não é único. Esta população de tamanho limitado pode nunca alcançar uma máxima cobertura do espaço de formas, para $\varepsilon = 0$ (limiar de afinidade nulo), mas certamente irá atingir algum estado próximo da máxima cobertura possível, dado seu tamanho restrito e valor de ε pré-definidos.

5.2.3.3. Espaço de Formas Euclidiano

Para que possamos aplicar o algoritmo SAND a espaços de formas com valores reais, uma outra função de energia deve ser proposta. Neste caso, medidas de distância, ou afinidade, Euclidiana ou de Manhattan podem ser empregadas.

Considerando um espaço de formas Euclidiano, a nova medida de energia a ser otimizada pode ser simplesmente definida como a soma das distâncias Euclidianas entre todos os vetores que representam os anticorpos do repertório imunológico:

$$E = \sum_{i=1}^{N} \sum_{j=i+1}^{N} ED(i, j) \,, \qquad \textbf{(5.6)}$$

onde ED é uma matriz quadrada e simétrica que contém a distância Euclidiana entre todos os anticorpos do repertório, de acordo com a Equação 3.1.

No caso dos espaços de Hamming, o critério de parada adotado para o algoritmo SAND pode ser E = 100%. No espaço Euclidiano, a medida de energia não é um valor percentual e, portanto, outro critério de parada considerando a diversidade populacional deve ser apresentado.

A abordagem proposta, entre outras possibilidades, envolve a análise de dados direcionais. Dado um conjunto de anticorpos (vetores) $\boldsymbol{Ab}_i$, $i = 1,..., N$, inicialmente é necessário transformá-los em vetores unitários, ou normalizá-los, resultando em um conjunto **I** de N vetores unitários de comprimento L ($\boldsymbol{I}_i \in \Re^L$, $i = 1,...,N$). O vetor direcional médio é

$$\bar{\boldsymbol{I}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{I}_i \,. \qquad \textbf{(5.7)}$$

Supondo uma medida de distância Euclidiana, uma métrica para determinar a diversidade máxima dentro de um conjunto de vetores unitários pode ser simplesmente dada por

$$\overline{R} = \left(\bar{\boldsymbol{I}}^T.\bar{\boldsymbol{I}}\right)^{1/2}, \quad \textbf{(5.8)}$$

onde ^{T} representa a operação de transposição matricial e $\overline{R}$ corresponde à distância do vetor médio à origem do sistema de coordenadas. A Equação (5.8) representa a amplitude do vetor resultante, porém nada pode ser inferido sobre a posição relativa entre os anticorpos da população (vetores individuais).

O critério de parada *SC* a ser empregado está baseado no índice I_{SC}

$$I_{SC}(\%) = 100 \cdot (1 - \overline{R}). \quad \textbf{(5.9)}$$

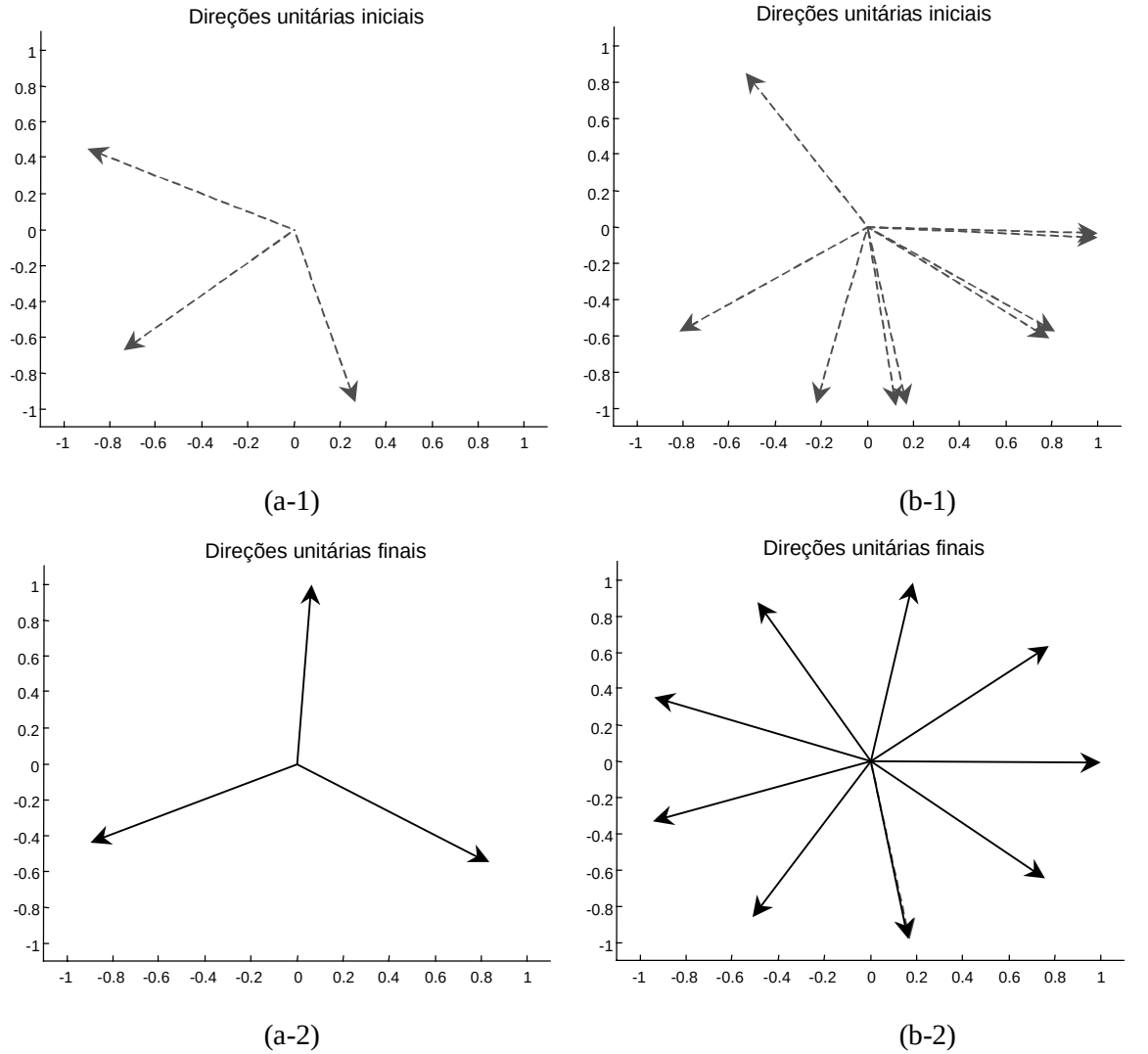


Figura 5.3. Geração de direções unitárias utilizando o algoritmo SAND para os espaços de formas Euclidianos. (a-1, a-2) Três vetores (N = 3). (b-1, b-2) Nove vetores (N = 9). As setas tracejadas indicam o conjunto inicial de vetores (direções) (a-1,b-1), e as setas sólidas representam os vetores (direções) otimizados (a-2, b-2).

A Equação (5.9) corresponde à medida percentual de diversidade e será igual a 100% quando $\overline{R}$ for zero. Em termos práticos, um valor de I_{SC} próximo de 100% para o critério de parada é uma escolha razoável.

A geração de um repertório diverso de anticorpos em $\Re^L$ corresponde a produzir um conjunto de direções que esteja bem distribuídas em uma hiperesfera. Para propósitos de visualização, considere o caso de gerar um conjunto de vetores em $\Re^2$ que apresente a melhor cobertura deste espaço. A Figura 5.3(a) e (b) apresenta os vetores evoluídos (otimizados) para $N = 3$ e $N = 9$, respectivamente. Note que o algoritmo é capaz de gerar direções com cobertura máxima do espaço $\Re^2$. O algoritmo SAND permite a obtenção de resultados similares para quaisquer valores de N e L, o que não pode ser obtido por nenhum outro algoritmo conhecido na literatura, principalmente para valores elevados de L.

5.2.4. Caracterização e Comparação

É preciso ressaltar algumas características fundamentais deste algoritmo. Primeiramente, foi discutido nos Capítulos 2 e 3 que o processo de reconhecimento antigênico efetuado pelo sistema imune adaptativo dá-se pela complementaridade de formas entre um antígeno e um anticorpo, ou seja, quanto mais complementares as moléculas, maiores suas afinidades. Sem perda de generalidade, o algoritmo SAND considera a afinidade entre anticorpos como sendo inversamente proporcional à distância entre eles, ou seja, o SAND tem por objetivo gerar anticorpos cujas formas (representadas por cadeias de atributos ou vetores) sejam as mais complementares possíveis. Neste caso, a afinidade entre eles é mínima, evitando que um anticorpo *i* reconheça um anticorpo *j* do repertório cujos atributos são complementares a ele. Entretanto, como o algoritmo SAND gera um repertório de anticorpos com máxima distância (complementaridade) entre seus elementos, o conceito de reconhecimento antigênico por complementaridade de formas pode ser diretamente empregado para uma dada população antigênica e repertório resultante. De forma recíproca, a afinidade Ag-Ab também pode ser avaliada por uma medida de similaridade, permitindo concluir que a estratégia adotada pelo SAND é aplicável aos dois casos. A avaliação de afinidade empregando medida de similaridade, ao invés de complementaridade, tem sido bastante utilizada em sistemas imunológicos artificiais, como em Hajela & Lee (1996), Oprea (1999) e Harmer & Lamont (2000), dentre vários outros.

A determinação da taxa de mutação α é um parâmetro crítico neste algoritmo e pode seguir os padrões propostos para os procedimentos de hipermutação somática durante uma resposta imune adaptativa, onde indivíduos com alta afinidade antigênica sofrem baixas taxas de mutação e vice-versa. No caso do SAND, a população como um todo determina uma única afinidade (energia).

De acordo com a descrição da Seção 5.2.3, a taxa de mutação α segue um padrão similar ao proposto por Kepler & Perelson (1993a,b), onde gerações com baixas taxas de mutação são intercaladas por picos de mutações, e mutações benéficas (capazes de aumentar a afinidade antigênica) são adquiridas seqüencialmente ao invés de simultaneamente. A Figura 5.4 ilustra o comportamento da taxa de mutação para a versão de Hamming do algoritmo SAND, dada a seguinte linha de comando: `[Ab] = sand(128,7,200,0.9,3,100)`.

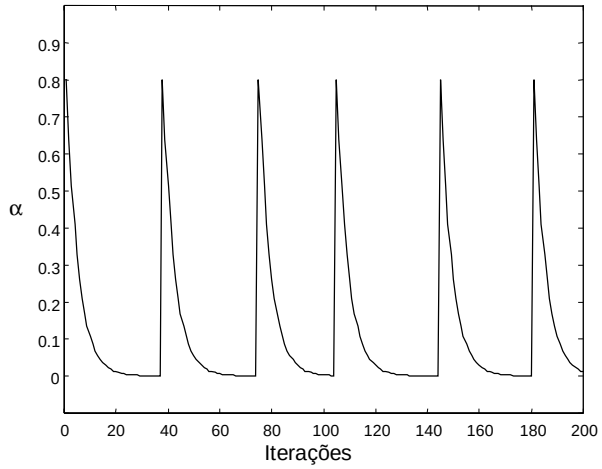


Figura 5.4. Comportamento típico da taxa de mutação (α) para o algoritmo SAND.

Em aplicações típicas de algoritmos genéticos (Seção 4.4), a estratégia evolutiva produz indivíduos cada vez mais aptos (com maior fitness) dentre uma vasta população de candidatos à solução. O algoritmo é utilizado para evoluir uma população de candidatos à solução onde um único indivíduo irá especificar a melhor solução, ou seja, um pico de fitness. Os operadores de seleção e recombinação (crossover, mutação e inversão) empregados no GA guiam a população em direção ao seu melhor indivíduo (Mitchell, 1998). Nas aplicações do SAND, busca-se uma população com caráter cooperativo, que, sob a forma de um conjunto de indivíduos, seja capaz de realizar uma tarefa específica: maximizar a cobertura do espaço de buscas, baseado em uma população de tamanho fixo. Sob este ponto de vista, não existe um único indivíduo da população que possa ser considerado o melhor, mas toda a população irá representar a solução. Este objetivo sugere que o algoritmo genético clássico não será capaz de produzir o resultado desejado e, ao invés de desenvolver um GA capaz de manter populações diversas (Forrest *et al*., 1993; Smith *et al*., 1993; Mahfoud, 1995; Matsui, 1999), optamos pelo algoritmo de simulated annealing, que também possui a vantagem de ser menos custoso computacionalmente para algumas aplicações.

A abordagem do SAND possui uma essência similar à abordagem Michigan empregada nos sistemas classificadores, propostos por Holland (1975). A abordagem Michigan pode ser vista como um modelo computacional da cognição, onde o conhecimento da entidade é expresso como um coleção de regras que interagem com o ambiente e sofre modificações. Assim como no nosso método de simulated annealing, todas as regras (ou seja, toda a população) representam um único candidato à solução.

5.2.5. Complexidade Computacional

O algoritmo SAND proposto para geração de diversidade possui duas etapas principais de processamento: 1) a determinação da energia da configuração (Passos 2 e 4), e 2) a mutação do repertório **Ab** de anticorpos (Passo 3).

Tabela 5.1. Complexidade computacional por geração para execução do algoritmo SAND, onde N é o tamanho da população de anticorpos de comprimento L.

	Tempo de Processamento	***Memória***
Espaço de Hamming	$O(N^2)$	$\propto N\,[2L + L(N-1)/2 + 1]$
Espaço Euclidiano	$O(N^2)$	$\propto L\,[2N + N(N-1)/2 + 1]$

O custo computacional do algoritmo é quadrático em relação à quantidade N de anticorpos, pois demanda a comparação de cada anticorpo do repertório com todos os outros (Equações (5.4) e (5.6)), possuindo um custo computacional associado da ordem de $O(N^2)$. O processo de mutação de toda a população tem custo computacional $O(N.L)$, como discutido na Seção 5.1.2. Assim, somando-se a complexidade das duas etapas e considerando a característica assintótica desta análise, é possível determinar a complexidade computacional total do algoritmo, como apresentado na Tabela 5.1.

Para a determinação da quantidade de memória necessária para executar o algoritmo, pode ser feita uma análise distinta para os espaços de Hamming e Euclidiano. Em ambos os casos, são armazenadas duas matrizes, **Ab** e $\mathbf{Ab}_{\mathrm{T}}$, contendo os repertórios de anticorpos atual e temporário, ambas de dimensão $S^{N\times L}$. Além disso, as matrizes que determinam a distância entre os anticorpos são simétricas e, portanto, é possível armazenar apenas suas partes triangulares superiores, de dimensão $S^{(N(N-1)/2)\times L}$. No caso dos espaços binários de Hamming, uma cadeia binária de dimensão L, também é armazenada e, para os espaços Euclidianos, um vetor resultante de comprimento L deve existir. A quantidade de memória associada a estes algoritmos é apresentada na Tabela 5.1.

5.3. CLONALG: Implementações Computacionais do Princípio de Seleção Clonal

Nesta seção, são descritas duas versões para a implementação computacional do princípio de seleção clonal, denominadas genericamente de CLONALG (*CLONal selection ALGorithm*), considerando explicitamente o processo de maturação de afinidade. Este algoritmo foi originalmente proposto por de Castro & Von Zuben (2000a) e brevemente apresentado na Seção 3.4.3.3. Duas versões do algoritmo estão disponíveis: a primeira para resolver problemas de aprendizagem de máquina e reconhecimento de padrões e a segunda para resolver problemas de otimização (de Castro & Von Zuben, 2001a).

5.3.1. Motivação e Escopo de Aplicações

Nas Seções 2.6 e 2.8 verificou-se que o princípio de seleção clonal é utilizado para descrever as características básicas de uma resposta imune adaptativa a estímulos antigênicos. Ele estabelece a idéia de que somente as células capazes de reconhecer antígenos irão se proliferar, sendo assim selecionadas em detrimento das outras. Durante a reprodução celular, estas células (clones) estarão sujeitas a um processo de mutação

somática com taxas elevadas que, juntamente com uma forte pressão seletiva, resultará no aumento da afinidade entre o anticorpo e o antígeno que o selecionou. Dessa forma, é possível concluir que estes processos são parcialmente responsáveis pela aprendizagem e memória imunológica, características particularmente interessantes sob o ponto de vista de engenharia. Além da capacidade de aprendizagem e memória, foi argumentado, na Seção 4.5, que a evolução celular efetuada pelo sistema imunológico constitui uma espécie de microevolução, onde variações ontogenéticas (dentro do indivíduo) podem de alguma forma contribuir para uma maior adaptabilidade do indivíduo ao seu ambiente. Dadas essas características do processo biológico, é suposto que a implementação computacional do algoritmo de seleção clonal pode levar ao desenvolvimento de ferramentas adaptativas capazes de resolver problemas de aprendizagem de máquina, reconhecimento de padrões e otimização, em que os processos de seleção e maturação são responsáveis pela geração de indivíduos cada vez melhores adaptados ao ambiente no qual estão inseridos.

5.3.2. Descrição e Análise do CLONALG

O algoritmo foi inicialmente proposto para resolver problemas de aprendizagem de máquina e reconhecimento de padrões (antígenos), onde uma população aleatória de anticorpos está presente e tem por objetivo aprender a reconhecer um conjunto de antígenos. Dadas suas características adaptativas, o algoritmo foi estendido para aplicações a problemas de otimização.

5.3.2.1. Reconhecimento de Padrões

Para facilitar a descrição do algoritmo, considere o diagrama de blocos da Figura 5.5. Para o problema de reconhecimento de padrões, é suposta a existência de um repertório inicial aleatório de anticorpos com a finalidade de reconhecer uma população conhecida de antígenos. Como proposto por alguns imunologistas, e discutido na Seção 2.8.1, existe um conjunto de memória, composto por indivíduos específicos do repertório de anticorpos apresentando altas afinidades antigênicas, que estará funcionalmente desconectado do restante da população. Seja a seguinte notação:

- **Ab**: repertório de anticorpos disponíveis ($\mathbf{Ab} \in S^{N\times L}$, $\mathbf{Ab} = \mathbf{Ab}_{\{r\}} \cup \mathbf{Ab}_{\{m\}}$);
- $\mathbf{Ab}_{\{m\}}$: repertório de anticorpos de memória ($\mathbf{Ab}_{\{m\}} \in S^{m\times L}$, $m \leq N$);
- $\mathbf{Ab}_{\{r\}}$: restante do repertório de anticorpos ($\mathbf{Ab}_{\{r\}} \in S^{r\times L}$, $r = N - m$);
- **Ag**: população de M antígenos a serem reconhecidos ($\mathbf{Ag} \in S^{M\times L}$);
- $\boldsymbol{f}_j$: vetor contendo a afinidade de todos os anticorpos em relação ao antígeno $\boldsymbol{Ag}_j$;
- $\mathbf{Ab}^j_{\{n\}}$: subconjunto de **Ab** composto pelos n anticorpos de maior afinidade a $\boldsymbol{Ag}_j$ ($\mathbf{Ab}^j_{\{n\}} \in S^{n\times L}$, $n \leq N$);
- $\mathbf{C}^j$: população de N_c clones gerada a partir de $\mathbf{Ab}^j_{\{n\}}$ ($\mathbf{C}^j \in S^{Nc\times L}$);
- $\mathbf{C}^j$*: população $\mathbf{C}^j$ após o processo de maturação de afinidade;
- $\mathbf{Ab}_{\{d\}}$: d anticorpos com baixa afinidade antigênica de $\mathbf{Ab}_{\{r\}}$ serão substituídos por $\mathbf{Ab}_{\{d\}}$ novos anticorpos ($\mathbf{Ab}_{\{d\}} \in S^{d\times L}$, $d \leq r$); e
- $\boldsymbol{Ab}^j_i$*: candidato de $\mathbf{C}^j$* a fazer parte do conjunto de memória.

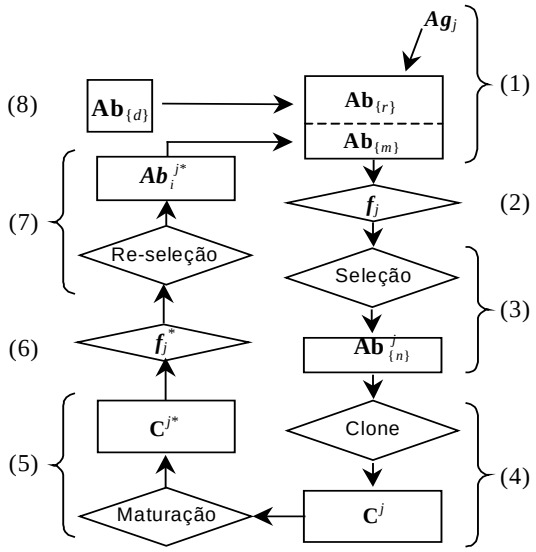


Figura 5.5. Diagrama de blocos do algoritmo de seleção clonal, CLONALG, para o problema de reconhecimento de padrões.

O algoritmo pode ser descrito como a seguir (Figura 5.5). Dada a existência de um repertório **Ab** de anticorpos composto por um conjunto de memória $\mathbf{Ab}_{\{m\}}$ mais o restante $\mathbf{Ab}_{\{r\}}$ da população, e supondo, sem perda de generalidade, que a quantidade m de anticorpos de memória é igual à quantidade M de antígenos a serem reconhecidos.

1. Em um instante de tempo t, um antígeno $\boldsymbol{Ag}_j$ ($\boldsymbol{Ag}_j \in \mathbf{Ag}$) é apresentado a todos os anticorpos da população **Ab**;
2. Um vetor de afinidades $\boldsymbol{f}_j$ em relação aos anticorpos da população **Ab** é determinado;
3. Do conjunto **Ab**, um subconjunto $\mathbf{Ab}^j_{\{n\}}$, composto pelos n anticorpos com maiores afinidades a $\boldsymbol{Ag}_j$, é selecionado baseado na afinidade $f_{i,j}$ de cada anticorpo $\boldsymbol{Ab}_i$ em relação ao antígeno $\boldsymbol{Ag}_j$;
4. Os n indivíduos selecionados irão se proliferar (clonagem) proporcionalmente às suas afinidades ao antígeno, gerando uma população $\mathbf{C}^j$ de clones: quanto maior a afinidade, maior o número de clones de cada um dos n anticorpos;
5. Em seguida, a população $\mathbf{C}^j$ de clones é submetida ao processo de maturação de afinidade gerando uma nova população $\mathbf{C}^{j*}$, onde cada anticorpo irá sofrer uma mutação com taxa inversamente proporcional a sua afinidade: quanto maior a afinidade, menor a taxa de mutação;
6. Determine a afinidade $\boldsymbol{f}_j^*$ entre o conjunto $\mathbf{C}^{j*}$ de clones mutados e o antígeno $\boldsymbol{Ag}_j$;
7. Desta população madura $\mathbf{C}^{j*}$, re-selecione o melhor anticorpo para ser um candidato a entrar no conjunto de memória $\mathbf{Ab}_{\{m\}}$. Se a afinidade deste anticorpo $\boldsymbol{Ab}_i^{j*}$ em relação a $\boldsymbol{Ag}_j$ for maior do que a afinidade do respectivo anticorpo de memória, então $\boldsymbol{Ab}_i^{j*}$ substitui este anticorpo de memória;
8. Finalmente, substitua d anticorpos de $\mathbf{Ab}_{\{r\}}$ por $\mathbf{Ab}_{\{d\}}$ novos indivíduos, induzindo diversidade no repertório (alguns membros de $\mathbf{Ab}_{\{d\}}$ podem ser escolhidos a partir

de $\mathbf{C}^j$* ou gerados utilizando um dos modelos de medula óssea propostos na Seção 3.4.3.1). Os anticorpos com menores afinidades são escolhidos para serem substituídos.

Após a apresentação de todos os antígenos ao repertório de anticorpos, diz-se que ocorreu uma *geração*.

Nesta implementação, a quantidade N_c de clones gerada no Passo 4 para todos os n anticorpos selecionados no Passo 3 é dada pela Equação (5.10):

$$N_c = \sum_{i=1}^{n} round(\beta . N / i), \quad \textbf{(5.10)}$$

onde β é um fator multiplicativo, N é a quantidade total de anticorpos do repertório **Ab** e *round*(·) é o operador que arredonda o valor entre parênteses para o inteiro mais próximo. Neste caso, os anticorpos estão ordenados da maior para a menor afinidade, portanto quanto maior i na Equação (5.10), menor a afinidade e consequentemente a quantidade de clones gerada para $\boldsymbol{Ab}_i$.

5.3.2.1.1. Pseudocódigo

Considere a implementação de uma função computacional para simular o algoritmo CLONALG para o problema de aprendizagem de máquina e reconhecimento de padrões. Esta função pode ter como argumentos os seguintes parâmetros:

Entrada:

- Comprimento L dos antígenos e anticorpos;
- População **Ab** composta por N anticorpos de comprimento L;
- População **Ag** composta por M antígenos de comprimento L;
- Quantidade *gen* de gerações a serem executadas (critério de parada);
- Número n de anticorpos a serem selecionados para clonagem;
- Fator multiplicativo β usado na definição da quantidade de clones; e
- Quantidade d de anticorpos com baixa afinidade $\boldsymbol{f}$, que serão substituídos.

Saída:

- Matriz $\mathbf{Ab}_{\{m\}}$ de anticorpos de memória; e
- Afinidade total **F** entre os anticorpos de memória ($\mathbf{Ab}_{\{m\}}$) e a população de antígenos **Ag**.

Dados os parâmetros de entrada e saída, é possível descrever um pseudocódigo para a implementação do CLONALG como a seguir:

```
function [Ab{m},F] = clonalg(Ab,Ag,L,gen,n,β,d);
t := 0;
F := 0;
% A cada geração, faça:
while t < gen & F < (M*L),
   for j = 1 to M,                          % Passo 1
      f(j,:)   := afinidade(Ab,Ag(j,:));    % Passo 2
      Ab{n}(j,:):= select(Ab,f,n);          % Passo 3
```

```
      C(j,:)   := clona(Ab{n},β,f);        % Passo 4
      C*(j,:)  := hypermut(C,f);           % Passo 5
      f(j,:)   := afinidade(C*,Ag(j,:));   % Passo 6
      Ab*      := select(C*,f,1);          % Passo 7
      Ab{m}(j,:):= insere(Ab{m}(j,:),Ab*); % Abm ← Ab se f(Ab*)>f(Abm(j,:))
      Ab{d}    := gera(d,L);               % Gera Abd aleatoriamente
      Ab{r}    := replace(Ab{r},Ab{d},f);  % Passo 8
   end;
   f := afinidade(Ab{m},Ag);
   F := soma(f);
   t := t + 1;
end;
```

5.3.2.2. Otimização

Para aplicar o algoritmo de seleção clonal proposto acima, CLONALG, a problemas de otimização, algumas poucas modificações devem ser efetuadas:

- No Passo (1) não existe uma população **Ag** de antígenos a serem reconhecidos, e sim uma função objetivo $g(\cdot)$ a ser otimizada. Dessa forma, a afinidade de um anticorpo (que pode ser interpretado como seu fitness, ou grau de adaptabilidade) passa a corresponder ao valor da função objetivo $g(\cdot)$ avaliada para um dado anticorpo: cada anticorpo $\boldsymbol{Ab}_i$ representa o valor codificado de um argumento da função $g(\cdot)$. Além disso, como não existe uma população específica de antígenos a serem reconhecidos, toda a população **Ab** corresponderá a anticorpos de memória ao final do processo, portanto não é necessário definir um subconjunto específico de memória $\mathbf{Ab}_{\{m\}}$;
- No Passo (7), n anticorpos serão selecionados para compor o conjunto **Ab**.

Se o objetivo do processo de otimização é determinar múltiplos ótimos dentro da população de anticorpos, então dois parâmetros podem apresentar valores previamente definidos para garantir um bom desempenho do algoritmo:

1. Supor $n = N$, ou seja, selecionar todos os anticorpos da população **Ab** para reprodução (clonagem) nos Passos (3) e (4), sendo cada anticorpo analisado localmente;
2. A reprodução proporcional à afinidade não necessariamente se aplica, significando que a quantidade de clones gerada para cada um dos N anticorpos é considerada a mesma, não privilegiando anticorpo algum por sua afinidade. A Equação (5.10), torna-se:

$$N_c = \sum_{i=1}^{n} round(\beta.N). \qquad \textbf{(5.11)}$$

O diagrama de blocos resultante do algoritmo CLONALG para aplicação a problemas de otimização está resumido na Figura 5.6.

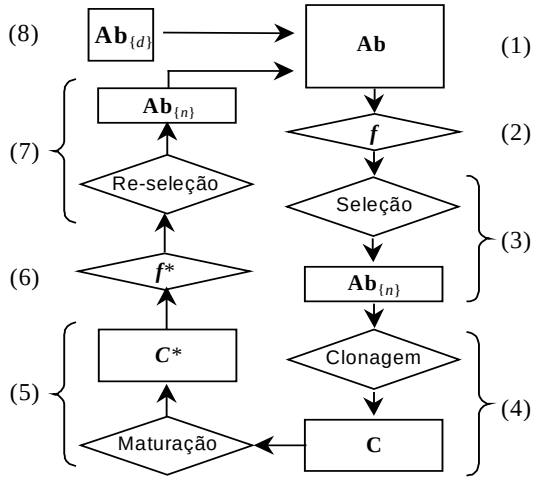


Figura 5.6. Diagrama de blocos do algoritmo de seleção clonal, CLONALG, para o problema de otimização.

5.3.2.2.1. Pseudocódigo

A implementação computacional do algoritmo CLONALG para otimização de funções pode ser feita utilizando o pseudocódigo abaixo, dados os parâmetros de entrada e saída:

Entrada:

- Comprimento *L* dos anticorpos;
- População **Ab** composta por *N* anticorpos;
- Quantidade *gen* de gerações a serem executadas (critério de parada);
- Número *n* de anticorpos a serem selecionados para clonagem (*n* = *N* para aplicações a problemas de otimização multimodal);
- Fator multiplicativo β usado na definição da quantidade de clones; e
- Quantidade *d* de anticorpos com baixa afinidade que serão substituídos.

Saída:

- Matriz **Ab**, correspondente aos anticorpos de memória; e
- Afinidade (fitness) ***f*** de cada anticorpo da matriz **Ab**.

```
function [Ab,f] = clonalg(Ab,L,gen,n,β,d)
% A cada geração, faça:
for t = 1 to gen,
   f      := decodifica(Ab);          % Passo 2
   Ab{n}  := select(Ab,f,n);          % Passo 3
   C      := clona(Ab{n},β,f);        % Passo 4
   C*     := hypermut(C,f);           % Passo 5
   f      := decodifica(C*);          % Passo 6
   Ab{n}  := select(C*,f,n);          % Passo 7
   Ab     := insere(Ab,Ab{n});        % Ab ← Ab{n}
   Ab{d}  := gera(d,L);               % Gera Ab{d} aleatoriamente
   Ab     := replace(Ab,Ab{d},f);     % Passo 8
end;
f := decodifica(Ab);
```

5.3.3. Caracterização e Comparação

O algoritmo CLONALG, como proposto acima, representa uma implementação computacional simplificada do princípio da seleção clonal dos linfócitos B durante uma resposta imune adaptativa (Seção 2.8). Tanto no caso de reconhecimento de padrões quanto para otimização, supõe-se que o organismo possui uma população **Ab** de anticorpos que serão estimulados por um antígeno (que pode ser um elemento $\boldsymbol{Ag}_j$ explícito a ser reconhecido, ou um valor da função objetivo a ser otimizada), e aqueles com maiores afinidades serão selecionados gerando uma população de clones. Durante o processo de clonagem, reprodução por mitose celular, alguns anticorpos sofrerão mutação genética com taxas elevadas e inversamente proporcionais a suas afinidades. Os melhores indivíduos dentre os mutados serão re-selecionados para compor um conjunto de memória. Indivíduos com baixa afinidade são substituídos, simulando o processo de apoptose e/ou edição de receptores.

Na Seção 4.4.2 apresentou-se os passos principais para a implementação de algoritmos evolutivos, resumidos sob a forma do diagrama de blocos da Figura 5.7. Enquanto os algoritmos evolutivos empregam um vocabulário inspirado na genética natural e são baseados na teoria da evolução neo-Darwiniana (Seção 4.4), o algoritmo de seleção clonal, CLONALG, utiliza o espaço de formas e a terminologia imunológica para descrever as interações de antígenos e anticorpos, no caso da aprendizagem de máquina e reconhecimento de padrões, e estudar a evolução celular no caso do procedimento para otimização. É possível verificar, comparando-se a Figura 5.7 com a Figura 5.5 ou a Figura 5.6, que o algoritmo CLONALG apresenta, dentre outras, as mesmas etapas de processamento dos algoritmos evolutivos, permitindo sua caracterização como uma estratégia evolutiva. Além disso, foi discutido na Seção 4.5 que a evolução dentro do sistema imunológico pode ser vista como uma microevolução da teoria Darwiniana.

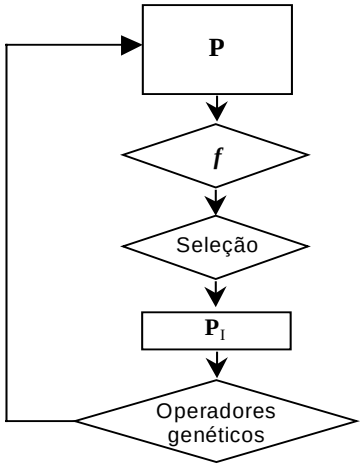


Figura 5.7. Diagrama de blocos de um algoritmo evolutivo básico. **P** corresponde a população de indivíduos, $\boldsymbol{f}$ é o fitness da população e $\mathbf{P}_I$ a população intermediária. Os operadores genéticos englobam a reprodução e variação genética dos descendentes, via crossover, mutação, etc.

Assim como no processo de reprodução celular envolvido no princípio da seleção clonal (mitose), no algoritmo CLONALG, o único operador responsável pela introdução direta de variação genética no repertório celular é uma mutação cuja taxa é elevada e inversamente proporcional à afinidade Ag-Ab. Dessa forma, verifica-se que existe uma correlação entre a taxa de variação da população e a afinidade, ou fitness, dos indivíduos, resultando num processo de auto-adaptação da taxa de variação genética a ser introduzida. Outro aspecto importante deste modelo é que cada indivíduo da população é analisado localmente: um conjunto de clones é gerado com cardinalidade proporcional à sua afinidade (Equações (5.10) e (5.11)), e o processo de seleção é efetuado na população de clones após sua maturação. Este procedimento sugere que o algoritmo é apropriado para solucionar problemas de otimização multimodal, onde cada anticorpo irá explorar sua vizinhança.

As *estratégias evolutivas* (EEs) também empregam uma abordagem de auto-adaptação dos mecanismos de recombinação e mutação (Tabela 4.2). Rechenberg (1973, 1994) propôs uma "janela de evolução" (*evolution window*), na qual variações genéticas em um indivíduo (perturbações causadas pela mutação) resultam em um progresso evolutivo apenas quando eles estão contidos em uma faixa estreita, confinada e determinável. Passos de mutação e recombinação fora da janela de evolução são ineficientes. Sendo assim, as estratégias evolutivas permitem o ajuste dinâmico das taxas de mutação durante a evolução; processo chamado de *meta-evolução* ou *otimização de segundo tipo*. As estratégias evolutivas concentram na mutação o principal operador do processo de variação genética, e portanto de busca, e a recombinação é necessária principalmente para o controle da mutação. As EEs utilizam mutações Gaussianas e regras determinísticas para a variação do desvio padrão das mutações, que devem estar linearmente correlacionadas (Bäck & Schwefel, 1993). Além disso, elas também não fazem a codificação da população. Por outro lado, a abordagem proposta para o algoritmo CLONALG é plausível sob o ponto de vista biológico e segue o padrão proposto na Seção 3.4.3.4, Figura 3.16. O que deve ser verificado é que os valores a serem adotados para as taxas de recombinação genética de algoritmos evolutivos, incluindo recombinação e mutação, devem ser proporcionais ao fitness do indivíduo e adaptados através de algum mecanismo ou regra específica. Além da similaridade quanto à utilização de taxas adaptativas de mutação, o algoritmo CLONALG possui um mecanismo de seleção inspirado no princípio de seleção clonal do sistema imunológico biológico que se assemelha à seleção (μ,λ) das estratégias evolutivas. Na seleção (μ,λ), λ descendentes são gerados para cada pai da população atual através de mutação e possivelmente recombinação e os μ melhores descendentes são selecionados. Outra diferença entre o algoritmo CLONALG e as EEs, é que o CLONALG trabalha com uma população de indivíduos codificados. Finalmente, as EEs foram originalmente propostas para solucionar problemas de otimização, enquanto o algoritmo CLONALG foi desenvolvido com o objetivo de implementar computacionalmente o princípio imunológico da seleção clonal para resolver problemas de reconhecimento de padrões e, posteriormente, adaptado a problemas de otimização, enfocando o caso multimodal.

Por fim, é possível caracterizar o algoritmo de aprendizagem CLONALG como uma abordagem competitiva (Seção 4.2.1.2.2), uma vez que os anticorpos estarão competindo pelo reconhecimento antigênico ou por uma maior afinidade.

5.3.4. Complexidade Computacional

O algoritmo de seleção clonal proposto, CLONALG, possui três etapas principais de processamento: 1) a determinação da afinidade (ou fitness) dos anticorpos da população (Passos 2 e 6), 2) a seleção (e re-seleção) dos n melhores indivíduos (Passos 3 e 7), e 3) a mutação da população **C** de clones (Passo 5).

O custo computacional para a determinação da afinidade da população depende do problema abordado, e nenhuma medida genérica pode ser apresentada. Por outro lado, é possível calcular estimativas de complexidade computacional para os processos de seleção e mutação, como descrito na Seção 5.1.2. Assim, o tempo computacional envolvido no primeiro processo de seleção, Passo 3, é $O(N)$. No segundo processo de seleção, Passo 7, o custo computacional é da ordem $O(N_c)$, onde N_c corresponde à quantidade de clones gerados a partir de $\mathbf{Ab}_{\{n\}}$. Durante a mutação dos N_c clones, é necessário um tempo de processamento da ordem de $O(N_c.L)$, onde L é o comprimento da cadeia de atributos utilizada para representar os anticorpos. Somando-se a complexidade de cada etapa é possível determinar a complexidade computacional do algoritmo, como apresentado na Tabela 5.2.

Na versão desenvolvida para resolver problemas de reconhecimento de padrões, a complexidade computacional por geração do algoritmo aparece multiplicada pelo tamanho M da população de antígenos. Por outro lado, se o algoritmo está sendo aplicado a problemas de otimização multimodal com $n = N$, os processos de seleção podem ser suprimidos do algoritmo, reduzindo o custo computacional total do algoritmo para $O(N_c.L)$

A quantidade de memória necessária para executar o algoritmo é proporcional à dimensão do vetor de afinidades $\boldsymbol{f}$ ($\boldsymbol{f} \in \Re^N$), somada à dimensão da matriz $\mathbf{Ab}_{\{n\}}$ ($\mathbf{Ab}_{\{n\}} \in S^{n \times L}$), mais a dimensão da matriz **C** de clones ($\mathbf{C} \in S^{Nc \times L}$), mais a dimensão da matriz total de anticorpos **Ab** ($\mathbf{Ab} \in S^{N \times L}$). No caso de reconhecimento de padrões, a memória requerida para executar o algoritmo aparece multiplicada pelos M antígenos. O resultado desta soma está apresentado na Tabela 5.2.

Tabela 5.2. Complexidade computacional por geração para execução do algoritmo CLONALG, onde N é o tamanho da população de anticorpos, N_c a quantidade de clones gerada a partir da seleção dos n melhores indivíduos de **Ab**, L é o comprimento das cadeias de atributos, e M a quantidade de antígenos a serem reconhecidos.

	Tempo de Processamento	***Memória***
Reconhecimento de padrões	$O(M(N + N_c.L))$	$\propto M(N + L(n + N_c + N))$
Otimização	$O(N + N_c.L)$	$\propto N + L(n + N_c + N)$
Otimização multimodal	$O(N_c.L)$	

5.4. ABNET: Uma Rede de Anticorpos

Nesta seção será proposta uma rede neural Booleana, denominada ABNET (*AntiBody NETwork*), ilustrando como diversos princípios imunológicos podem ser empregados no desenvolvimento de arquiteturas e algoritmos de treinamento para redes neurais artificiais. As principais características da ABNET englobam: aprendizagem competitiva, geração automática da estrutura da rede e representação binária (Booleana) do conjunto de pesos. Este modelo foi originalmente proposto por de Castro & Von Zuben (2001f) destacando os aspectos cognitivos do sistema imunológico como discutido na Seção 4.3.1, ou seja, os problemas de aprendizagem, reconhecimento de padrões e memória imunológica.

5.4.1. Motivação e Escopo de Aplicações

Até recentemente, a determinação adequada da arquitetura de uma rede neural artificial para um dado problema envolvia a experiência do projetista na implementação de procedimentos de tentativa e erro (de Castro *et al.*, 1999a). Atualmente, o projeto automático de arquiteturas de RNAs começa a fazer parte do processo de treinamento da rede, através de uma melhor exploração dos padrões de entrada.

O objetivo da ABNET é explorar qualitativamente o algoritmo de seleção clonal através da definição automática de um repertório binário de anticorpos, representado em um espaço de formas de Hamming, e modelado utilizando uma arquitetura de rede neural artificial. A arquitetura de rede empregada é do tipo feedforward com uma única camada (Seção 4.2.1.1.1), onde os neurônios de saída da rede e seus respectivos vetores de pesos associados representam os anticorpos do sistema imunológico, e o reconhecimento dos antígenos é feito por toda a rede, e não por anticorpos individuais. O princípio da seleção clonal será empregado para a determinação da arquitetura da rede, controlando quais unidades deverão ser expandidas e quais eliminadas. O processo de maturação de afinidade será responsável pela adaptação, ou aprendizagem, da rede. Este modelo, assim como as redes neurais auto-organizadas, foi desenvolvido para aplicações genéricas a problemas de reconhecimento de padrões (particularmente binários) e clusterização.

5.4.2. Descrição e Análise da ABNET

Assim como no caso do algoritmo CLONALG para o problema de reconhecimento de padrões, será suposta a existência de uma população **Ag** de antígenos a ser reconhecida por um repertório **Ab** de anticorpos. A população antigênica corresponde a um grupo de padrões a serem reconhecidos e clusterizados pela rede. É suposto também que o repertório de anticorpos possui um único anticorpo no início do processo de aprendizagem, de forma que o repertório tem que ser construído enquanto submetido à população **Ag** (entretanto, a rede pode ser inicializada com qualquer tamanho não-nulo pré-definido). O repertório de anticorpos será modelado sob a forma de uma rede neural artificial Booleana, chamada ABNET. As principais características da ABNET são:

- Arquitetura construtiva baseada no princípio da seleção clonal;
- Poda da rede, representando a morte dos anticorpos pouco estimulados;

- Pesos Booleanos das conexões (espaço de formas binário de Hamming); e
- Aprendizagem competitiva e não-supervisionada simulando o processo de hipermutação somática direcionada.

O objetivo é construir um repertório de anticorpos capaz de apresentar máxima cobertura de um espaço conhecido de antígenos. Como a rede possui pesos binários, assumindo apenas os valores {0,1}, o espaço de formas binário de Hamming é o mais adequado para a derivação da ABNET.

Seja a seguinte notação:

- $\boldsymbol{Ab}_i$: i-ésimo anticorpo do repertório **Ab**. Cada anticorpo corresponde ao vetor de pesos $\boldsymbol{w}_i$ ligando a unidade de saída i a todas as entradas da rede;
- $\boldsymbol{Ag}_j$: j-ésimo antígeno da população **Ag** (Ag $\in S^{M \times L}$);
- τ_i: nível de concentração de cada anticorpo, ou seja, a quantidade de antígenos sendo reconhecidos por cada anticorpo $\boldsymbol{Ab}_i$;
- $\boldsymbol{f}$: vetor contendo as afinidades de todos os anticorpos da rede em relação ao antígeno $\boldsymbol{Ag}_j$ apresentado; e
- $\boldsymbol{v}$: vetor que rotula qual anticorpo i reconhece cada antígeno j.

Sob o ponto de vista das redes neurais artificiais, os padrões binários a serem reconhecidos são geralmente denominados de *padrões de treinamento* (ou de *entrada*), e as unidades que compõem a rede são chamadas *neurônios* (Seção 4.2). Na engenharia imunológica, os padrões de entrada correspondem aos antígenos, e os neurônios, juntamente com seus pesos associados, são denominados de anticorpos. Um anticorpo i ($\boldsymbol{Ab}_i$) será representado pelo vetor de pesos $\boldsymbol{w}_i$ conectando as entradas da rede à unidade de saída i. A única função das unidades de saída da rede é efetuar a soma ponderada dos sinais de entrada pelos respectivos vetores de pesos, o que corresponde a uma combinação linear entre as entradas e seus pesos associados. Uma característica particular da ABNET é a representação binária dos pesos das conexões no espaço de formas de Hamming. Sem perda de generalidade, a arquitetura inicial é composta por uma única unidade de saída, cujo vetor de pesos representa um único anticorpo hipotético na rede de anticorpos. Outro aspecto importante deste modelo, que não é considerado pelo algoritmo CLONALG, é a existência de um nível de concentração τ_i, que determina a concentração de antígenos para cada anticorpo i da rede. O nível de concentração τ_i, juntamente com a afinidade antigênica $f_{i,j}$, irá determinar qual anticorpo será selecionado para se reproduzir.

Como os antígenos a serem reconhecidos e os anticorpos da rede estão representados em um espaço de formas binário de Hamming, utilizou-se a Equação 3.3 para medir a afinidade entre estas moléculas. Por conveniência, esta equação será repetida a seguir:

$$f_{i,j} = f(\boldsymbol{Ab}_i, \boldsymbol{Ag}_j) = \sum_{m=1}^{L} \delta_m, \text{onde} \quad \delta_m = \begin{cases} 1 & \text{se } Ab_{i,m} \neq Ag_{j,m} \\ 0 & \text{outros casos} \end{cases}. \qquad \textbf{(5.12)}$$

Um anticorpo k do repertório **Ab**, com maior afinidade ao antígeno $\boldsymbol{Ag}_j$ dado, é aquele neurônio da rede cujo vetor de pesos $\boldsymbol{w}_k$ apresenta a maior distância de Hamming (Equação (5.12)) a este antígeno (reconhecimento por complementaridade):

$$k = \arg\max_i \left\| \boldsymbol{Ag}_j - \boldsymbol{Ab}_i \right\|. \quad \textbf{(5.13)}$$

A construção deste modelo requer um último parâmetro a ser definido: $\boldsymbol{v}$ que representa um vetor para rotular os anticorpos de maior afinidade. Por exemplo, se $\boldsymbol{Ab}_7$ é o anticorpo de maior afinidade a $\boldsymbol{Ag}_9$, então $v_9 = 7$.

O laço principal do algoritmo de treinamento da ABNET pode ser resumido como a seguir (Figura 5.8):

1. Escolha aleatoriamente um antígeno (padrão de entrada) $\boldsymbol{Ag}_j$, de acordo com sua densidade de probabilidade na população de antígenos;
2. Determine a afinidade $\boldsymbol{f}_j$ de todos os anticorpos da rede ao antígeno $\boldsymbol{Ag}_j$, de acordo com a Equação (5.12);
3. Determine o anticorpo $\boldsymbol{Ab}_k^j$ de maior afinidade a este antígeno $\boldsymbol{Ag}_j$ de acordo com a Equação (5.13);
4. Submeta o vetor de pesos $\boldsymbol{w}_k$, correspondente ao anticorpo $\boldsymbol{Ab}_k{}^j$, ao processo de maturação gerando $\boldsymbol{Ab}_k{}^{j*}$. Este processo é equivalente à atualização de pesos das redes neurais artificias e será descrito separadamente na Seção 5.4.2.3;
5. Incremente o nível de concentração τ_i do anticorpo selecionado ($i = k$);
6. Atribua $v_j = k$; e
7. Identifique o anticorpo mais estimulado para ser clonado (Equação (5.14), Seção 5.4.2.1) e o menos estimulado para ser um candidato a ser eliminado.

O processo de maturação de afinidade na resposta imune adaptativa e a atualização de pesos das redes neurais artificiais servem ao mesmo propósito: aumentar a qualidade da resposta do sistema ao padrão a ser reconhecido. Nosso objetivo é construir uma rede de anticorpos com dimensão mínima que seja função da distribuição de probabilidade P(**Ag**) da população de antígenos e de um limiar de afinidade ε a ser descrito posteriormente. Como P(**Ag**) não é conhecida a priori, o nível de concentração de antígenos τ_j é utilizado para estimar seu valor.

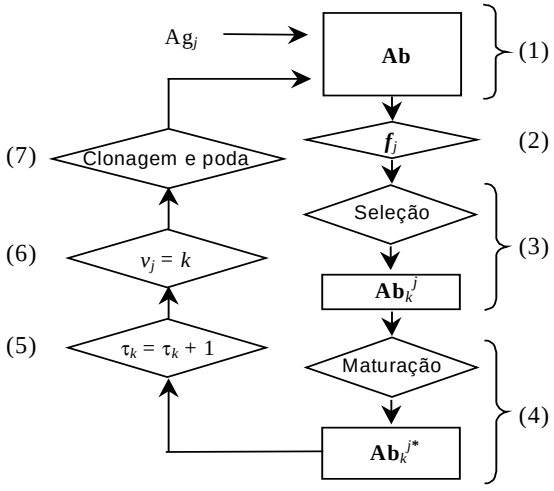


Figura 5.8. Principais passos do algoritmo de treinamento da ABNET.

5.4.2.1. Construção da Rede (Clonagem)

Este processo simula os mecanismos de seleção e reprodução dos anticorpos mais estimulados, de acordo com o princípio da seleção clonal. No nosso modelo, o anticorpo mais estimulado é aquele que possui a maior afinidade $f_{k,j}$ ao antígeno $\boldsymbol{Ag}_j$ e cujo nível de concentração antigênica τ_k é o mais alto. Como o objetivo é gerar uma rede parcimoniosa, cada anticorpo selecionado irá sofrer uma única mitose, e os dois anticorpos gerados poderão fazer parte da memória imunológica após suas respectivas maturações.

Assim, o processo de construção (ou crescimento) da rede é baseado em dois parâmetros: o nível de concentração τ_i e o limiar de afinidade ε. Todos os anticorpos $\boldsymbol{Ab}_i$ cujo nível de concentração for maior do que 1, $\tau_i > 1$, constituem candidatos em potencial a serem clonados, e o anticorpo *s* que estiver submetido à maior concentração antigênica será escolhido como único candidato. Este processo é descrito na Equação (5.14), e se nenhum anticorpo *i* satisfizer este critério, então a arquitetura da rede permanece inalterada:

$$s = \arg\max_{i \in O} \boldsymbol{Ab}_i, \text{ onde } O = \{i \mid \tau_i > 1\}. \qquad \textbf{(5.14)}$$

Se a afinidade do anticorpo *s* em relação ao antígeno que possui maior afinidade a ele for maior do que o limiar de afinidade ε, então *s* é considerado o anticorpo mais estimulado e é clonado; senão a arquitetura da rede permanece inalterada. Quanto mais antígenos um anticorpo reconhece, mais estimulado este anticorpo estará. A rede promove o reforço dos anticorpos que são muito estimulados, favorecendo sua clonagen. O processo de crescimento é verificado a cada β iterações. A idéia de efetuar o crescimento a cada β iterações é estabelecer um *intervalo de amadurecimento* (aprendizagem) para que o repertório de anticorpos tenha tempo de aumentar sua afinidade aos antígenos através do processo de maturação que ocorre a cada iteração do algoritmo, como será descrito na próxima seção.

O vetor de pesos do novo clone, ou seja, o novo anticorpo, é o complemento exato do antígeno que possui a menor afinidade ao anticorpo clonado. Este aspecto não está em perfeito acordo com o princípio da seleção clonal, mas ele permite a geração de uma rede (ou repertório) de anticorpos que apresente uma maior afinidade a todos os antígenos dados, e o processo de crescimento descrito guia a ABNET direto a esta configuração, reduzindo a quantidade de iterações necessárias para convergência.

Para que clusters de anticorpos similares sejam gerados automaticamente, o novo clone deve ser inserido na vizinhança do anticorpo clonado. Esta idéia simula a preservação de vizinhança, considerada pelos mapas auto-organizáveis de Kohonen descritos na Seção 4.2.

Como ilustração da especificação dos parâmetros τ e *v* e do funcionamento do mecanismo de crescimento da ABNET, considere a existência da seguinte população de antígenos a serem reconhecidos: $\boldsymbol{Ag}_1$=[0,0,0], $\boldsymbol{Ag}_2$=[1,1,1], $\boldsymbol{Ag}_3$=[1,0,0]. A configuração inicial da rede contém um único anticorpo representado pelo vetor de pesos $\boldsymbol{w}$=[1,1,1]. Como existe um único anticorpo na rede, os elementos v_j, j = 1,...,3, do vetor de rótulos $\boldsymbol{v}$, contêm o mesmo valor 1, ou seja, o anticorpo $\boldsymbol{Ab}_1$ reconhece todos os antígenos (Figura 5.9(a)). Isto ocorre porque, sendo o único neurônio, ele é aquele que possui a maior afinidade a todos os antígenos.

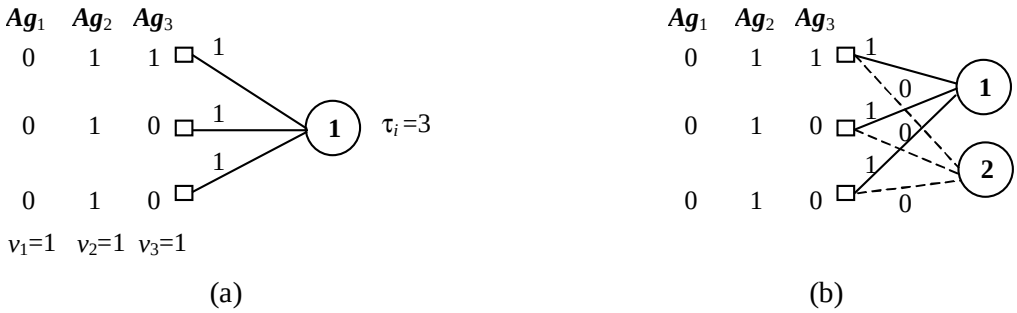


Figura 5.9. Processo de construção da rede. (a) Especificação dos parâmetros τ_i e v_j, onde τ_i define o nível de concentração de cada anticorpo e v_j rotula os anticorpos com maiores afinidade a cada um dos antígenos dados. $\boldsymbol{Ag}_j$, j = 1,...,3, são os antígenos (padrões de entrada ou treinamento); $\boldsymbol{w}_1$ = [1,1,1] representa $\boldsymbol{Ab}_1$. (b) Clonagem e definição do vetor de pesos correspondente ao novo anticorpo, $\boldsymbol{w}_2$ = [0,0,0] representa $\boldsymbol{Ab}_2$.

O antígeno com maior afinidade (maior distância de Hamming) em relação a este anticorpo é $\boldsymbol{Ag}_1$, e o antígeno de menor afinidade é $\boldsymbol{Ag}_2$ (Equação (5.12)). Este anticorpo $\boldsymbol{Ab}_1$ será selecionado para se reproduzir e seu clone terá pesos complementares a $\boldsymbol{Ag}_2$, $\boldsymbol{w}_2$=[0,0,0]. A Figura 5.9(b) ilustra a geração de um novo anticorpo e a definição de seu vetor de pesos associados. Este procedimento garante uma maior e mais rápida cobertura do espaço de antígenos.

5.4.2.2. Poda da Rede (Morte Celular Programada)

A política de poda, simulando a apoptose (morte programada dos anticorpos pouco estimulados), dá-se como a seguir: se um anticorpo p possui seu nível de concentração igual a zero ($\tau_p = 0$) por uma quantidade de iterações maior do que um limiar pré-definido, então este anticorpo pode ser removido da rede. Como proposto por de Castro & Von Zuben (1999b) em um outro contexto, após executar uma poda em uma rede competitiva, o valor da taxa de aprendizagem, correspondente, neste caso, à taxa de hipermutação a ser descrita na próxima seção, deve ser reinicializada. No treinamento da ABNET, a reinicialização da taxa de hipermutação foi empregada durante os processos de poda e clonagem da rede.

Para ilustrar como ocorre a poda, considere o mesmo exemplo anterior: uma população de antígenos composta por $\boldsymbol{Ag}_1$=[0,0,0], $\boldsymbol{Ag}_2$=[1,1,1] e $\boldsymbol{Ag}_3$=[1,0,0]. Suponha que uma ABNET com três anticorpos ($\boldsymbol{w}_1$=[1,1,1], $\boldsymbol{w}_2$=[0,0,0], $\boldsymbol{w}_3$=[1,0,0]) foi construída de acordo com a Figura 5.10(a). Neste caso, os vetores $\boldsymbol{v}$ = [1,2,1] e τ = [2,1,0], indicam que o anticorpo $\boldsymbol{Ab}_3$, não estimulado por nenhum antígeno, será removido (podado) da rede após uma quantidade de gerações celulares pré-especificada. Após remover $\boldsymbol{Ab}_3$, a rede resultante está apresentada na Figura 5.10(b). Esta política de poda objetiva manter a especificidade dos anticorpos, devido ao fato de que somente aqueles anticorpos que não reconhecem nenhum antígeno serão candidatos a serem removidos.

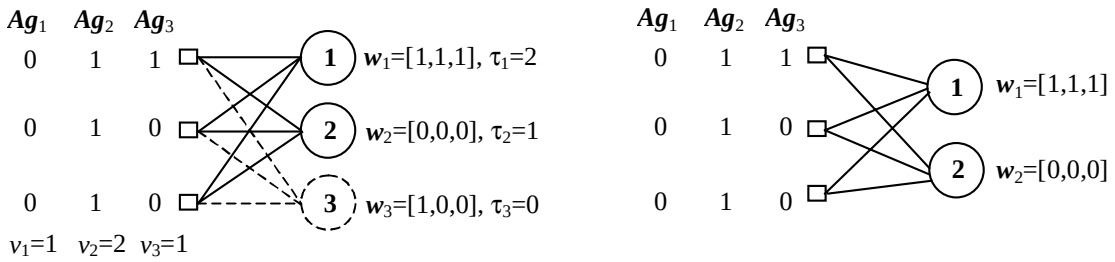


Figura 5.10. Poda de anticorpos não estimulados. (a) O anticorpo $\boldsymbol{Ab}_3$, não está sendo estimulado por nenhum antígeno ($\tau_3 = 0$). $\boldsymbol{Ag}_j$, j = 1,...,3, são os antígenos (padrões de entrada); $\boldsymbol{w}_1$ = [1,1,1], $\boldsymbol{w}_2$ = [0,0,0] e $\boldsymbol{w}_3$ = [0,0,0] representam $\boldsymbol{Ab}_1$, $\boldsymbol{Ab}_2$ e $\boldsymbol{Ab}_3$, respectivamente, e, $\varepsilon = 1$. (b) Rede resultante com apenas dois anticorpos, após um período de tempo maior do que um limiar pré-definido.

A Figura 5.11 ilustra um processo hipotético de construção da ABNET. As circunferências sólidas representam os anticorpos restantes após crescimento e poda, as circunferências em branco representam os anticorpos clonados e as circunferências sombreadas correspondem aos anticorpos que foram removidos durante o processo de aprendizagem/crescimento da rede. As unidades de entrada e suas conexões são representadas apenas na rede inicial (a) e na final (c).

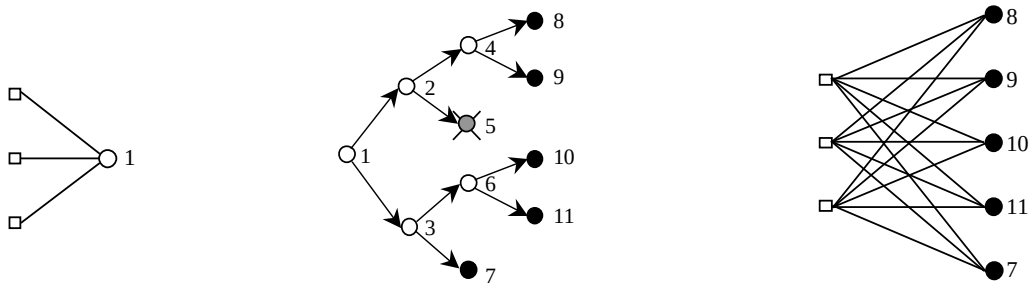


Figura 5.11. Construção da rede. Quadrados: unidades de entrada; circunferências sólidas: anticorpos restantes; circunferências em branco: anticorpos clonados; circunferências sombreadas: anticorpos eliminados. (a) ABNET inicial. (b) Processos de crescimento e poda da rede (as conexões são omitidas e as setas indicam os processos de clonagem). (c) Arquitetura resultante da rede com os anticorpos restantes rotulados.

5.4.2.3. Atualização de Pesos (Maturação de Afinidade)

O processo de atualização de pesos da ABNET é semelhante a uma mutação de múltiplos pontos com taxas variáveis (Seções 3.4.3.4 e 4.4.3). A taxa de hipermutação α irá determinar quantas posições da cadeia de atributos, que representa os anticorpos, serão mutadas.

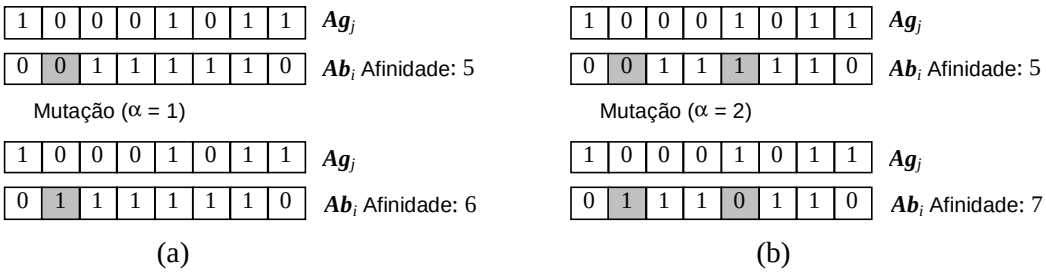


Figura 5.12. Processo de atualização de pesos para antígenos de comprimento $L = 8$, onde as posições de mutação são escolhidas aleatoriamente dentre as complementares (a) $\alpha = 1$. (b) $\alpha = 2$.

Como o espaço de formas utilizado é o binário de Hamming, as posições do anticorpo que não forem complementares ao antígeno reconhecido serão candidatas a sofrerem mutação, resultando em um processo direcionado de busca no espaço de formas. O processo de atualização de pesos, correspondente à maturação de afinidade, é direcionado para que os anticorpos se tornem o complemento do conjunto de antígenos mais rapidamente (Figura 5.12). Quanto maior a distância de Hamming (complementaridade) entre um antígeno e um anticorpo, menor a taxa de mutação e vice-versa. No caso do espaço de formas binário de Hamming, a taxa de mutação $\alpha \in Z^{+}_{\{0\}}$, onde $Z^{+}_{\{0\}}$ é o conjunto de inteiros não-negativos, possui um esquema de decrescimento em unidades, e após uma quantidade pré-definida de iterações, o valor de α é reduzido de 1, até que ele atinja $\alpha = 0$, representando a ausência de mutação. Isso geralmente ocorre quando a afinidade Ag-Ab é máxima.

5.4.3. Pseudocódigo

Considere a implementação de uma função computacional para simular a ABNET. Esta função pode ter como argumentos os seguintes parâmetros:

Entrada:

- Comprimento L das cadeias de atributos;
- População **Ag** composta por M antígenos de comprimento L;
- Quantidade *gen* de gerações a serem executadas (critério de parada);
- Limiar de afinidade ε;
- Taxa inicial α_0 de hipermutação; e
- A cada β iterações a taxa α de hipermutação é decrescida de uma unidade e um anticorpo é escolhido para ser um candidato à clonagem. Se um anticorpo p não é estimulado por nenhum antígeno durante β iterações, então ele é eliminado da rede.

Saída:

- Matriz **Ab** de anticorpos de memória; e
- Matriz **F** contendo a afinidade de cada anticorpo $\boldsymbol{Ab}_i$, $i = 1,...N$ (o valor de N é automaticamente definido ao longo da execução do algoritmo), em relação a cada antígeno $\boldsymbol{Ag}_j$, $j = 1,...M$.

Dados os parâmetros de entrada e saída, é possível descrever um pseudocódigo para a implementação da ABNET como a seguir:

```
function [Ab,F] = abnet(Ag,L,gen,ε,α0,β);
Ab := gera(1,L);                          % Gera Ab inicial
% A cada geração, faça:
for t = 1 to gen,
   for j = 1 to M,                        % Passo 1
      f(j,:)   := afinidade(Ab,Ag(j,:));     % Passo 2
      Abk      := select(Ab,f,1);            % Passo 3
      Abk*     := dmut(Abk,Ag(j,:),α);       % Passo 4
      τk       := τk + 1;                    % Passo 5
      vj       := k;                         % Passo 6
   end;
   if (t mod β) = 0,
      [Ab,α] := split(Ab,v,τ,ε,α0);      % Passo 7
      α      := α - 1;
   end;
   [Ab,α] := poda(Ab,τ,t,β,α0);          % Passo 7
   F      := afinidade(Ab,Ag);
end;
```

5.4.4. Caracterização e Comparação

Nesta seção, foi proposto um algoritmo construtivo para a geração de uma rede neural Booleana baseado nas propriedades do princípio de seleção clonal e maturação de afinidade do sistema imunológico. O repertório de anticorpos é modelado utilizando-se uma abordagem conexionista (rede neural artificial), com uma rede de anticorpos sendo gerada, onde os vetores de pesos da rede representam os anticorpos. Várias características da resposta imune adaptativa foram consideradas, cada uma com uma finalidade distinta:

- Expansão clonal dos anticorpos mais estimulados: crescimento da rede;
- Morte programada dos anticorpos pouco estimulados: poda da rede; e
- Maturação de afinidade: atualização dos pesos (aprendizagem).

O emprego de um processo de restauração e controle da taxa de hipermutação, juntamente com a definição do intervalo de amadurecimento, devem simular os mecanismos propostos por Kepler & Perelson (1993a,b), onde intervalos de expansão clonal e ausência de mutação são intercalados com a ocorrência de altas taxas de mutação e nenhuma expansão. Estes procedimentos devem conduzir a uma estratégia que visa a otimização do controle dos mecanismos de expansão clonal e maturação de afinidade (Kepler & Perelson, 1993a).

As arquiteturas de rede modeladas não possuem um tamanho pré-definido. Os antígenos a serem reconhecidos serão responsáveis pela seleção e expansão dos anticorpos disponíveis. Durante a evolução da rede, ela desenvolve uma série de características auto-regulatórias como a determinação automática de um tamanho específico. Uma vez que a ABNET atinge um equilíbrio em relação à quantidade de unidades na rede, os anticorpos clonados devem competir com os já estabelecidos para que eles possam entrar na rede, ou seja, fazer parte do repertório.

Sob o ponto de vista de imunologia, a tarefa desempenhada pela ABNET é equivalente à geração de um repertório de anticorpos com tamanho mínimo para reconhecer qualquer antígeno de uma população específica e conhecida. Sob o ponto de vista de sistemas inteligentes, o algoritmo apresenta uma estratégia competitiva (não supervisionada) para a definição de arquitetura e treinamento de uma rede neural artificial Booleana. Como no caso das RNAs, a utilização de um limiar de afinidade ε permite o estudo da capacidade de generalização, ou resposta reativa cruzada, da ABNET.

O mecanismo de seleção utilizado pela ABNET pode ser visto como o dual do esquema de competição aplicado a redes neurais competitivas, onde, no primeiro caso, o neurônio (anticorpo) escolhido é o que possui a maior afinidade (maior distância de Hamming) ao antígeno e, no segundo caso, o neurônio escolhido é o que possui menor distância ao padrão de entrada. Além disso, a ABNET é capaz de controlar a dimensão de sua arquitetura como algumas classes de redes neurais auto-organizadas (Fritzke, 1994; Cho, 1997; de Castro & Von Zuben, 1999b), e também possui uma taxa de aprendizagem variável, como no caso dos mapas auto-organizáveis de Kohonen (Seção 4.2.2.2).

A rede de Hamming (Lippman, 1987) se assemelha à ABNET no sentido de que ambas são competitivas e calculam a distância de Hamming entre um conjunto de pesos e os padrões de entrada, mas diferenciam-se completamente quanto ao algoritmo de treinamento. Na rede de Hamming, a arquitetura é fixa e contém duas camadas, os pesos são especificados (e não treinados) de acordo com o conjunto de treinamento e as unidades da camada de saída possuem um bias. Por outro lado, a ABNET é construída durante a aprendizagem, possui uma única camada e não apresenta bias, resultando em uma arquitetura construtiva e parcimoniosa. Ambas as redes são projetadas para tratar conjuntos de dados binários de forma eficiente e são interessantes para a implementação em hardware, principalmente devido às suas operações intrinsicamente binárias. Exemplos de implementação em hardware de redes de Hamming que talvez possam servir de base para o caso da ABNET podem ser encontrados nos trabalhos de Robinson *et al.* (1992), Çilingiroglu (1993) e Schimid *et al.* (1998).

A classe de redes neurais em que os vetores de entrada e saída são strings binárias (0 ou 1) é geralmente denominada de Booleana. Neste grupo de redes neurais, a representação obtida pelo algoritmo de treinamento torna-se um circuito lógico implementável por *portas lógicas* simulando unidades intermediárias e de saída. Exemplos de algoritmos derivados de princípios de álgebra Booleana podem ser encontrados em Biswas & Kumar (1990) e Gray & Michel (1992). Neste caso, o algoritmo de aprendizagem é completamente diferente do empregado pela ABNET, e utiliza operações de álgebra Booleana ao invés da inspiração biológica.

5.4.5. Complexidade Computacional

O algoritmo de treinamento da ABNET, possui três etapas principais de processamento: 1) a determinação da afinidade (ou fitness) dos anticorpos da população (Passo 2); 2) a seleção do melhor indivíduo (Passo 3); e 3) a mutação do anticorpo selecionado (Passo 4).

Tabela 5.3. Complexidade computacional por geração para o treinamento da ABNET, onde N é o tamanho atual da população de anticorpos (dimensão da ABNET), L é o comprimento das cadeias de atributos que representam as moléculas, e M é a quantidade de antígenos a serem reconhecidos.

	Tempo de Processamento	***Memória***
ABNET	$O(M(L + \log N))$	$\propto N(L + M)$

O custo computacional para a determinação da afinidade da população depende do problema abordado e nenhuma medida genérica pode ser apresentada. Por outro lado, é possível calcular estimativas de complexidade computacional para o processo de seleção e mutação, como descrito na Seção 5.1.2.

Uma particularidade do algoritmo de treinamento da ABNET é o seu processo construtivo. A quantidade de anticorpos do repertório varia dinamicamente: o processo é iniciado com apenas uma unidade na rede e vai aumentando até que a rede atinja uma dimensão de equilíbrio. Portanto, o custo computacional do algoritmo também varia dinamicamente até que a arquitetura da rede atinja este estado de equilíbrio, começando com um baixo custo computacional e aumentando à medida que novas unidades vão sendo inseridas. Na situação final da rede, em que uma quantidade de anticorpos foi inserida e ela atingiu uma arquitetura com dimensão estável, considere que a rede possui N anticorpos (unidades de saída). Assim, o custo computacional envolvido no único processo de seleção, Passo 3, é $O(\log N)$, pois deseja-se encontrar apenas o anticorpo de maior afinidade. Durante a mutação direcionada deste anticorpo, é necessário um tempo de processamento da ordem de $O(L)$, onde L é o comprimento da cadeia de atributos utilizada para representar os anticorpos. Estes passos são executados para os M antígenos apresentados. Somando-se a complexidade de cada etapa, é possível determinar a complexidade computacional do algoritmo, como apresentado na Tabela 5.3.

A cada iteração, a quantidade de memória requerida para executar o algoritmo é proporcional ao número atual N de anticorpos na rede. A matriz **Ab** de anticorpos tem dimensão $S^{N \times L}$, e a matriz **F** de afinidades é de dimensão $S^{N \times M}$, assim a memória requerida para executar este algoritmo é proporcional a $N(L + M)$, como apresentado na Tabela 5.3.

5.5. aiNet: Um Modelo de Rede Imunológica Artificial

Nesta seção, é apresentado um modelo de rede imunológica artificial, chamada aiNet (*Artificial Immune NETwork*), inspirado na teoria da rede imunológica proposta por Jerne (1974a) e discutida na Seção 2.10. Este algoritmo foi originalmente proposto por de Castro & Von Zuben (2000c, 2001g) e a rede desenvolvida apresenta características importantes, como a capacidade de descrever a estrutura interna dos antígenos (dados de treinamento), seu perfil de distribuição de probabilidade e relações de vizinhança (clusters). A aiNet incorpora o algoritmo de seleção clonal, CLONALG (Seção 5.3), como parte do processo de treinamento da rede.

5.5.1. Motivação e Escopo de Aplicações

Na Seção 2.10, foi argumentado que a teoria da rede imunológica é particularmente interessante para o desenvolvimento de ferramentas computacionais, pois ela fornece uma medida aproximada de propriedades emergentes como aprendizagem e memória, tolerância ao próprio, dimensão e diversidade de populações celulares. Características da rede imunológica como estrutura, dinâmica e metadinâmica podem ser diretamente empregadas na construção de um modelo de rede imunológica artificial, denominada aiNet, para resolver problemas de engenharia. No nosso modelo de rede imunológica, o princípio da seleção clonal irá controlar a quantidade e forma dos anticorpos da rede (sua dinâmica e metadinâmica), enquanto técnicas de clusterização hierárquica e teoria de grafos serão utilizadas para definir e interpretar a estrutura final da aiNet. Assim, como no caso da ABNET, o algoritmo de treinamento é genérico, porém a rede resultante irá depender do problema tratado, ou seja, o conjunto de antígenos a serem reconhecidos guiará a busca pela estrutura final da rede e forma dos anticorpos. As principais aplicações da aiNet incluem aprendizagem de máquina, reconhecimento de padrões, compressão e clusterização de dados.

5.5.2. Descrição

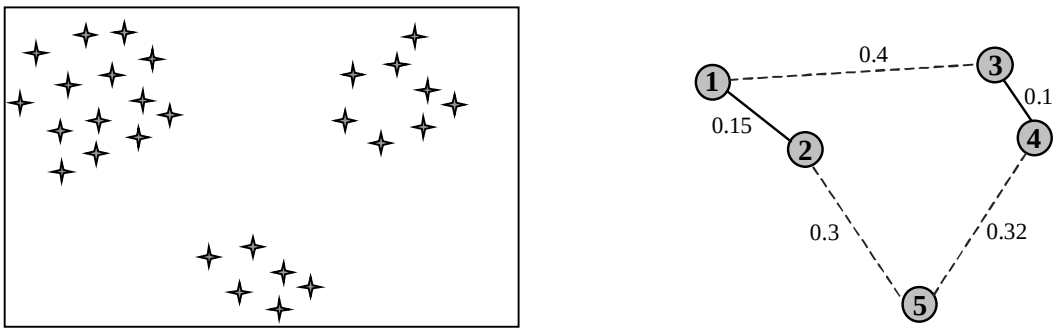
Dado um conjunto **Ag** de antígenos, onde cada antígeno (padrão ou amostra de treinamento) $\boldsymbol{Ag}_j$, $j = 1,...,M$, é descrito por L variáveis (atributos ou características) em um espaço de formas Euclidiano, uma rede imunológica artificial deve ser construída para responder às seguintes questões: (1) Existe algum grupo ou subgrupo intrínseco aos antígenos? (2) Se existir(em), quantos são? (3) Quais são as propriedades relevantes destes grupos de antígenos? (4) Como podemos gerar regras de decisão para classificar novos antígenos? (5) Qual é a conformação dos grupos no espaço de formas?

O modelo de rede imunológica artificial, aiNet, pode ser formalmente definido como abaixo:

Definição 5.1: A *rede imunológica artificial*, chamada aiNet, é um *grafo com conexões ponderadas*, não necessariamente totalmente interconectado, composto por um conjunto de nós, denominados *anticorpos*, e conjuntos de pares de nós chamados *conexões*, com um valor característico associado, chamado de *peso da conexão* ou simplesmente *peso*.

Como a estrutura da rede poderá ser determinada por alguns métodos de clusterização, iremos definir primeiramente o que significa um cluster e clusterização de dados:

Definição 5.2: *Clusterização* é um processo de agrupamento de dados que apresentam um elenco de propriedades similares e distintas daquelas presentes em outros dados não pertencentes ao mesmo agrupamento.



(a) (b)

Figura 5.13. Ilustração da aiNet. (a) Conjunto de antígenos a serem reconhecidos e clusterizados, contendo três regiões de alta concentração antigênica. (b) aiNet com os anticorpos rotulados e os respectivos pesos das conexões. As linhas tracejadas indicam conexões que deverão ser detectadas e removidas com o objetivo de gerar subgrafos desconexos, cada um caracterizando um cluster diferente da rede.

Os clusters da rede servirão como *imagens internas* que mapeiam clusters existentes no conjunto de antígenos em clusters existentes nos anticorpos que formam a aiNet. Como ilustração, suponha que existe um conjunto de antígenos composto por três regiões de alta concentração antigênica (densidade de dados), de acordo com a Figura 5.13(a). Uma arquitetura hipotética de rede gerada pelo algoritmo de treinamento da aiNet é apresentada na Figura 5.13(b). O número dentro das unidades indicam seus rótulos, os números ao lado das conexões representam seus pesos associados, e as linhas tracejadas sugerem conexões a serem eliminadas objetivando detectar clusters e definir a arquitetura final da rede. Note a presença de três clusters distintos de anticorpos, cada qual com um número diferente de moléculas, conexões e pesos associados. Estes clusters constituem imagens internas dos clusters correspondentes do conjunto de antígenos. Note também que a quantidade de anticorpos na rede é bem menor do que a quantidade de antígenos a serem reconhecidos, indicando uma redução na cardinalidade da informação a ser armazenada.

O algoritmo de treinamento da aiNet possui duas etapas distintas. Na primeira delas, é feita a interação dos anticorpos da rede e um antígeno a ser reconhecido, e na segunda etapa é quantificada a interação apenas dos anticorpos da rede. As afinidades Ag-Ab e Ab-Ab são avaliadas utilizando-se medidas de proximidade (ou similaridade). O objetivo é utilizar uma métrica de distância para gerar um conjunto de anticorpos que constituam imagens internas dos antígenos a serem reconhecidos, e avaliar o grau de similaridade entre os anticorpos da rede, de forma a permitir um controle da cardinalidade do repertório. Sendo assim, a afinidade Ag-Ab é inversamente proporcional à distância entre eles: quanto menor a distância, maior a afinidade e vice-versa. Enquanto uma alta afinidade Ag-Ab resulta numa resposta imune adaptativa seguindo o algoritmo CLONALG (princípio da seleção clonal), uma alta afinidade entre anticorpos promove uma supressão na rede (eliminação de anticorpos).

É importante salientar que, na estrutura formal para a engenharia imunológica proposta no Capítulo 3, assim como nos sistemas imunológicos biológicos, o reconhecimento

antigênico ocorre por complementaridade de formas, e não por similaridade. Entretanto, em diversas aplicações de sistemas imunológicos artificiais (por exemplo, Oprea, 1999; Hart & Ross, 1999; Hajela & Yoo, 1999), e para os objetivos propostos para a aiNet, a geração de um conjunto de anticorpos com características similares, e não complementares, aos antígenos é mais apropriada, pois permite uma interpretação direta da rede resultante. A mesma abordagem foi empregada para o algoritmo SAND, descrito na Seção 5.2.

Como proposto na teoria original da rede imunológica (Seção 2.10) e nos modelos contínuos apresentados (Seção 3.4.3.5), os anticorpos presentes na rede irão competir pelo reconhecimento antigênico e aqueles que forem bem sucedidos promoverão a ativação da rede e proliferação celular seguindo o princípio da seleção clonal, enquanto aqueles que falharem no reconhecimento serão eliminados. Além disso, quando um anticorpo reconhecer um antígeno próprio, ou seja, outro anticorpo, isso resultará em uma supressão da rede. Na aiNet, a supressão é efetuada eliminando-se um entre dois anticorpos que se reconhecem (auto-reativos), dado um limiar de supressão σ_s. A interação de cada par Ag-Ab é avaliada no espaço de formas Euclidiano através de uma medida de afinidade (similaridade) *f*, que reflete a probabilidade de se iniciar uma resposta clonal. De forma semelhante, uma afinidade $s_{i,j}$ será especificada a cada par Ab-Ab, refletindo seu grau de interação.

Supondo que na iteração *t* a aiNet possui *N* anticorpos, a seguinte notação será empregada:

- **Ab**: repertório de anticorpos disponíveis ($\mathbf{Ab} \in S^{N\times L}$, $\mathbf{Ab} = \mathbf{Ab}_{\{d\}} \cup \mathbf{Ab}_{\{m\}}$);
- $\mathbf{Ab}_{\{m\}}$: repertório total de anticorpos de memória ($\mathbf{Ab}_{\{m\}} \in S^{m\times L}$, $m \leq N$);
- **Ag**: população de *M* antígenos a serem reconhecidos ($\mathbf{Ag} \in S^{M\times L}$);
- $\boldsymbol{f}_j$: vetor contendo a afinidade de todos os anticorpos em relação ao antígeno $\boldsymbol{Ag}_j$. A afinidade é inversamente proporcional à distância Ag-Ab;
- **S**: matriz de similaridade entre os anticorpos de memória com elementos $s_{i,j}$;
- **C**: população de N_c clones gerada a partir de **Ab** ($\mathbf{C} \in S^{Nc\times L}$);
- **C***: população **C** após o processo de maturação de afinidade;
- $\boldsymbol{d}_j$: vetor contendo a afinidade de todos os clones de **C*** em relação ao antígeno $\boldsymbol{Ag}_j$;
- ζ: percentual de anticorpos maduros a serem selecionados;
- **M**: clones de memória para o antígeno $\boldsymbol{Ag}_j$. Restantes do processo de supressão clonal e seleção dos ζ anticorpos maduros;
- $\mathbf{Ab}_{\{d\}}$: *d* novos anticorpos a serem inseridos na população **Ab** ($\mathbf{Ab}_{\{d\}} \in S^{d\times L}$);
- σ_d: limiar de morte natural; e
- σ_s: limiar de supressão.

O algoritmo de treinamento da aiNet tem por objetivo construir um conjunto de memórias que melhor represente a estrutura do conjunto de antígenos a serem reconhecidos. Quanto mais específicos forem os anticorpos da rede, menos parcimoniosa será a arquitetura (baixa compressão de dados), e quanto mais generalistas os anticorpos, mais parcimoniosa a arquitetura da rede em relação à quantidade de anticorpos (alta compressão de dados). As conexões entre os componentes da rede serão definidas, utilizando métodos estatísticos ou de teoria de grafos, após a aplicação do algoritmo de treinamento descrito a seguir:

A cada iteração, faça:

1.1. Para cada antígeno $\boldsymbol{Ag}_j$, $j = 1,...,M$, ($\boldsymbol{Ag}_j \in \mathbf{Ag}$), faça:

1.1.1. Determine sua afinidade $f_{i,j}$, $i = 1,...,N$, em relação a todo o repertório **Ab**. $f_{i,j} = 1/D_{i,j}$, $i = 1,...,N$:

$$D_{i,j} = \| \boldsymbol{Ab}_i - \boldsymbol{Ag}_j \|, \; i = 1,..., N \qquad \textbf{(5.15)}$$

1.1.2. Do conjunto **Ab**, um subconjunto $\mathbf{Ab}_{\{n\}}$ composto pelos n anticorpos com maiores afinidades é selecionado;

1.1.3. Os n indivíduos selecionados irão se proliferar (clonagem) proporcionalmente à sua afinidade $f_{i,j}$, gerando uma população **C** de clones: quanto maior a afinidade, maior o tamanho do clone para cada um dos n anticorpos selecionados (Equação (5.23));

1.1.4. Em seguida, a população **C** de clones é submetida ao processo de maturação de afinidade gerando uma nova população **C***, onde cada anticorpo k de **C*** irá sofrer uma mutação com taxa α_k inversamente proporcional à afinidade $f_{i,j}$ de seu anticorpo pai: quanto maior a afinidade, menor a taxa de mutação:

$$\boldsymbol{C}_k^* = \boldsymbol{C}_k + \alpha_k (\boldsymbol{Ag}_j - \boldsymbol{C}_k); \; \alpha_k \propto 1/f_{k,j}; \; k = 1,..., N_c; \; N \leq N_c. \qquad \textbf{(5.16)}$$

1.1.5. Determine a afinidade $d_{k,j} = 1/D_{k,j}$ entre os elementos de **C*** e o antígeno $\boldsymbol{Ag}_j$:

$$D_{k,j} = \| \boldsymbol{C}_k^* - \boldsymbol{Ag}_j \|, \; k = 1,..., N_c. \qquad \textbf{(5.17)}$$

1.1.6. Da população **C***, re-selecione ζ% dos anticorpos que apresentam maior $d_{k,j}$ e os coloque em uma matriz **M** de clones de memória (memória clonal);

1.1.7. *Morte programada*: Elimine todos os clones de memória cuja afinidade $d_{k,j}$ ao antígeno $\boldsymbol{Ag}_j$ é superior a um limiar σ_d:

Se $D_{k,j} > \sigma_d$, então exclua o anticorpo k do conjunto **M**. **(5.18)**

1.1.8. Determine a afinidade $s_{i,j}$ entre os clones de memória:

$$s_{i,j} = \| \boldsymbol{M}_i - \boldsymbol{M}_j \|, \forall i, j. \qquad \textbf{(5.19)}$$

1.1.9. *Supressão clonal*: elimine aqueles clones de memória cuja afinidade $s_{i,j}$ for inferior ao limiar de supressão σ_s:

Se $s_{i,j} < \sigma_s$, então exclua o anticorpo i do conjunto **M** **(5.20)**

1.1.10. Concatene a matriz total de anticorpos de memória com a matriz de clones de memória **M** para o antígeno $\boldsymbol{Ag}_j$: $\mathbf{Ab}_{\{m\}} \leftarrow [\mathbf{Ab}_{\{m\}}; \mathbf{M}]$;

1.2. Calcule a afinidade entre todos os anticorpos de memória:

$$s_{i,j} = \| \boldsymbol{Ab}^i_{\{m\}} - \boldsymbol{Ab}^j_{\{m\}} \|, \; \forall i, j. \qquad \textbf{(5.21)}$$

1.3. *Supressão da rede*: elimine os anticorpos de memória tais que $s_{i,j} < \sigma_s$:

Se $s_{i,j} < \sigma_s$, então exclua $\boldsymbol{Ab}^i_{\{m\}}$ da rede **(5.22)**

1.4. A matriz total de anticorpos corresponde à matriz de memória $\mathbf{Ab}_{\{m\}}$ concatenada com d novos anticorpos: $\mathbf{Ab} \leftarrow [\mathbf{Ab}_{\{m\}}; \mathbf{Ab}_{\{d\}}]$

2. Teste o critério de parada.

O limiar de supressão σ_s controla o nível de especificidade dos anticorpos, a acurácia da clusterização e a plasticidade da rede. É sugerido um valor inicial pequeno para este limiar (p.ex., $\sigma_s \leq 10^{-3}$) seguido por um ajuste fino de acordo com o desempenho da rede. A Seção 6.5.3 apresenta a análise de sensibilidade da aiNet em relação aos parâmetros de treinamento definidos pelo usuário, incluindo σ_s.

Para a determinação do tamanho N_c do conjunto de clones gerado para cada um dos j, $j = 1,...,M$, antígenos, poderíamos ter empregado a Equação (5.10), como proposto no caso do algoritmo CLONALG para o problema de reconhecimento de padrões. No entanto, optamos por uma outra forma de determinação de N_c, como descrito pela Equação (5.23), com a vantagem de eliminar o parâmetro β (embora ele possa ser facilmente introduzido na equação) definido pelo usuário e a desvantagem de reduzir o controle do usuário sobre N_c.

$$N_c = \sum_{i=1}^{n} round\left(N - D_{i,j}.N\right), \qquad \textbf{(5.23)}$$

onde N é a quantidade total de anticorpos do repertório **Ab**, $round(\cdot)$ é o operador que arredonda o valor entre parênteses para o inteiro mais próximo e $D_{i,j}$ é a distância entre o anticorpo selecionado i e o antígeno $\boldsymbol{Ag}_j$, dada pela Equação (5.15). É fácil verificar por esta equação que quanto maior a afinidade $\boldsymbol{Ag}_j$-$\boldsymbol{Ab}_i$, ou seja, quanto menor $D_{i,j}$, maior a quantidade de clones gerada para o anticorpo $\boldsymbol{Ab}_i$.

Devido à quantidade de operações efetuadas por este algoritmo, não será apresentado um diagrama de blocos específico para ele, mas note que do Passo 1.1 ao Passo 1.1.7 o algoritmo da aiNet corresponde ao algoritmo CLONALG para o problema de reconhecimento de padrões (Figura 5.5). Além da seleção clonal via o algoritmo CLONALG, os Passos de 1.1.8 a 1.1.10 e 1.2 a 1.3 simulam parte da atividade da rede imunológica. A afinidade entre os anticorpos e o antígeno j apresentado é aumentada de acordo com a Equação (5.16), onde o parâmetro α_k corresponde à taxa de hipermutação individual de cada anticorpo pertencente ao conjunto **C**. Esta equação executa uma busca direcionada, onde a distância entre o antígeno $\boldsymbol{Ag}_j$ e os anticorpos é diminuída proporcionalmente a α_k. Desta forma, o processo de busca otimiza localmente os anticorpos $\boldsymbol{C}_k$, $k = 1, ..., N_c$, com o objetivo de aumentar suas capacidades de reconhecimento antigênico ao longo das iterações. Este processo é característico de uma busca em que cada elemento é otimizado localmente, para que se alcance um melhor comportamento global do sistema.

De acordo com este algoritmo, uma resposta imune adaptativa (expansão clonal) ocorre após a apresentação de cada antígeno à rede. Existem também dois passos supressivos no algoritmo (Passos 1.1.9 e 1.3) denominados, respectivamente, de *supressão clonal* e *supressão da rede*. A supressão clonal é responsável por eliminar anticorpos com alta afinidade entre si dentro de cada clone de memória (anticorpos de **M**), enquanto a supressão da rede elimina anticorpos com alta afinidade considerando toda a rede ($\mathbf{Ab}_{\{m\}}$).

Após a fase de aprendizagem, os anticorpos devem representar imagens internas dos antígenos (ou grupos de antígenos) apresentados. Este algoritmo segue a estrutura genérica de rede imunológica dada pela Equação 3.17, em que a taxa de variação populacional é

diretamente proporcional à entrada de novos anticorpos (Passo 1.4), menos a morte dos anticorpos pouco estimulados (Passo 1.1.7) e auto-reativos (Passos 1.1.9 e 1.3), mais a reprodução dos anticorpos estimulados (Passo 1.1.3).

Para avaliar a convergência da aiNet, é possível propor diferentes critérios:

- Interrompa o processo iterativo após uma quantidade pré-definida de iterações;
- Interrompa o processo iterativo quando a rede atingir uma dimensão pré-definida;
- Avalie a afinidade média (correspondente ao erro médio) entre a população de anticorpos e os antígenos apresentados. Este critério é bastante preciso quando o número de anticorpos da rede é aproximadamente igual ao número de antígenos a serem reconhecidos; e
- A aiNet é dita ter convergido se o erro médio aumenta durante *k* iterações consecutivas.

5.5.3. Pseudocódigo

Considere a implementação de uma função computacional para treinar a aiNet de acordo com o algoritmo proposto. Esta função pode ter como argumentos os seguintes parâmetros:

Entrada:

- Comprimento *L* das cadeias de atributos;
- População **Ag** composta por *M* antígenos;
- Quantidade *gen* de gerações a serem executadas (critério de parada);
- Número *n* de anticorpos a serem selecionados para clonagem;
- Percentual ζ% de anticorpos maduros a serem selecionados;
- Limiar σ_d de morte natural;
- Limiar σ_s de supressão; e
- Quantidade *d* de novos anticorpos a serem inseridos na rede.

Saída:

- Matriz $\mathbf{Ab}_{\{m\}}$ de anticorpos de memória; e
- Matrix **S** de similaridade entre os anticorpos de memória da rede, composta por elementos $s_{i,j}$.

Dados os parâmetros de entrada e saída, é possível descrever um pseudocódigo para a implementação da aiNet como a seguir:

```
function [Ab{m},S] = aiNet(Ag,L,gen,n,ζ,σd,σs,d);
Ab := gera(N0,L);                          % Gera Ab inicial (N0 qualquer)
% A cada iteração, faça:
for t = 1 to gen,                             % Passo 1
   for j = 1 to M,                            % Passo 1.1
      f(j,:)      := afinidade(Ab,Ag(j,:));  % Passo 1.1.1
      Ab{n}(j,:)  := select(Ab,f(j,:),n);    % Passo 1.1.2
      C           := clona(Ab{n},1,f(j,:));  % Passo 1.1.3
      C*          := dmut(C,Ag(j,:),f(j,:)); % Passo 1.1.4
      f(j,:)      := afinidade(C*,Ag(j,:));  % Passo 1.1.5
      M           := select(C*,f(j,:),ζ);    % Passo 1.1.6
```

```
      [M,f(j,:)]  := suppress(M,1/f(j,:),σd); % Passo 1.1.7
      S           := afinidade(M,M);          % Passo 1.1.8
      [M,S]       := suppress(M,S,σs);        % Passo 1.1.9
      Ab{m}       := insere(Ab{m},M);         % Passo 1.1.8
   end;
   S              := afinidade(Ab{m},Ab{m});  % Passo 1.2
   [Ab{m},S]      := suppress(Ab{m},S,σs);    % Passo 1.3
   Ab{d}          := gera(d,L);               % Gera Abd aleatoriamente
   Ab             := insere(Ab{m},Ab{d});     % Passo 1.4
end;
```

5.5.4. Extração de Conhecimento e Estrutura da aiNet Treinada

Como os anticorpos de memória representam imagens internas dos antígenos, ou grupos de antígenos, aos quais a rede está submetida, eles são representados por cadeias de atributos de mesma dimensão dos dados de entrada (antígenos). Esta propriedade torna o processo de visualização dos anticorpos representados uma tarefa difícil caso a dimensão das moléculas seja superior a três ($L > 3$), o que geralmente ocorre em problemas do mundo real. Para amenizar esta dificuldade, é sugerida a utilização de técnicas de clusterização hierárquica para interpretar a rede resultante, permitindo a identificação e separação automática de clusters sem que seja necessária uma visualização direta da aiNet em um espaço de formas Euclidiano S^L, com L qualquer.

A estrutura da aiNet poderia simplesmente ser determinada interconectando-se todos os anticorpos da rede de acordo com a matriz **S**. Porém, esta estratégia não permite inferir nada sobre a estrutura intrínseca da aiNet e, consequentemente, dos antígenos. Uma forma simples de reduzir a complexidade da arquitetura de uma rede totalmente interconectada é suprimir as conexões cujos pesos extrapolam um limiar pré-definido. Esta idéia, embora simples, desconsidera qualquer informação contida na rede (indiretamente nos dados de treinamento), podendo levar a interpretações incorretas da aiNet resultante. O objetivo principal desta seção é o de fornecer ferramentas formais e sofisticadas de interpretação da aiNet gerada pelo algoritmo de treinamento apresentado na Seção 5.5.2. Explicitamente, objetivamos determinar: 1) a quantidade de anticorpos pertencentes a cada cluster, ou classe (sempre que um cluster corresponder a uma classe), 2) a forma de cada cluster e 3) a pertinência de anticorpos da rede a cada um dos clusters identificados. Para isso, utilizamos os parâmetros de saída da função para o treinamento da aiNet: $\mathbf{Ab}_{\{m\}}$, matriz contendo os anticorpos de memória, e a matriz triangular superior **S** (Seção 5.5.6), contendo a afinidade entre estes anticorpos de memória, mais alguns princípios de análise de clusters.

O problema pode ser colocado da seguinte forma:

Dada uma aiNet treinada composta por m anticorpos de memória (*matriz* $\mathbf{Ab}_{\{m\}} \in S^{m \times L}$), *e suas respectivas afinidades* $s_{i,j}$, *derive um esquema de clusterização capaz de detectar separações inerentes entre subconjuntos* (*clusters*) *de* $\mathbf{Ab}_{\{m\}}$ *em um espaço de formas governado por uma medida de afinidade D.*

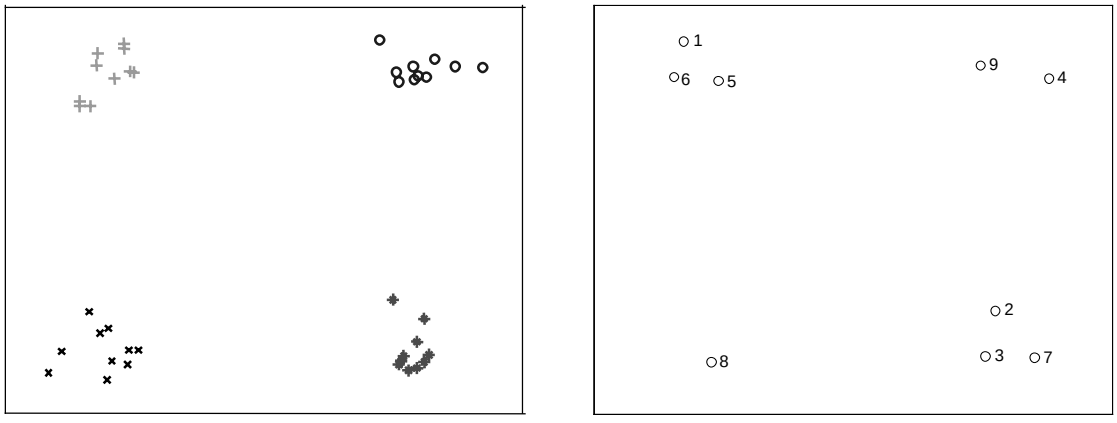
No caso particular do algoritmo implementado, a medida de afinidade D é dada pela distância Euclidiana (Equação 3.1).

Os algoritmos a serem apresentados são conhecidos da literatura de análise de dados, mas serão adaptados e interpretados para o paradigma da engenharia imunológica, mais especificamente da aiNet. Sob esta perspectiva, a aiNet torna-se responsável pela extração de conhecimento e compressão da informação contida no conjunto de antígenos (dados de entrada), enquanto as técnicas de análise de clusters serão utilizadas para a detecção de clusters na rede, ou seja, interpretação da aiNet. A aiNet pode ser vista como um sistema de pré-processamento para as técnicas de análise de clusters, tornando-se uma ferramenta poderosa para a filtragem e preprocessamento de dados de um conjunto amostral.

5.5.4.1. Técnicas de Clusterização Hierárquica

Para ilustrar os métodos a serem empregados e os que serão propostos, considere o problema de clusterizar os dados apresentados na Figura 5.14(a). Como pode ser visto nesta figura, existem 40 amostras subdivididas em quatro clusters distintos e linearmente separáveis com 10 amostras cada. A Figura 5.14(b) representa os anticorpos resultantes de um treinamento da aiNet para os seguintes parâmetros de entrada: *gen* = 10, $n = 4$, $\zeta = 0.2$, $\sigma_d = 1.0$, $\sigma_s = 0.1$ e $d = 10$. A população inicial de anticorpos continha $N_0 = 10$ moléculas. As técnicas de clusterização hierárquica serão utilizadas para determinar quais anticorpos da rede estarão conectados entre si. Como pode ser visto, a rede resultante contém apenas 9 anticorpos, reduzindo a complexidade do problema (quantidade de dados) em 77,5%.

As técnicas de clusterização hierárquica podem ser subdivididas em métodos *aglomerativos* que fazem sucessivas fusões de *m* dados em grupos, e métodos *divisivos* que particionam o conjunto de *m* dados sucessivamente em partições menores. Os resultados de ambos os métodos, aglomerativos e divisivos, podem ser representados sob a forma de um *dendrograma*, que é um diagrama bidimensional que ilustra as fusões ou partições feitas em cada nível (Everitt, 1993). Para análise da aiNet, enfocaremos os métodos aglomerativos, mais especificamente o método do *vizinho mais próximo* (*nearest neighbor* ou *single link*), e a análise de clusters baseada no *centróide*.



(a) (b)

Figura 5.14. Problema didático ilustrativo. (a) Antígenos. (b) Anticorpos resultantes da aplicação do algoritmo de treinamento da aiNet ao problema apresentado em (a). Parâmetros de treinamento: gen = 10, $n = 4$, $\zeta = 0.2$, $\sigma_d = 1.0$, $\sigma_s = 0.1$ e $d = 10$.

Dada a conceituação da aiNet sob a forma de um grafo interconectado de nós, ou anticorpos (Definição 5.1), algumas estratégias da teoria de grafos para detecção e descrição de clusters serão empregadas, em particular a *árvore geradora mínima* (*minimal spanning tree*).

Um dos parâmetros de saída da função que implementa a aiNet (Seção 5.5.3) é a matriz **S** de dimensão $N \cdot (N - 1)/2$. A matriz **S** de afinidades entre os anticorpos de memória da rede e sua matriz de coordenadas $\mathbf{Ab}_{\{m\}}$ estão disponíveis, portanto a aplicação de técnicas hierárquicas de clusterização como o vizinho mais próximo, o vizinho mais distante, e o centróide, para a construção do dendrograma, é direta. De forma resumida, estes algoritmos operam como a seguir:

1. *Vizinho mais próximo* (*nearest neighbor-NN* ou *single link*): grupos compostos inicialmente por um único indivíduo são fundidos de acordo com a distância entre seus membros mais próximos, os grupos com menores distâncias sendo fundidos primeiro. Cada fusão reduz em uma unidade a quantidade de grupos.
2. *Vizinho mais distante* (*furthest neighbor* ou *complete link*): este método opera de forma exatamente oposta ao anterior, no sentido de que a distância entre grupos é tomada entre os pares de indivíduos mais distantes.
3. *Centróide*: os grupos estão contidos em um espaço Euclidiano e são substituídos pelas coordenadas de seus centróides. A distância entre grupos é definida como a distância entre seus centróides. O procedimento funde grupos de acordo com a distância entre seus centróides. Os grupos com menores distâncias sendo fundidos primeiro.

A partir da matriz de similaridades **S** e empregando os métodos descritos acima, deseja-se construir uma árvore, ou conjunto aninhado de clusters dos elementos, de forma a produzir um apelo visual da similaridade entre os grupos gerados pelos anticorpos da rede. Antes de definir o que é um dendrograma, considere os seguintes conceitos da teoria de grafos (Zahn, 1971):

- *Grafo G ponderado*: conjunto de pontos chamados *nós* (ou anticorpos, no caso da aiNet) e um conjunto de pares de nós chamados de *arcos* (ou conexões, no caso da aiNet) com um *peso* associado a cada arco;
- Um *caminho* em um grafo é uma seqüência de arcos que têm um nó comum;
- Um *circuito* é um caminho fechado;
- Um *grafo conexo* possui caminhos entre quaisquer pares de nós; e
- Uma *árvore* é um grafo conexo sem nenhum circuito.

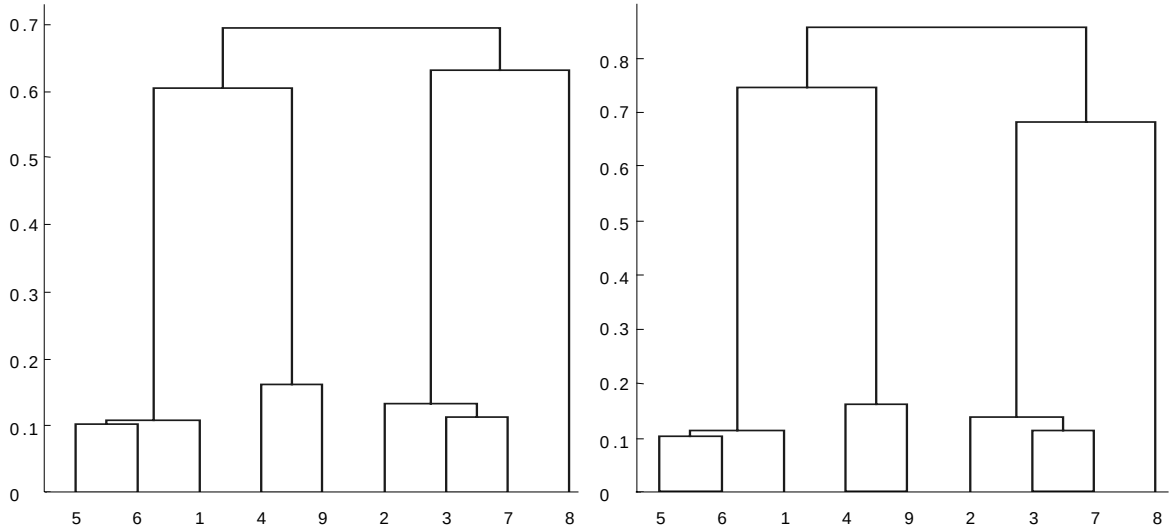
Definição 5.3: Um *dendrograma* é definido como uma árvore com raiz e ponderada, onde todos os nós terminais estão a uma mesma distância (comprimento do caminho) da raiz (Lapointe & Legendre, 1991).

Detalhes sobre a construção de um dendrograma a partir de uma matriz de similaridade (ou dissimilaridade) podem ser encontrados em Hartigan (1967) e Hubert *et al.* (1998). Para a interpretação da aiNet, três características podem descrever adequadamente um dendrograma: 1) topologia, 2) rótulos e 3) altura dos clusters (Lapointe & Legendre, 1995):

1. A *topologia* de um dendrograma representa sua estrutura ou forma. Ela é influenciada pelo algoritmo de clusterização hierárquica empregado;
2. Os *rótulos* identificam os *m* elementos sendo clusterizados; e
3. A *altura dos clusters* permite que a distância entre os clusters seja descrita por uma escala de alturas associada aos $m - 1$ nós do dendrograma, partindo da raiz.

A Figura 5.15 ilustra dois dendrogramas para os anticorpos apresentados na Figura 5.14(b). Observe a topologia, rótulos e alturas dos clusters. A Figura 5.15(a) ilustra o dendrograma para o método do vizinho mais próximo e a Figura 5.15(b) corresponde ao método do centróide.

Virtualmente nenhum procedimento de clusterização fornece informações quanto ao número de clusters presentes nos dados. Procedimentos não-hierárquicos usualmente requerem a especificação deste parâmetro antes de que a clusterização seja efetuada (razão pela qual escolhemos os métodos hierárquicos ao invés dos não-hierárquicos) e os métodos hierárquicos produzem uma série de soluções variando de *m* clusters até uma solução com um único cluster. Como pode ser visto na Figura 5.15, o dendrograma pode ser quebrado em diferentes níveis resultando em clusterizações distintas dos dados. Neste caso, as grandes variações nas alturas dos clusters permitem-nos distinguir quatro clusters entre os anticorpos da rede, estando de acordo com a Figura 5.14(b). Este procedimento é chamado de *tamanho do passo* (*stepsize*) e envolve a análise da diferença entre os *valores de fusão* entre os níveis hierárquicos. Uma ampla revisão sobre diferentes técnicas de determinação do número de clusters em um conjunto de dados pode ser encontrada em Milligan *et al.* (1985).



(a) (b)

Figura 5.15. Dendrogramas para os anticorpos apresentados na Figura 5.14, o eixo *x* contém os rótulos dos anticorpos da rede e o eixo *y* apresenta a altura dos clusters. (a) Vizinho mais próximo. (b) Centróide.

A *árvore geradora mínima* (MST – *minimal spanning tree*) de um grafo constitui um mecanismo poderoso para determinação de uma estratégia adaptativa de detecção do esquema de conectividade do grafo (Zahn, 1971; Leclerc, 1995). Sendo assim, a MST funcionará como uma alternativa para a determinação da arquitetura da aiNet.

Definição 5.4: Uma árvore é uma *árvore geradora* (*spanning tree*) se ela é um subgrafo contendo todos os nós do grafo.

Definição 5.5: Uma *árvore geradora mínima* de um grafo é uma árvore geradora com peso mínimo, onde o peso de uma árvore é definido como sendo a soma dos pesos de seus arcos.

A Figura 5.16(a) ilustra a árvore geradora mínima para os anticorpos definidos pela aiNet. A visualização desta árvore só é factível caso a dimensão dos anticorpos seja inferior a 3 ($L < 3$). Utilizando-se o algoritmo conhecido como *algoritmo de Prim* (Prim, 1957) para construir a MST, é possível desenhar um histograma como na Figura 5.16(b) representando as distâncias entre anticorpos vizinhos.

Definição 5.6: Um *caminho minimax* entre um par de nós é aquele que minimiza o custo (peso máximo do caminho), sobre todos os caminhos.

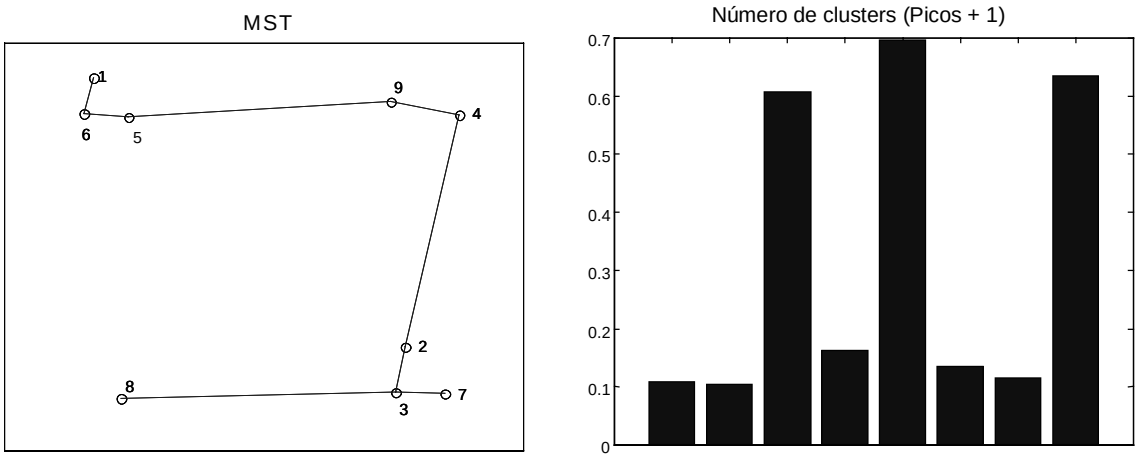
Esta definição é importante no contexto da aiNet, pois a MST percorre os caminhos minimax, forçando a conexão entre dois nós *i* e *j* mais próximos antes de sair em busca de outro nó. Se a MST de um grafo *G* é única, então o conjunto de caminhos minimax de *G* define a MST, senão ele define a união de todas as MSTs de *G* (Carroll, 1995).

A MST pode ser utilizada para determinar a quantidade de clusters da aiNet, que será igual ao número de picos de seu *histograma* mais 1, como ilustrado na Figura 5.16(b). Quando o algoritmo de treinamento da aiNet gera mais do que um anticorpo para cada cluster, a quantidade de clusters também pode ser medida pelo número de vales do respectivo histograma, porém, sem que haja uma visualização dos anticorpos resultantes, é difícil avaliar a quantidade de anticorpos gerados para cada cluster antes de fazer a análise do histograma da MST. Estes picos (vales) no histograma correspondem a grandes variações nas distâncias minimax entre os anticorpos gerados, permitindo a identificação de regiões com alta densidade de anticorpos intercaladas por regiões de baixa densidade no espaço de formas. Por outro lado, os dendrogramas nos permitem não apenas definir a quantidade de clusters, mas também os elementos pertencentes a cada um deles. Para que seja possível definir automaticamente a quantidade de clusters e seus elementos a partir de uma MST, podemos utilizar algumas das técnicas propostas por Zahn (1971).

O problema agora resume-se a remover conexões de uma MST tal que as subárvores resultantes correspondam aos clusters observáveis da aiNet. No exemplo da Figura 5.14, é preciso um algoritmo capaz de detectar e remover as conexões 5-9, 2-4 e 3-8, destacadas na Figura 5.17.

O seguinte critério é empregado:

Uma conexão (*i*,*j*) *da MST cujo peso associado* $s_{i,j}$ *for significativamente maior do que a média dos pesos vizinhos, em ambos os lados da conexão* (*i*,*j*), *deverá ser removida. Esta conexão é chamada inconsistente.*



(a) (b)

Figura 5.16. Determinação da quantidade de clusters da aiNet. (a) MST. (b) Histograma da MST apresentada em (a).

Existem duas formas diretas de medir a significância ou inconsistência mencionada acima:

1. Verificar quantos desvios padrões separam $s_{i,j}$ do peso médio dos vizinhos em cada lado da conexão (i,j); e
2. Calcular o *fator*, ou *razão*, r entre $s_{i,j}$ e as respectivas médias. Neste caso, uma desigualdade triangular não se faz necessária e, portanto, não precisamos restringir a análise a um espaço métrico.

Para ilustrar este critério, suponha que todas as conexões da MST cujo $s_{i,j}$ é maior do que a média dos pesos das conexões vizinhas mais 2 desvios padrões serão removidas, ou seja, um fator $r = 2$ é adotado. As conexões 5-9, 2-4 e 3-8 serão selecionadas para remoção (linhas tracejadas na Figura 5.17).

Após determinar quais conexões serão removidas, é possível detectar a quantidade de clusters existentes na aiNet e seus respectivos anticorpos. Se c_i, $i = 1,..., n_c$ (n_c = quantidade de clusters), representa o i-ésimo cluster da rede, então $c_1 = [1,6,5]$, $c_2 = [9,4]$, $c_3 = [2,3,7]$ e $c_4 = [8]$.

O critério proposto falharia na determinação correta dos clusters nas seguintes ocasiões:

- Quando o algoritmo de treinamento da aiNet for capaz de determinar a arquitetura mínima para um certo conjunto de dados, ou seja, um único anticorpo representando cada cluster. Este problema pode ser aliviado através de uma redução no valor do parâmetro σ_s, resultando em uma rede com anticorpos mais específicos. Além disso, geralmente a quantidade de clusters dos dados não é conhecida, e caso seja, σ_s pode ser ajustado para que a aiNet atinja uma dimensão satisfatória; e
- Quando não existir uma distância relativa entre os clusters que seja maior do que a distância entre elementos do mesmo cluster. Embora essa pareça uma limitação do algoritmo, qualquer outra estratégia não-supervisionada de clusterização, baseada em critérios de distância, falharia nesta situação.

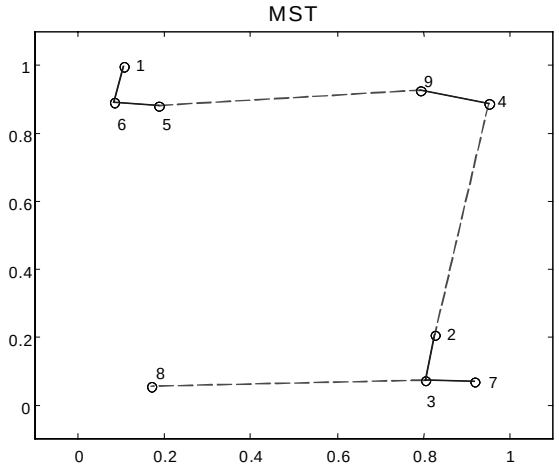


Figura 5.17. Conexões a serem removidas (linhas tracejadas) baseadas no fator $r = 2$.

Como último aspecto de clusterização a ser considerado, temos o problema da *representação dos clusters*. Suponha que cada cluster possa ser unicamente representado pelo seu *centro de massa*, ou *centróide*, v_i, $i = 1,...,n_c$, e que a distância entre os clusters seja definida pela distância entre os centróides. A Figura 5.18 ilustra os anticorpos resultantes do treinamento da aiNet e as conexões automaticamente determinadas pelo método proposto para a MST considerando um fator $r = 2$. As estrelas representam os centróides de cada cluster.

A representação dos clusters pelo seu centróide de massa é geralmente útil quando os clusters são compactos e isotrópicos (com formas semelhantes). Nos casos em que os clusters são não-compactos e/ou não-isotrópicos, este esquema não é apropriado para representá-los. Estes casos ficarão mais claros no próximo capítulo, onde apresentaremos simulações computacionais do algoritmo. A representação dos clusters pelos seus centros de massa é interessante pois ela permite especificar *níveis de pertinência* de cada anticorpo da rede a cada um dos clusters detectados, resultando em um esquema de *clusterização gradual* (ou *nebulosa*).

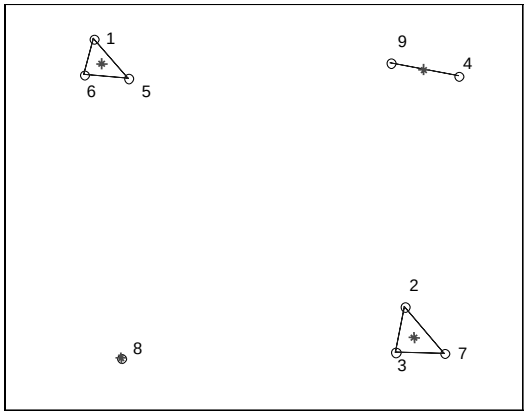


Figura 5.18. aiNet resultante. A rede é composta por quatro clusters (subgrafos) distintos. As estrelas representam o centróide de cada cluster.

5.5.4.2. Clusterização Gradual da aiNet

Os métodos de clusterização discutidos na seção anterior geram *partições* no conjunto de dados. Em uma partição, cada elemento pertence a apenas um cluster. Assim, os clusters são disjuntos neste esquema de clusterização. A *clusterização fuzzy*, ou *clusterização nebulosa*, estende esta noção de forma a associar cada padrão a todos os clusters através de uma função de pertinência (Bezdek & Pal, 1992). As técnicas de clusterização nebulosa mais conhecidas são os algoritmos de *fuzzy k-means* e *fuzzy c-means*, que iterativamente atualizam os centros dos clusters de acordo com uma matriz de proximidade **U** atual, até que a variação em **U** seja desprezível. Uma breve exposição de espaços de partições nebulosa é apresentada por Bezdek & Pal (1992):

> Seja n_c um inteiro, $1 < n_c < M$, e seja $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_M\}$ um conjunto de M dados não rotulados em $\Re^L$. Dado **X**, dizemos que n_c subconjuntos nebulosos $\{u_i: \mathbf{X} \rightarrow [0,1]\}$ constituem uma *c-partição nebulosa* de **X** caso os $(n_c.M)$ valores $\{u_{i,k} = u_i(\boldsymbol{x}_k),\ 1 \leq k \leq M,\ 1 \leq i \leq n_c\}$ satisfaçam as três condições abaixo:
>
> $0 \leq u_{i,k} \leq 1$ para todo i, k; **(5.24)**
>
> $\Sigma\, u_{i,k} = 1$ para todo k; **(5.25)**
>
> $0 < \Sigma\, u_{i,k} < 1$ para todo i. **(5.26)**
>
> Cada conjunto de $(n_c.M)$ valores satisfazendo as condições de (5.24) a (5.26) pode ser colocado em uma matriz $\mathbf{U} = [u_{i,k}]$, $\mathbf{U} \in \Re^{nc \times M}$. O conjunto de todas essas matrizes é chamado de *c-partição nebulosa não degenerada* de **X**.

Após determinar a quantidade e os membros de cada cluster e representá-los pelos seus respectivos centróides, é possível aplicar o conceito de clusterização nebulosa (ou gradual), em que cada anticorpo terá um valor de pertinência em relação a cada um dos centróides dos clusters. No contexto da engenharia imunológica, a clusterização nebulosa relaxa a pertinência dos anticorpos da rede em relação aos centróides dos clusters, $\mathbf{U} = [u_{i,k}]$, que neste caso, pode assumir qualquer valor no intervalo [0,1], satisfazendo a condição (5.24). As condições (5.25) e (5.26) são relaxadas, de forma que a soma das pertinências não seja necessariamente igual a 1. Esta é uma forma natural de estender as *c*-partições nebulosas não-degeneradas de **X** a um espaço de soluções nebulosas ainda maior.

A matriz **U** para a aiNet pode ser determinada calculando-se a distância **U*** entre todas os m anticorpos de memória resultantes da rede e os n_c centróides dos clusters v_i, $i = 1,\ldots,n_c$, normalizando suas linhas no intervalo [0,1] e passando-as por uma função não-linear, de forma que quanto menor a distância entre um anticorpo e seu respectivo centróide, mais perto da unidade estará seu valor de pertinência. Este último passo pode ser efetuado aplicando-se uma função do tipo sigmoidal a $\mathbf{U} = 1./\mathbf{U}^*$, onde o operador ./ significa que cada valor da matriz **U** será determinado pela inversão do respectivo elemento de **U***.

5.5.5. Caracterização e Comparação

Baseado nas propriedades das redes neurais artificiais apresentadas na Seção 4.2, é possível classificar a aiNet como uma abordagem *conexionista*. Sob esta perspectiva, os anticorpos da rede correspondem a seus nós, a posição dos anticorpos no espaço de formas é o *estado do nó*, os valores de concentração e afinidade determinam o perfil de conectividade da rede, e os antígenos correspondem aos *padrões de treinamento*. As conexões entre os nós, representadas pela matriz **S** (ilustradas genericamente na Figura 5.1(a) e para o caso particular do exemplo tratado nesta seção na Figura 5.18), permitem avaliar a afinidade existente entre os nós da rede, quantificando o reconhecimento idiotípico. A *representação em grafo* da aiNet descreve sua *arquitetura* final, definindo a quantidade de clusters e seus elementos constituintes. A *dinâmica* da rede governa a aprendizagem e plasticidade do sistema, enquanto a *metadinâmica* é responsável por uma melhor exploração do espaço de formas e manutenção da diversidade populacional.

O algoritmo de treinamento pode ser classificado como *competitivo* (Seção 4.2.1.2.2), pois um conjunto de anticorpos está competindo entre si pelo reconhecimento antigênico e conseqüente sobrevivência. A estrutura geral da aiNet é diferente das redes neurais artificiais quanto à funcionalidade das unidades e conexões. No caso das RNAs, os nós são unidades de processamento e as conexões armazenam as informações que definem a função de transferência da rede (conhecimento). Na aiNet, as unidades armazenam as informações (imagens internas do universo antigênico), as conexões quantificam a afinidade, ou grau de interação dos anticorpos da rede, e o processamento é efetuado pelo algoritmo de seleção clonal (CLONALG para reconhecimento de padrões) somado a mecanismos de interação e supressão da teoria da rede imunológica. A Tabela 4.1 apresenta genericamente as principais similaridades entre a engenharia imunológica e as redes neurais artificiais.

Além da visão conexionista da aiNet, é possível categorizar seu algoritmo de treinamento como *evolutivo*, pois ele segue os principais passos do algoritmo CLONALG, proposto na Seção 5.3 e caracterizado na Seção 5.3.3. Uma vantagem da evolução da aiNet sobre a evolução do CLONALG é a utilização da teoria da rede imunológica para o controle da quantidade de anticorpos na rede. Entretanto, a introdução de um mecanismo para avaliar a afinidade entre os anticorpos da rede traz um elevado custo computacional associado, como será apresentado na próxima seção.

A essência deste modelo de rede imunológica artificial é diferente da maioria dos modelos existentes na literatura em dois aspectos. Primeiro, e mais importante, ele é discreto no tempo (iterativo) e não contínuo (baseado em equações diferenciais ordinárias). Segundo, o objetivo principal da aiNet é o de resolver problemas de engenharia e não reproduzir fenômenos biológicos objetivando uma melhor compreensão dos mesmos.

Existem dois modelos na literatura que podem ser diretamente comparáveis à aiNet: uma rede imunológica artificial proposta por Hunt & Cooke (1996) e outra por Timmis (2000).

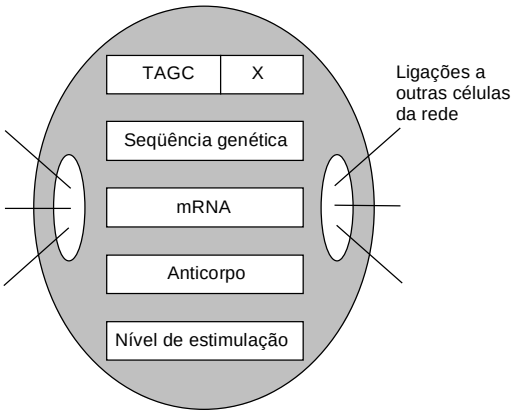


Figura 5.19. Estrutura de uma célula B (Hunt & Cooke, 1996).

O sistema imunológico artificial desenvolvido por Hunt & Cooke (1996) apresenta um modelo de rede imunológica artificial contendo os seguintes elementos:

1. Medula óssea: responsável pela geração das células do SIA e definição do ponto de inserção do antígeno na rede;
2. Célula B: composta por diversos elementos, como ilustrado na Figura 5.19, onde as bibliotecas genéticas, a seqüência de genes e o mRNA fazem parte do procedimento de geração das moléculas de anticorpo. Este modelo também contém o nível de estimulação da célula;
3. Anticorpos: possuindo um paratopo, a medida de afinidade e o processo de ligação;
4. Antígeno: composto por um único epítopo.

A Tabela 5.4 faz uma comparação entre a aiNet e o modelo de Hunt & Cooke (1996).

Tabela 5.4. Relação entre a aiNet e o SIA proposto por Hunt & Cooke (1996).

	aiNet	**SIA**
Nós	Cadeia de atributos (anticorpos)	Célula B (Figura 5.19)
Inicialização da rede	Aleatória, com pouca influência no comportamento final da aiNet	Crítica para o tempo de processamento
Apresentação antigênica	O antígeno é apresentado para toda a rede	O antígeno é apresentado para uma região aleatória da rede
Medida de afinidade	Equação 3.3	Equação 3.4
Morte celular	Eliminação de anticorpos com baixa afinidade antigênica e alta afinidade em relação a outro anticorpo	Eliminação de células com baixo nível de estimulação (considera interações Ag-Ab e Ab-Ab, Equação (5.27))
Hipermutação	Importante para a aprendizagem	Promove diversidade

O nível de estimulação de cada célula, dado pela Equação (5.27), e a hipermutação somática deste SIA são baseadas em uma variação do modelo de rede proposto por Farmer *et al.* (1986) e brevemente apresentado na Seção 3.4.3.5.

$$estimulação = c\left[\sum_{j=1}^{N} m(a, xe_j) - k_1 \sum_{j=1}^{N} m(a, xp_j) + k_2 \sum_{j=1}^{n} m(a, y)\right] - k_3 \quad \textbf{(5.27)}$$

onde o primeiro termo representa a afinidade dos anticorpos em relação a seus vizinhos, o segundo termo representa a não-afinidade dos anticorpos em relação a seus vizinhos, o terceiro termo corresponde à ligação Ag-Ab e o último termo modela a tendência das células pouco estimuladas morrerem. Os coeficientes k_i, $i = 1,...,3$, são constantes.

Este sistema imunológico artificial foi posteriormente aperfeiçoado (Hunt *et al.*, 1999) para eliminar algumas de suas limitações, como inicialização eficiente da rede e aplicação de medidas alternativas de afinidade. Timmis (2000) consolidou esta proposta de SIA através do desenvolvimento de uma técnica para o controle da dimensão da arquitetura resultante da rede, e de diversas aplicações e extensivas comparações com outras abordagens na área de análise de dados.

5.5.6. Complexidade Computacional

O algoritmo proposto para o treinamento da aiNet possui um laço interno que percorre todos os antígenos da população e um laço externo que varre as gerações de anticorpos. No laço interno (dos antígenos), seis etapas principais de processamento podem ser identificadas, juntamente com seus respectivos custos computacionais associados:

1. A determinação da afinidade entre o antígeno dado e todos os anticorpos da rede (Passo 1.1.1): custo $O(N)$;
2. A seleção dos n melhores indivíduos (Passo 1.1.2): custo $O(N)$;
3. A mutação direcionada da população **C** de clones gerados (Passo 1.1.4): custo $O(N_c.L)$;
4. A determinação da afinidade entre o antígeno dado e todos os N_c clones gerados (Passo 1.1.5): custo $O(N_c)$
5. A re-seleção dos ζ melhores indivíduos maduros (Passo 1.1.6): custo $O(N_c)$; e
6. O cálculo da afinidade entre os c clones de memória (Passo 1.1.8): custo $O(c^2)$, onde c corresponde a $\zeta.N$ anticorpos, menos aqueles eliminados no Passo 1.1.7.

No laço externo (rede), a principal etapa de processamento é a determinação da afinidade entre os m anticorpos de memória da rede, com custo associado $O(m^2)$, semelhante ao algoritmo SAND (Seção 5.2). Devido à característica assintótica da complexidade e assumindo que $m > c$, o termo de ordem $O(m^2)$ predomina sobre todos os outros, portanto a complexidade da aiNet está resumida na Tabela 5.5. Note que a quantidade m de anticorpos de memória na rede varia dinamicamente, sendo maior nas primeiras iterações, como será visto na Seção 6.5.3, que faz uma análise de sensibilidade dos parâmetros da aiNet definidos pelo usuário.

Tabela 5.5. Complexidade computacional por geração para execução do algoritmo de treinamento da aiNet, onde N é o tamanho atual da população de anticorpos, N_c a quantidade de clones gerada a partir da seleção dos n melhores indivíduos de **Ab**, L é o comprimento das cadeias de atributos que representam as moléculas e m é a quantidade de anticorpos de memória.

	Tempo de Processamento	***Memória***
aiNet	$O(m^2)$	$\propto N + L(n + N_c + N) + m(m-1)/2$

A quantidade de memória necessária para executar o algoritmo é igual à do algoritmo CLONALG proposto na Seção 5.3, mais o custo de se armazenar a matriz **S** contendo as afinidades entre os m anticorpos de memória da rede. Lembrando que esta matriz é simétrica e tem diagonal principal nula, permitindo armazenar apenas $m(m-1)/2$ elementos, a quantidade de memória requerida para o algoritmo de treinamento da aiNet está apresentada na Tabela 5.5.