3
Redes Neurais

3.1. Introdução

As redes neurais artificiais, ou comumente conhecidas como "*Neural Networks*", foram motivadas em princípio pela extraordinária capacidade do cérebro humano para executar tarefas de grande complexidade, não lineares e com processamento paralelo da informação, tais como reconhecimento de padrões, percepção, classificação, generalização de conceitos e controle motor. Estes fatos geraram o interesse do estudo detalhado da constituição do cérebro e sua mimetização na concepção de sistemas com as capacidades acima referidas, designadas por redes neurais artificiais (*ANN*).

As redes neurais (*NN*) estão constituídas por unidades básicas independentes designadas como neurônios (processadores ou nós). Cada unidade possui ligações para outras unidades, as quais se comunicam entre si através de sinapses, formando uma rede de nós, daí o nome de rede neural.

As primeiras informações sobre redes neurais surgiram em 1943, com o primeiro modelo lógico-matemático de um neurônio biológico, que foi desenvolvido pelo neurofisiólogo Warren McCulloch, do *Instituto Tecnológico de Massachusetts*, e do matemático Walter Pitts, da *Universidade de Illinois*. Desenvolveram um modelo matemático de neurônio simulando o comportamento de uma célula nervosa, a qual consistia num modelo de resistores variáveis e amplificadores.

As redes neurais realizam o processamento de informações baseado na organização dos neurônios do cérebro, as quais têm a capacidade de aprender e tomar decisões baseadas na aprendizagem. Uma rede neural é interpretada como uma estrutura neural de organismos inteligentes, capaz de armazenar conhecimento baseado em aprendizagem e da experiência.

Uma rede neural é um processador maciçamente paralelo, distribuído, constituído por unidades de processamento simples, que têm a capacidade de armazenamento de conhecimento experimental e de torná-lo disponível para uso. As redes neurais são semelhantes ao cérebro pelos seguintes aspectos [14]:

- O conhecimento é obtido pela rede neural a partir do ambiente, através do processo de aprendizado.
- A força de conexão entre cada neurônio, conhecida como peso sináptico, é usada para armazenar o conhecimento aprendido.

3.2. Estrutura do neurônio

O neurônio é a unidade fundamental de processamento de informação para a operação de uma *NN*. Neurônios são elementos processadores interligados, trabalhando em paralelo para desempenhar uma determinada tarefa. A Figura 3.1 mostra a estrutura de um neurônio biológico, e na Figura 3.2 apresenta-se o modelo de um neurônio artificial, na qual se desenham formas básicas que constituem uma *NN*. Os elementos básicos identificados no modelo neural são: os *pesos de conexão*, o N*et* e a *Função de ativação* [15].

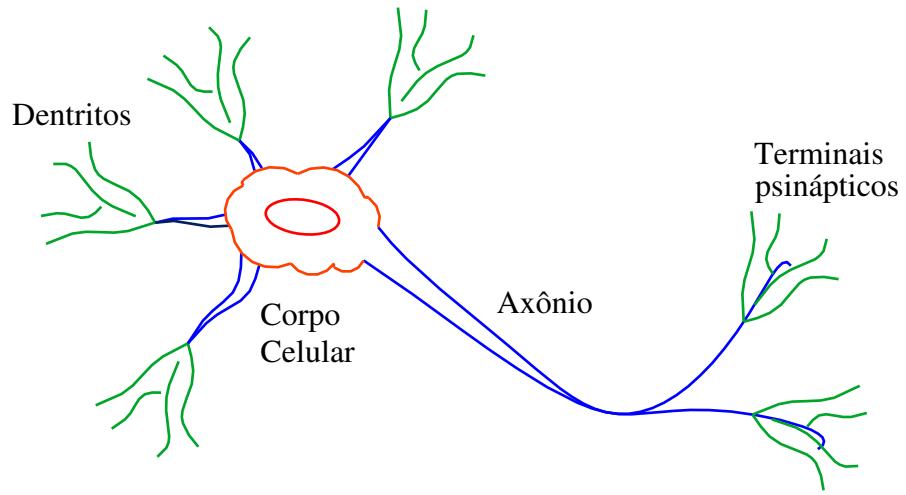

Figura 3.1. Esquema básico do neurônio biológico.

O neurônio recebe múltiplos sinais de outros neurônios através de seus dentritos, e cada um destes sinais são multiplicados pelo próprio peso da conexão. Estes sinais são adicionados no corpo celular ou função somatória, e quando este sinal composto alcança um valor umbral, um sinal potencial é enviado pelo axônio, o qual é a saída do neurônio [14].

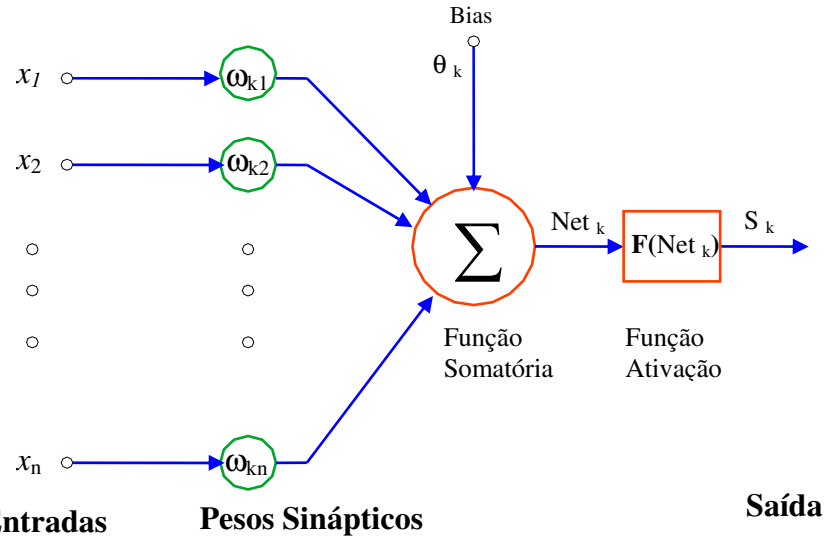

Figura 3.2. Modelo básico de um neurônio Artificial ”Perceptron”.

onde x são as entradas, ω_{kn} são os pesos ou parâmetros da rede, os quais representam a memória da rede, θ_k é conhecido como o termo polarizador (bias), Net_k representa a combinação linear dos pesos, e S_k representa a saída do neurônio.

O modelo do neurônio da Figura 3.2, além das entradas, também tem um termo constante chamado “bias” θ_k, o qual tem a função de aumentar ou diminuir a entrada Net_k da função de ativação, modificando este positiva ou negativamente.

3.2.1. Peso de conexão

O valor de sinapses, ou conexão, é caracterizado por um peso. Especificamente, o sinal de entrada x_j da sinapse j conectada para o neurônio k é multiplicado pelo peso sináptico ω_{kj}. O primeiro sub-índice faz referência ao neurônio, e o segundo faz referência à entrada.

3.2.2. Função Somatório (Net)

A *Net* é definida como uma função soma, e executa o somatório dos sinais produzidos pelo produto entre os sinais de entrada e os pesos de conexão

(sinapses) de cada neurônio. Esta operação representa uma combinação linear:

$$Net_k = \sum_{j=1}^{p} \omega_{kj}.x_j + \theta_k \tag{3.1}$$

3.2.3. Função de Ativação

A função de ativação é uma função que determina o nível de ativação do neurônio artificial, o qual limita a amplitude do sinal de saída do neurônio para uma faixa de valores finitos. Geralmente, escreve-se a faixa de amplitude normalizada da saída de um neurônio como um intervalo fechado [0,1] ou [-1,1] [14][16].

- **Função Degrau**

  Modelo proposto por McCulloch e Pits[1943], esta função de ativação modela a característica de "todo - ou- nada" do neurônio, e é expressa pela equação:

$$F(Net_k) = \begin{cases} 1 & \text{Se} \quad Net_k \geq 0 \\ 0 & \text{Se} \quad Net_k < 0 \end{cases} \tag{3.2}$$

  Na Figura 3.3(a), apresenta-se a função degrau.

- **Função de Ativação Linear**

  A função linear é apresentada na Figura 3.3(b), onde o fator de amplificação dentro da região de operação é assumido unitário. Portanto, esta função é considerada como uma aproximação para um amplificador não linear, e é apresentada pela equação:

$$F(Net_k) = \begin{cases} 1 & \text{Se} \quad 0,5 \leq Net_k \\ Net_k & \text{Se} \quad -0,5 < Net_k < 0,5 \\ 0 & \text{Se} \quad Net_k < -0,5 \end{cases} \tag{3.3}$$

- **Função de Ativação Logsig**

  A função *logsig* é a função de ativação mais comumente utilizada na construção de *ANN*. É definida como uma função estritamente crescente, que apresenta um equilíbrio entre o desempenho linear e não-linear, e definida por:

$$F(Net_k) = \frac{1}{1+e^{-a.Net_k}} \tag{3.4}$$

onde a é o parâmetro de inclinação da função *logsig* e Net_k é o valor de ativação do neurônio. Na Figura 3.3(c), apresenta-se a função *logsig*, e observa-se que à medida que o parâmetro a aumenta, tendendo-o ao infinito, esta função comporta-se como uma função de grau.

- **Função de Ativação Tansig**

  Na Figura 3.3 (d) apresenta-se a função *Tansig*. Esta função também possui a forma sigmoidal, e é a segunda função mais utilizada na construção de *ANN*. A principal diferença com relação à função *Logsig* é que pode assumir valores na faixa de [-1, 1], e é definida pela equação.

$$F(Net_k) = \frac{2}{1+e^{-a.Net_k}} - 1 \tag{3.5}$$

O parâmetro a, assim como na Equação (3.4), determina a inclinação da função.

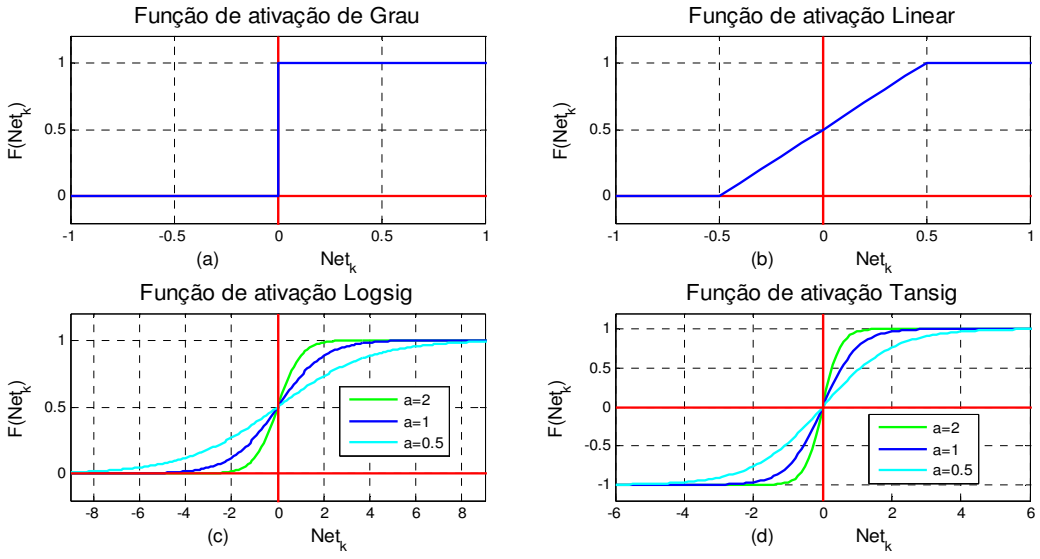

Figura 3.3. Funções de Ativação: (a) Função de Grau. (b) Função linear. (c) Função Logsig. (d) Função Tansig.

3.3. Arquitetura das Redes Neurais

A arquitetura das redes neurais artificiais pode ser formada por uma ou mais camadas de neurônios. Estas camadas estão formadas por neurônios enfileirados e ligados entre si. A entrada de uma camada intermediária " p " é a saída da camada anterior " $p-1$ " ou a camada de entrada, e a saída desta mesma camada " p " é uma entrada da camada seguinte " $p+1$ " ou a camada de saída, como se apresenta na Figura 3.4.

As redes neurais, baseadas em sua arquitetura, são classificadas em dois grupos, as redes Perceptron e as redes Multilayer Perceptron (*MLP*).

3.3.1. Rede Perceptron

O perceptron foi proposto por Rosenblatt [1962], dando continuidade ao trabalho de McCulloch-Pitts. Este pode ser considerado o primeiro modelo de redes neurais, e é capaz de aprender somente sobre problemas linearmente separáveis. Sua arquitetura consiste em uma camada de entrada e uma de saída, como se apresenta na Figura 3.4.

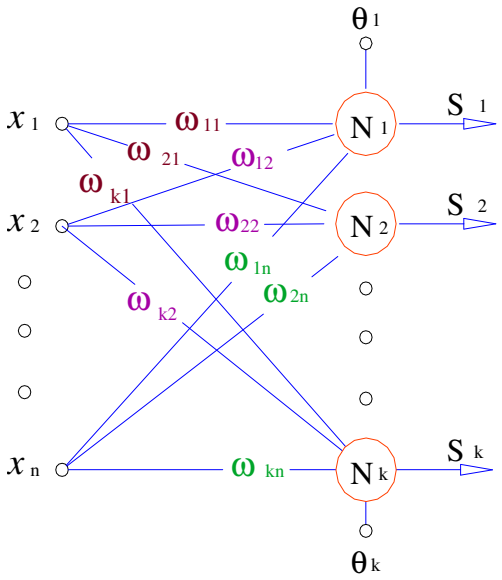

Figura 3.4. Rede perceptron com k Neurônios de saída.

onde $X=[x_1\ x_2\ \ldots x_n]$ representa a camada de entrada, N_k é o $k-ésimo$ neurônio, ω_{jk} é o peso do neurônio N_k em relação à entrada x_j, e S_k representa

a $k-ésima$ saída.

Seguindo a regra de propagação, o valor de Net do $j-ésimo$ neurônio é definido pela equação

$$Net_j = \sum_{i=1}^{k} \omega_{ji}.x_i + \theta_j \tag{3.6}$$

Geralmente, nas redes perceptron a função de ativação é a "*função Degrau*", e sua saída S_j é determinada por

$$S_j = \begin{cases} 1 & \text{Se } Net_j > 0 \\ 0 & \text{Se } Net_j \leq 0 \end{cases} \tag{3.7}$$

Durante o processo de treinamento do perceptron, busca-se encontrar um conjunto de pesos que determine uma reta que separe as diferentes classes, de maneira que a rede possa classificar corretamente as entradas apresentadas. O algoritmo de aprendizado para obter os pesos desejados é determinado de a maneira descrita a seguir.

Para um padrão de entrada, obtêm-se uma saída da rede s_j, com seu respectivo valor de saída desejado t_j. O erro na saída é definido por

$$e_j = t_j - s_j \tag{3.8}$$

Então, a variação do peso $\Delta\omega_{ji}$ é definida por

$$\Delta\omega_{ji} = \eta.x_i.e_j \tag{3.9}$$

onde η é a taxa de aprendizado.

Entretanto, os pesos são atualizados depois de cada entrada com a regra

$$\omega_{ji}^{j+1} = \omega_{ji}^{j} + \Delta\omega_{ji}^{j} = \omega_{ji}^{j} + \eta.x_i.e_j \tag{3.10}$$

O algoritmo de aprendizado aplicado em um dos neurônios da rede perceptron é o mesmo para todos os demais neurônios.

O perceptron de um único neurônio é limitado a trabalhar na classificação de padrões com apenas duas classes. Entretanto, o incremento dos neurônios na camada de saída do perceptron permite classificar padrões de mais de duas classes, sempre que as classes forem linearmente separáveis [17].

3.3.2. Rede Multilayer Perceptron

A rede multilayer perceptron (*MLP*) é considerada como uma das mais importantes *ANN*. Este tipo de rede tem sido aplicado com sucesso em diversas áreas, na solução de problemas complexos, desempenhando tarefas de classificação de padrões, reconhecimento de voz, reconhecimento de imagem, e controle. Uma vez que as redes perceptron só representam funções linearmente separáveis, um dos motivos do ressurgimento da área de redes neurais foi devido ao desenvolvimento do algoritmo *BackPropagation*. A rede MLP é uma generalização da rede perceptron estudada na seção anterior.

As *ANN* do tipo *MLP* estão constituídas de três partes: a primeira parte é denominada camada de entrada (*input layer*), a qual é constituída de um conjunto de unidades sensoriais; a segunda parte é constituída de uma ou mais camadas escondidas (*hidden layers*); e a terceira parte é constituída de uma camada denominada camada de saída (*ouput layer*). Com exceção da camada de entrada, todas as outras camadas estão constituídas por neurônios, as quais implicam em um esforço computacional.

Na Figura 3.5, apresenta-se a arquitetura de uma rede neural *MLP* com uma camada de entrada constituída de n unidades sensoriais, duas camadas escondidas constituída de i e j neurônios respectivamente, e uma camada de saída constituída de k neurônios, formando a arquitetura *MLP* (n-i-j-k).

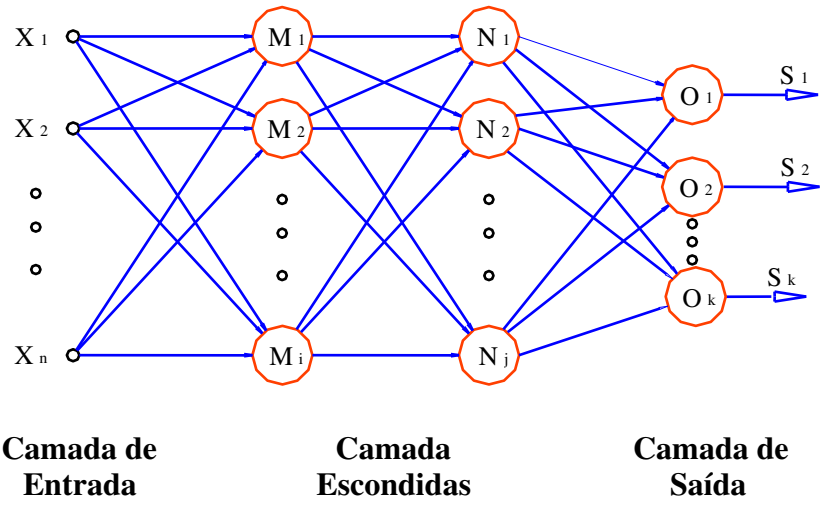

Figura 3.5. Arquitetura de uma rede neural MLP (*n-i-j-k*).

Algumas características básicas das redes *MLP* são as seguintes:

- A função de ativação de cada um dos neurônios da rede é uma *função não-linear*. Uma característica importante desta função é que precisa ser suave e diferenciável em todos os pontos. Geralmente, a função de ativação que satisfaz estes requerimentos é uma *função não-linear sigmoidal*, as mais comuns utilizadas são as *funções Logsig* e *Tansig*.
- As redes *MLP* são constituídas de uma ou mais camadas escondidas, que contém neurônios que não formam parte da camada de entrada ou saída da rede. Estes neurônios escondidos permitem que a rede possa aprender tarefas mais complexas progressivamente, pelo processo de aprendizado.
- Uma rede *MLP* é uma rede *feedforward*, onde o fluxo de dados é sempre em uma única direção. Assim, a saídas dos neurônios de uma camada qualquer se conectam unicamente às entradas dos neurônios da camada seguinte. Portanto, o sinal de entrada se propaga através da rede em um só sentido, isto é, não existe realimentação.
- As redes *MLP* apresentam um alto grau de conectividade ou podem ser completamente conectada, o que é determinado pelas sinapses da rede, caso em que um neurônio de uma camada qualquer é conectado a todos os neurônios da camada seguinte. Além disso, uma rede *MLP* pode ser parcialmente conectada, no caso em que alguma das sinapses esteja faltando. Na prática, a falta de algumas sinapses dentro da rede é representada fazendo o peso das sinapses igual a zero. Entretanto, neste estudo consideram-se redes *MLP* completamente conectadas.

A combinação destas características, junto com a capacidade de aprendizagem, produto da experiência através do processo de treinamento das redes *MLP*, determina o esforço computacional.

O número de variáveis ou nós da camada de entrada é determinado pela complexidade do problema e a dimensionalidade do espaço de observação, de

onde são gerados os sinais de entrada. O número de neurônios na camada de saída é determinado pela dimensionalidade da resposta desejada. Além disso, a configuração e modelagem de uma rede MLP têm como objetivo determinação dos seguintes aspectos:

1. A determinação do número de camadas escondidas.
2. A determinação do número de neurônios em cada uma das camadas escondida.
3. A determinação do valor dos pesos de conexão e as funções de ativação.

Existe uma variedade de metodologias para a escolha destes aspectos, mas normalmente estas são feitas de forma empírica. Na Figura 3.6, apresentam-se estes aspectos.

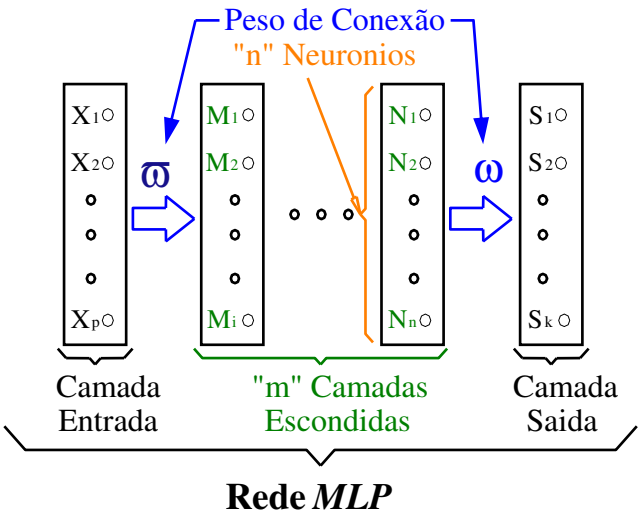

Figura 3.6. Configuração da rede MLP.

O tamanho do modelo de *ANN* é determinado pelos aspectos 1 e 2 acima, mas não há regras para a determinação de tais especificações, sendo geralmente determinados baseado na experiência do especialista. O número de camadas escondidas em uma *ANN* influi na relação entrada-saída. Isto é, uma *ANN* com um maior número de camadas escondidas tem melhor capacidade de extrair as características de algum processo aleatório desconhecido, sobre o qual a rede esta tentando adquirir conhecimento. O incremento excessivo do número de camadas escondidas e neurônios dentro da *ANN* produzem uma rede muito grande que não responderá corretamente a padrões nunca vistos. Isto é, devido à rede extrair muitas características,ela pode tentar, de forma inapropriada, modelar também o

ruído. E um número muito pequeno de camadas escondidas e neurônios produz uma *ANN* muito pequena, que não é capaz de armazenar todos os padrões necessários, a ponto de não modelar fielmente os dados. Além disso, uma rede muito grande demanda um maior esforço computacional, então se deve ter um compromisso entre *Convergência* e *Generalização*.

A *Convergência* é a capacidade da rede de aprender todos os padrões do conjunto de treinamento, a qual esta estritamente relacionada com o tamanho da rede. Entanto, a *Generalização* é a capacidade de um modelo de aprendizado responder corretamente a padrões novos que lhe são apresentados. Um modelo com boa generalização é aquele que responde corretamente a padrões contidos na base de dados, e também a padrões novos contidos na base de teste. A capacidade de generalização é o principal objetivo no processo de aprendizado.

Outro aspecto está relacionado com a determinação da função de ativação de cada um dos neurônios, e do algoritmo de treinamento utilizado para obter os pesos de conexão desejados que resolva o problema ou tarefa. Um dos algoritmos de treino que tem sido aplicado na solução de diversos problemas complexos é o algoritmo conhecido na literatura como *backpropagation*, o qual é baseado na retro-propagação do erro.

A idéia básica do algoritmo *backpropagation* foi descrita por Werbos em sua tese de doutorado na Universidade de *Harvard* em 1974. No entanto, o algoritmo de aprendizado *backpropagation* surgiu após 1985, através da publicação do livro *Parallel Distributed Processing* de Rumelhart e McClelland em 1986.

Além do algoritmo *backpropagation*, existem vários algoritmos de aprendizado, ou também chamados regras de aprendizado, algumas das quais são o algoritmo de aprendizado *Levenberg Marquardt* (utiliza uma aproximação do método de Newton), o algoritmo do gradiente descendente (usado pelo algoritmo *backpropagation*), algoritmos competitivos que se caracterizam pelas conexões laterais dos neurônios com seus vizinhos, estabelecendo uma competição entre os neurônios (usado pelas redes Hopfiel e Kohonen).

A obtenção dos pesos desejados da *ANN* é obtida mediante a atualização de pesos no processo de treinamento, o que pode ser feito de duas maneiras, *Batch* ou Incremental.

No processo de treinamento *Batch ou por ciclos*, a atualização dos pesos acontece somente após a apresentação de todos os padrões do conjunto de treinamento. Assim, todos os padrões são avaliados com a mesma configuração de pesos. Entretanto, no treinamento *Incremental (ou por Padrão)*, a atualização dos pesos é feita após a apresentação de cada novo padrão. Isto leva a que a rede aprenda melhor o último padrão apresentado, e por isso é recomendável apresentar os dados de forma aleatória.

A eficiência dos dois métodos de treinamento depende do problema em questão. O processo de treinamento (aprendizado) é mantido até que os pesos se estabilizem e o erro quadrático médio seja suficientemente pequeno, ou seja, menor que um erro admissível, de modo que o objetivo desejado seja atingido. Assim, deve-se encontrar um ponto ótimo de parada com erro mínimo e máxima capacidade de generalização.

Após o processo de treinamento, algumas vezes a *ANN* pode se especializar demasiadamente em relação aos padrões do conjunto de treinamento, gerando um problema de aprendizado conhecido como super-aprendizado ou *over-fitting*. Geralmente este problema pode ser evitado por meio de um método *Early Stopping*. Este método divide o conjunto de padrões em um novo conjunto de treinamento e validação, após cada varredura do conjunto de treinamento, a rede é avaliada com o conjunto de validação. O algoritmo pára, quando o desempenho com o teste de validação deixa de melhorar.

O procedimento de treinamento utilizado é o treinamento *Supervisionado*, a rede é treinada com um conjunto de dados de treinamento que fornece à rede os valores de entrada e seus respectivos valores de saída desejada, como apresentado na Figura 3.7.

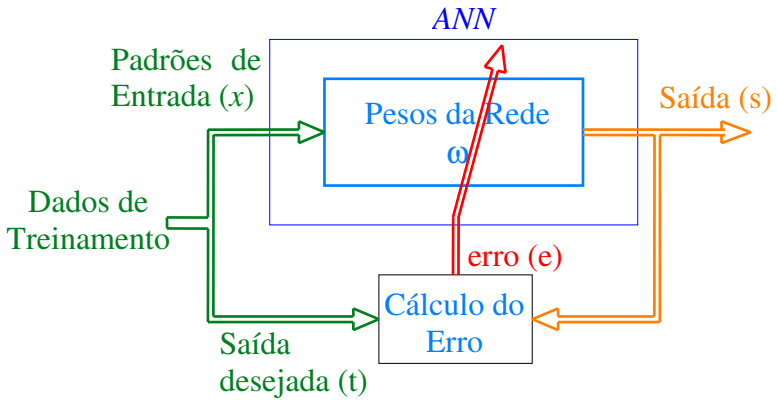

Figura 3.7. Treinamento Supervisionado.

3.4. Algoritmo Backpropagation

Backpropagation é o algoritmo de treinamento mais utilizado pelas *ANN*. Este algoritmo é de tipo supervisionado, e consiste no ajuste dos pesos da rede baseado na minimização do erro médio quadrático pelo algoritmo do gradiente descendente. Portanto, uma parte essencial do algoritmo é o cálculo das derivadas parciais do erro em relação aos parâmetros da *ANN*. O desenvolvimento deste algoritmo permitiu resolver o problema da atualização dos pesos nas camadas escondidas, e com este o ressurgimento das redes neurais.

3.4.1. Fases do Algoritmo *Backpropagation*

O algoritmo é composto de duas fases: *Propagação* e *Retropropagação*, descritos a seguir.

3.4.1.1. Propagação (*forward*)

Nesta fase, um conjunto de padrões é aplicado na camada de entrada da *ANN*, e a resposta desta é propagada como entrada na seguinte camada. Este efeito é propagado através da rede, camada a camada, até que finalmente é produzido um conjunto de saída como a resposta atual da rede. Durante esta etapa, os pesos sinápticos da rede são fixos. Observe na Figura 3.9(b) que o sinal flui através da rede no sentido direito, da esquerda para a direita. Estes sinais que se propagam de forma direta, neurônio a neurônio, através da rede desde a entrada até a saída, são denominados *Sinais Funcionais*. Esta denominação nasceu do fato de estes sinais

serem obtidos na saída de cada neurônio como uma função dos sinais de entrada de cada um. Na Figura 3.9(a), apresenta-se a fase de propagação em uma rede *MLP* (4-3-2) [14].

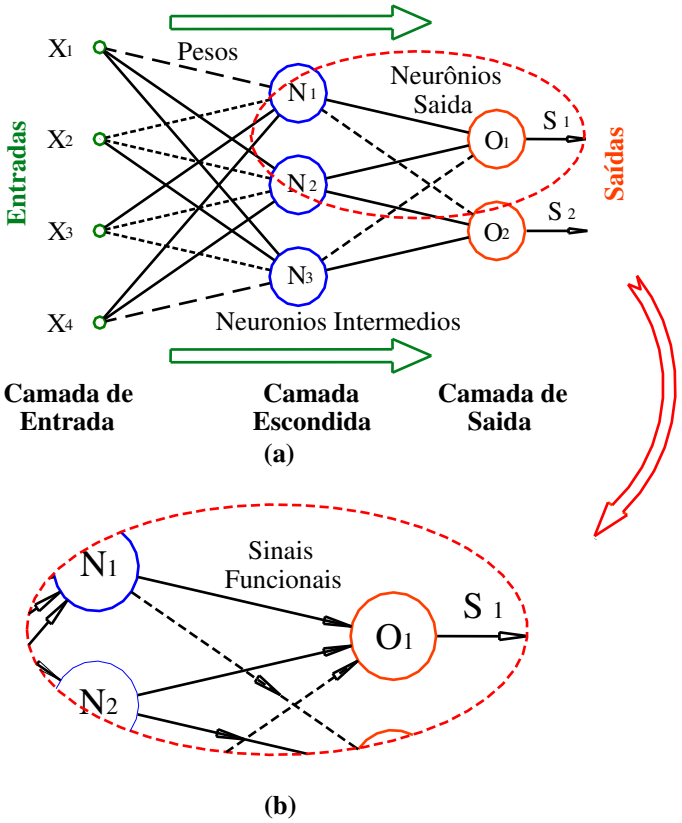

Figura 3.9. (a) Fase de propagação (b) Sinais Funcionais.

3.4.1.2. Retropropagação (*backward*)

Na fase de Retropropagação, desde a camada de saída até a camada de entrada são ajustados todos os pesos sinápticos em concordância com uma correlação do erro. Especificamente, a resposta atual da rede é subtraída de uma resposta desejada (*target*), gerando um sinal erro. Este sinal de erro novamente é retropropagado através da rede no sentido contrario ao sinal funcional, daí o nome de "*error back-propagation*".

Os pesos da *ANN* são ajustados para fazer com que a resposta atual da rede se aproxime à resposta desejada. E o erro originado em um neurônio na camada de saída da rede, que é propagado de volta através da rede, camada a camada, é denominado como sinal de erro. Este sinal é referido como sinal de erro porque o cálculo do erro em cada neurônio da rede é uma função de erro de algum tipo. Na Figura 3.10, apresenta-se a fase de retropropagação e os sinais de erro [14].

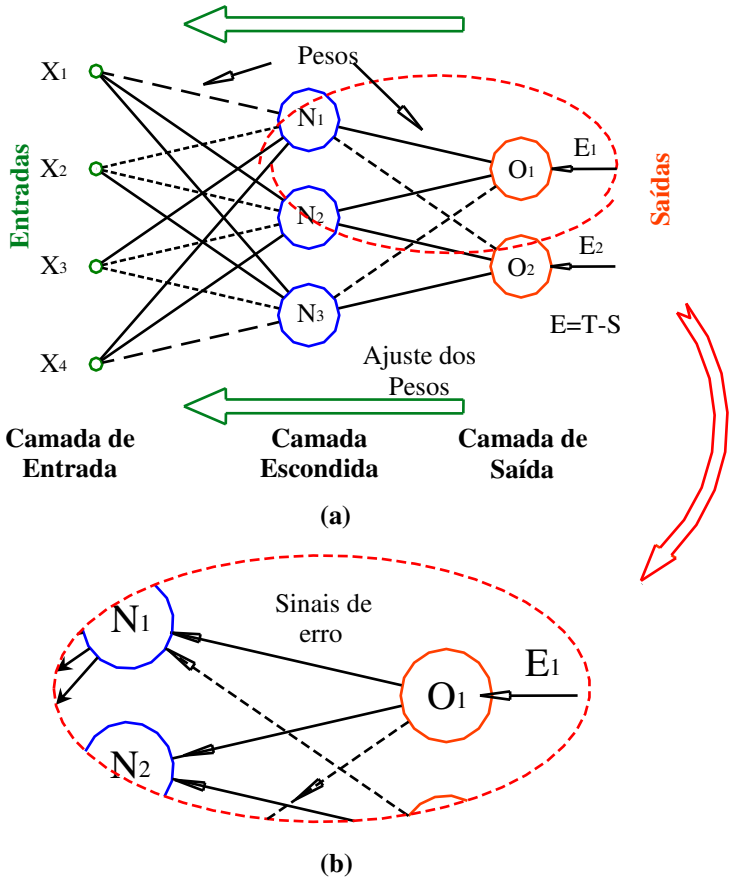

Figura 3.10. (a) Fase de Retro-propagação (b) Sinais de erro.

3.4.2. Algoritmo de Aprendizado

O algoritmo de aprendizado *backpropagation* é baseado na propagação do erro, que inicia na camada de saída e vai para as camadas anteriores da *ANN*, de acordo com o grau de participação que cada neurônio teve no nível posterior. Antes de passar à descrição do algoritmo, é conveniente fazer algumas considerações quanto às características básicas da *ANN*:

- Topologia da rede – Múltiplas camadas (*MLP*).
- Função de ativação – Função não-linear, diferenciável em todo o domínio.
- Tipo de aprendizado – supervisionado.
- Regra de propagação – $Net_j = \sum_{i=1}^{k} \omega_{ji}.x_i + \theta_j$.
- Valores de entrada – saída – binários ou contínuos.

O processo de aprendizado do algoritmo *backpropagation* tem como objetivo a atualização dos pesos sinápticos da rede, o que é feito por meio da minimização do erro quadrático médio pelo método do *Gradiente Descendente*. Baseado neste método, o fator de atualização do peso ω_{ij} relativo à entrada i do neurônio j é dado por

$$\Delta\omega_{ij} = -\eta.\frac{\partial E}{\partial \omega_{ij}} \tag{3.11}$$

onde, $\Delta\omega_{ij}$ é a variação do peso do neurônio j da conexão i, η é a taxa de aprendizado, e E é o somatório do erro quadrático médio total.

O valor do erro quadrático médio total (E) é definido pela Equação (3.12), como a soma do erro quadrático médio de todo os padrões [19].

$$E = \frac{1}{2}.\sum_{p}\sum_{i=1}^{k}(t_i^p - s_i^p)^2 \tag{3.12}$$

onde, p é o numero de padrões, k é o numero de neurônios na saída, e t_i^p e s_i^p são o valor desejado e a saída gerada pela rede para o neurônio i quando é apresentado o padrão p. Além disso, pode-se assumir sem perda de generalidade que a minimização de cada padrão vai a levar à minimização do erro total, de modo que o erro passa a ser definido por

$$E = \frac{1}{2}.\sum_{i=1}^{k}(t_i - s_i)^2 \tag{3.13}$$

3.4.2.1. Calculo da $\Delta\omega_{ij}$

Usando a regra da cadeia na Equação (3.11), pode-se expressar

$$\Delta\omega_{ij} = -\eta.\frac{\partial E}{\partial \omega_{ij}} = -\eta.\frac{\partial E}{\partial net_j}.\frac{\partial net_j}{\partial \omega_{ij}} \tag{3.14}$$

Onde, $net_j = \sum s_i.\omega_{ij} + \theta_j$, e assim

$$\frac{\partial net_j}{\partial \omega_{ij}} = S_i \tag{3.15}$$

onde, s_i é o valor de entrada recebido pela conexão i do neurônio j.

Entretanto, o outro termo da Equação (3.14) é o valor calculado do erro do neurônio j. Este termo depende da camada na qual se encontra o neurônio, e é representado por

$$e_j = -\frac{\partial E}{\partial net_j} \tag{3.16}$$

Assim, das equações acima, pode-se estabelecer que

$$\Delta\omega_{ij} = \eta.s_i.e_j \tag{3.17}$$

Entretanto, o valor do peso atualizado é definido por

$$\omega_{ij}^{t+1} = \omega_{ij}^{t} + \Delta\omega_{ij}^{t+1} = \omega_{ij}^{t} + \eta.s_i.e_j \tag{3.18}$$

onde, ω_{ij}^{t+1} é o valor de peso atualizado, ω_{ij}^{t} é o valor de peso na iteração anterior no processo de aprendizado, e $\Delta\omega_{ij}^{t}$ é a variação do peso.

3.4.2.2. Calculo do Erro (e_j)

Usando a regra da cadeia na Equação (3.16), pode-se estabelecer que:

$$e_j = -\frac{\partial E}{\partial net_j} = -\frac{\partial E}{\partial s_j}.\frac{\partial s_j}{\partial net_j} \tag{3.19}$$

onde $s_j = \varphi(net_j)$, então:

$$\varphi'(net_j) = \frac{\partial s_j}{\partial net_j} \tag{3.20}$$

Entretanto, o valor do termo $\frac{\partial E}{\partial s_j}$ depende da camada à qual pertence o neurônio j.

- **Na camada de saída**

O neurônio j pertence à camada de saída, como é apresentado na Figura 3.11. O valor de E é calculado pela Equação (3.13), de modo que o termo $\frac{\partial E}{\partial s_j}$ é

calculado por

$$\frac{\partial E}{\partial s_j} = -(t_j - s_j) \tag{3.21}$$

Assim, o valor do de e_j para o neurônio j na camada de saída é

$$e_j = (t_j - s_j).\varphi'(net_j) \tag{3.22}$$

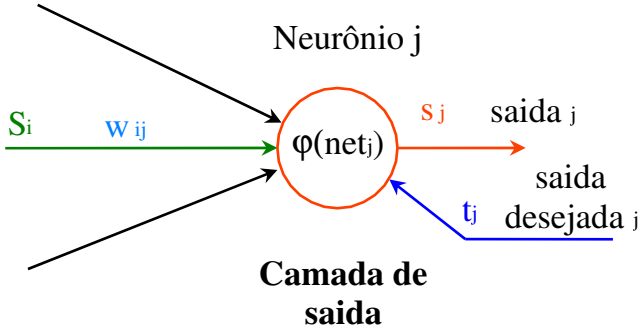

Figura 3.11. Cálculo de erro e_j na camada de saída.

- **Na camada escondida**

O caso em que o neurônio j pertence à camada escondida é apresentado na Figura 3.12.

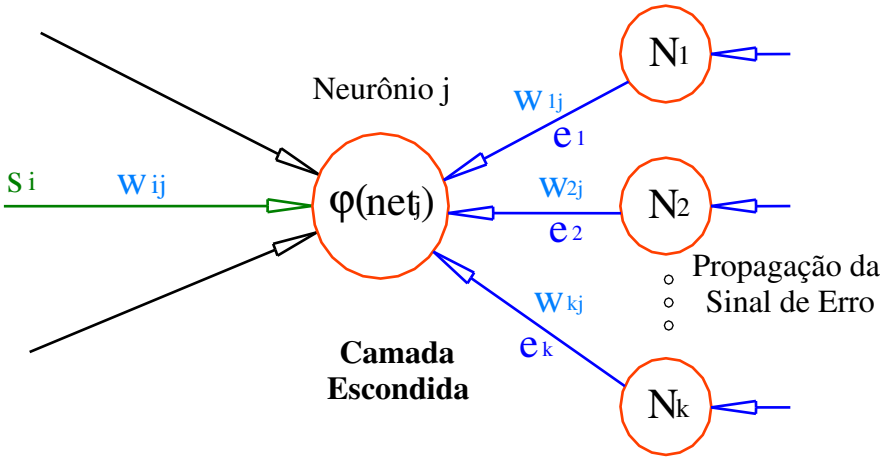

Figura 3.12. Calculo de erro e_j na camada escondida.

O valor de E para este neurônio j é calculado pela Equação (3.13), tomando em consideração a propagação do erro da camada seguinte. O termo $\frac{\partial E}{\partial s_j}$ é calculado por

$$E = \frac{1}{2}.(t_k - s_k)^2$$

$$\frac{\partial E}{\partial s_j} = -\sum_k (t_k - s_k).\frac{\partial s_k}{\partial s_j} \tag{3.23}$$

Usando a regra da cadeia na Equação (3.23), pode-se estabelecer que:

$$\frac{\partial E}{\partial s_j} = \sum_k (t_k - s_k).\frac{\partial s_k}{\partial net_k}.\frac{\partial net_k}{\partial s_j} = -\sum_k (t_k - s_k).\varphi'(net_k).\omega_{jk}$$

$$\frac{\partial E}{\partial s_j} = -\sum_k (e_k.\omega_{jk}) \tag{3.24}$$

O valor de e_j para o neurônio j na camada escondida é então

$$e_j = \sum_k (e_k.\omega_{jk}).\varphi'(net_j) \tag{3.25}$$

3.4.3. Parâmetros de Aprendizado

O algoritmo *backpropagation* faz a aprendizagem da rede através de um processo iterativo de ajuste dos pesos, seguindo uma trajetória de movimento sobre a superfície de erro no espaço de pesos sinápticos, a qual, a cada instante, segue a direção de minimização de erro, podendo chegar ao mínimo global. Alguns parâmetros que influenciam o desempenho do processo de aprendizagem são *Taxa de Aprendizado* e *Termo de Momentum*.

3.4.3.1. Taxa de Aprendizado (η)

A taxa de aprendizado (η) é um parâmetro constante no intervalo [0,1] que interfere na convergência do processo de aprendizado. A influência deste parâmetro esta relacionada à mudança nos pesos sinápticos. Assim, uma taxa de aprendizado muito pequena implica numa trajetória suave e pequenas mudanças nos pesos a cada iteração. No entanto, requer-se um tempo de treinamento muito longo, e dependendo da inicialização dos pesos é possível cair no problema de mínimo local, pois a *ANN* nesse caso não consegue calcular uma mudança nos pesos que faça a rede sair do mínimo local. Na Figura 3.13 apresenta-se o efeito produzido com uma η pequena.

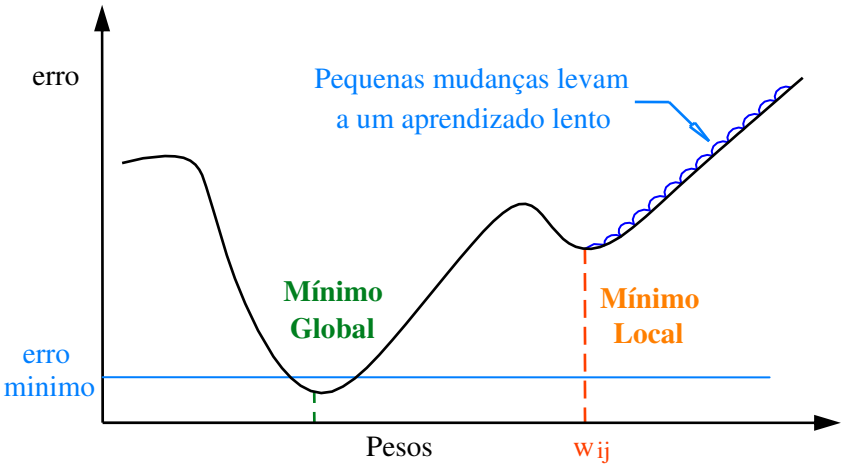

Figura 3.13. Taxa de aprendizado pequeno, com problema do mínimo local.

Entretanto, se a taxa de aprendizado for muito grande (perto de 1), ocorre uma maior mudança nos pesos, aumentando a velocidade do aprendizado, o que pode levar a oscilações em torno do mínimo global. Assim, que a taxa de aprendizado não deve ser muito pequena, nem muito grande. Na Figura 3.14, mostra-se o efeito causado por uma taxa de aprendizado muito grande [19].

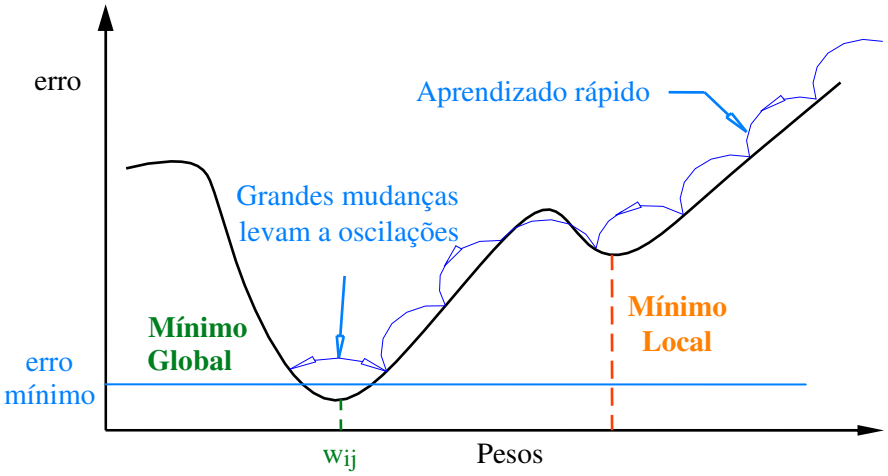

Figura 3.14. Taxa de aprendizado grande, com problema de oscilações.

O ideal seria utilizar a maior taxa de aprendizado possível que não levasse a uma oscilação, obtendo o aprendizado mais rápido. De acordo com BEALE [20], a taxa de aprendizado (η) deveria ser decrementada progressivamente, de modo que o gradiente descendente pelo qual são atualizados os pesos possa encontrar uma melhor solução. Desta forma, utiliza-se uma taxa de aprendizado (η)

adaptativa, a qual inicia com um valor grande, e que decresce exponencialmente à medida que o treinamento evolui.

3.4.3.2. Termo de Momentum (α)

Uma maneira de aumentar a taxa de aprendizado sem levar à oscilação durante a execução do algoritmo *backpropagation* está na à modificação da Equação (3.17) para incluir o termo momentum, o qual determina as mudanças passadas dos pesos. Deste modo, o termo de momentum leva em consideração o efeito das mudanças anteriores dos pesos na direção do movimento atual dos pesos. A mudança dos pesos tomando em consideração o efeito do termo de momentum é determinada por

$$\Delta\omega_{ij}^{t+1} = \eta.s_i.e_j + \alpha.\Delta\omega_{ij}^{t} \tag{3.26}$$

onde, $\Delta\omega_{ij}^{t+1}$ e $\Delta\omega_{ij}^{t}$ são a variação do peso do neurônio j em relação à conexão i no instante $t+1$ e t respectivamente, η é a taxa de aprendizado, e α é o termo de momentum. Na Figura 3.15, apresenta-se o efeito do termo de momentum na aprendizagem da rede [14], [19].

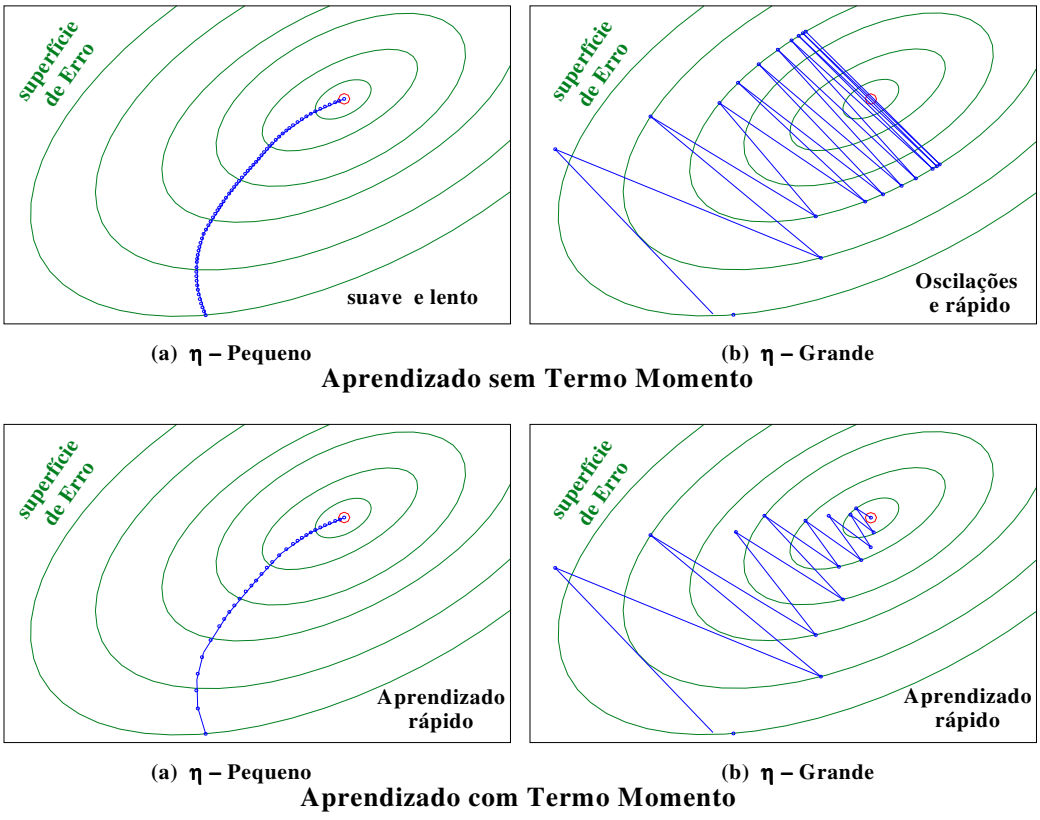

Figura 3.15. Aprendizado com Termo de Momentum.

3.5.Modelagem da *ANN*

Alguns fatores importantes para a modelagem da *ANN* que influem no desempenho das redes são:

- **Seleção de variáveis**

Na maioria das aplicações, as bases de dados contêm um grande número de atributos ou variáveis, os quais são introduzidos com o objetivo de obter um melhor desempenho. Entretanto, na maioria dos casos, muito destes dados são redundantes ou irrelevantes. Portanto, o problema está na seleção das características ou variáveis mais relevantes dentre todos os atributos da base de dados. Neste sentido, alguns dos algoritmos que possuem a capacidade de escolha de variáveis em um menor tempo e custo computacional são a correlação cruzada, a auto-correlação, o estimador por mínimos quadrados (*LSM*), e SIE (*Single Input Effectiveness*).

- **Limpeza de dados**

Geralmente, na etapa de obtenção da base de dados, suas transformações podem levar a conter valores incorretos, ausência de dados, e inconsistências. Deste modo, a limpeza de dados é importante no desempenho do modelo. Este problema é superado pelo conhecimento dos limites dos dados, a identificação de "*outliers*" baseado na visualização, e o uso de informações estatísticas para estabelecer como neutros os dados ausentes ou incorretos.

- **Representação das variáveis**

Dependendo do tipo de dado e do comportamento, as variáveis podem ser codificadas para obter um melhor aprendizado. Assim, os dados podem ser representados como discretos ou contínuos. A representação discreta transforma os dados em uma representação.

- **Normalização**

A normalização torna-se importante quando existem dados muito dispersos no domínio da variável. Nesses casos, valores muito altos podem saturar a função de ativação. Assim, uma maneira fácil de normalizar os dados consiste em somar todas as características e dividir pela soma. Outros tipos de normalização consistem na divisão pela faixa máxima de valores, ou também subtrair do valor médio e dividir pelo desvio padrão.

- **Generalização**

Um aspecto bastante importante ao longo do processo de treinamento é garantir que a rede tenha uma boa generalização. O método mais comum de se garantir uma boa generalização consiste em reservar uma parte da base de dados para validar a generalização.

3.6. Vantagens e Desvantagens das *ANN*

Algumas das principais vantagens e desvantagens, apresentada em aplicações praticas, pelas *ANN* são as seguintes:

*3.6.1.*Vantagens das *ANN*

- A capacidade de lidar com informações incompletas e ruidosas, verificando-se a capacidade da rede em fornecer uma saída satisfatória.
- Não é necessária informação sobre o ambiente a *priori*, pois o aprendizado é feito através da apresentação de padrões à rede.
- Processamento paralelo.
- Habilidade de aprender por meio de exemplos, e a capacidade de generalização a partir dos padrões de treinamento, fornecendo saídas satisfatórias a padrões novos que não foram vistos durante o treinamento.

3.6.2.Desvantagens das *ANN*

- A dificuldade de justificar o comportamento das *ANN* em determinadas situações, isto pois são consideradas como “caixas pretas”, nas quais não se sabe por que a rede chega a um determinado resultado [21].
- Nas *ANN* que utilizam o algoritmo de *backpropagation* o tempo de treinamento tende a ser muito longo. Esta é uma limitação para arquiteturas muito grande ou grande quantidade de dados de treinamento.
- A dificuldade em determinar a arquitetura ideal da *ANN*, de modo que ela seja grande o suficiente para conseguir resolver o problema, e ao mesmo tempo pequeno o suficiente para apresentar um treinamento rápido [22].
- Dependendo do tipo de aprendizagem, outra desvantagem nas *ANN* é a necessidade de ter uma base de dados para o processo de treinamento.