Capítulo 4

Multilayer Perceptrons

As redes *Multilayer Perceptron* (MLPs) têm sido aplicadas com sucesso em uma variedade de áreas, desempenhando tarefas tais como: classificação de padrões (reconhecimento), controle e processamento de sinais.

Uma RNA do tipo MLP é constituída por um conjunto de nós fonte, os quais formam a camada de entrada da rede (*input layer*), uma ou mais camadas escondidas (*hidden layers*) e uma camada de saída (*output layer*). Com exceção da camada de entrada, todas as outras camadas são constituídas por neurônios e, portanto, apresentam capacidade computacional. O MLP é uma generalização do Perceptron que estudamos no Capítulo 3.

A Figura 4.1 mostra a arquitetura de uma rede neural MLP com uma camada de entrada, 2 camadas escondidas e uma camada de saída.

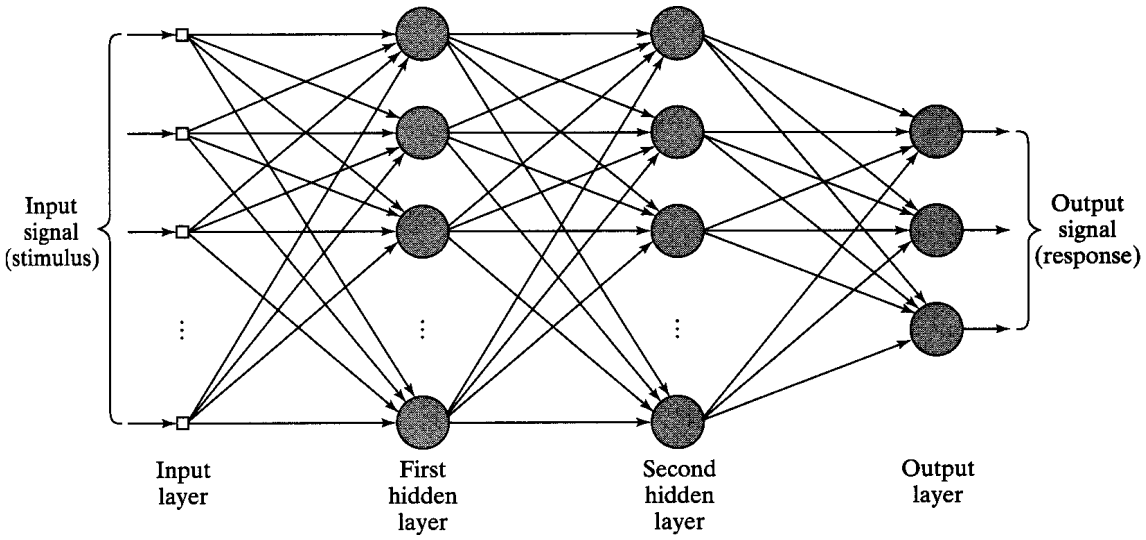

Figura 4.1: Arquitetura de uma rede neural *multilayer perceptron* com duas camadas escondidas.

Duas características de tal estrutura são imediatamente aparentes:

1. Uma rede *multilayer perceptron* é uma rede progressiva. Conforme estudamos no Capítulo 1, uma RNA é dita progressiva (*feedforward*) quando as saídas dos neurônios em qualquer particular camada se conectam unicamente às entradas dos neurônios da camada seguinte, sem a presença de laços de realimentação. Conseqüentemente, o sinal de entrada se propaga através da rede, camada a camada, em um sentido progressivo.

2. A rede pode ser completamente conectada, caso em que cada nó (computacional ou não) em uma camada é conectado a todos os outros nós da camada adjacente. De forma alternativa, uma rede MLP pode ser parcialmente conectada, caso em que algumas sinapses poderão estar faltando. Redes localmente conectadas representam um tipo importante de redes parcialmente conectadas. O termo "local" se refere à conectividade de um neurônio em uma camada da rede com relação a somente um sub-conjunto de todas as possíveis entradas. Na prática, a falta de uma determinada sinapse em um MLP é emulada fazendo-se sua transmitância constante e igual a zero. Neste estudo, no entanto, consideraremos apenas MLPs completamente conectados.

O número de nós fonte na camada de entrada da rede é determinado pela dimensionalidade do espaço de observação, que é responsável pela geração dos sinais de entrada. O número de neurônios na camada de saída é determinado pela dimensionalidade requerida da resposta desejada. Assim, o projeto de uma rede MLP requer a consideração de três aspectos:

I A determinação do número de camadas escondidas;

II A determinação do número de neurônios em cada uma das camadas escondidas;

III A especificação dos pesos sinápticos que interconectam os neurônios nas diferentes camadas da rede.

Os aspectos I e II determinam a complexidade do modelo de RNA escolhido e, infelizmente, não há regras determinadas para tal especificação. A função das camadas escondidas em uma RNA é a de influir na relação entrada-saída da rede de uma forma ampla. Uma RNA com uma ou mais camadas escondidas é apta a extrair as estatísticas de ordem superior de algum desconhecido processo aleatório subjacente, responsável pelo "comportamento" dos dados de entrada, processo sobre o qual a rede está tentando adquirir conhecimento. A RNA adquire uma perspectiva global do processo aleatório, apesar de sua conectividade local, em virtude do conjunto adicional de pesos sinápticos e da dimensão adicional de interações neurais proporcionada pelas camadas escondidas.

O aspecto III envolve a utilização de algoritmos de treino supervisionados. As RNAs MLPs têm sido aplicadas na solução de diversos e difíceis problemas através da utilização de tais algoritmos. O algoritmo de treino quase universalmente utilizado para tanto é o algoritmo de retro-propagação do erro, conhecido na literatura como *Backpropagation Algorithm* ou simplesmente *Backprop*.

O algoritmo *backpropagation* baseia-se na heurística do aprendizado por correção de erro (em que o erro é retro-propagado da camada de saída para as camadas intermediárias da RNA). Este algoritmo pode ser visto como uma generalização do Algoritmo *Least Mean Square* (LMS) desenvolvido por Bernard Widrow, que estudamos para o caso especial de um único neurônio linear, no Capítulo 3.

O termo *backpropagation* surgiu após 1985. No entanto, a idéia básica foi primeiramente descrita por Werbos em sua tese de doutorado em 1974. Em 1986, foi redescoberto por Rumelhart, Hinton e Williams e popularizado através da publicação do livro *Parallel Distributed Processing* de Rumelhart e McClelland em 1986.

O desenvolvimento do *backpropagation* representa um marco fundamental em redes neurais, pois é um método computacionalmente eficiente para o treinamento de redes MLPs e por ter resolvido o problema de realizar a propagação reversa do erro em RNAs com múltiplas camadas, problema este que atrasou por muitos anos o desenvolvimento da área de redes neurais artificiais.

Basicamente, o algoritmo *backpropagation* consiste de dois passos através das diferentes camadas do MLP: um passo direto e um passo reverso.

No passo direto um padrão de atividade do processo a ser aprendido (ou vetor de entrada) é aplicado aos nós de entrada do MLP e o seu efeito se propaga através da rede, camada por camada, produzindo na camada de saída a resposta do MLP à excitação aplicada (vetor de saída). Durante o passo direto os pesos sinápticos são todos fixos.

Durante o passo reverso os pesos sinápticos são todos ajustados de acordo com a regra de aprendizado por correção de erro. Especificamente, a resposta do MLP à excitação é subtraída de um padrão de resposta desejado para aquela excitação aplicada, de forma a produzir um sinal de erro, de forma semelhante ao algoritmo LMS. Este sinal de erro é, então, propagado de volta através dos mesmos neurônios utilizados no passo direto, mas no caminho contrário do fluxo de sinal nas conexões sinápticas - daí o nome *backpropagation.* Os pesos sinápticos são, então, ajustados de forma que a resposta obtida do MLP aproxime-se mais do padrão de resposta desejado.

Uma rede MLP apresenta três características distintas, de cuja combinação com a habilidade de aprender através da experiência (através do treinamento), deriva sua capacidade computacional:

1. O modelo de cada neurônio do MLP inclui uma função de ativação não-linear. É importante salientar que esta não-linearidade é suave (ou seja, a função é diferenciável em qualquer ponto), ao contrário da função utilizada no modelo do Perceptron de Rosenblatt (função signum). Uma forma comumente utilizada de não-linearidade que satisfaz este requisito é a não-linearidade sigmoidal definida pela função logística:

$$y_j = \frac{1}{1+\exp(-v_j)} \tag{4.1}$$

onde v_j é o potencial de ativação (isto é, a soma ponderada de todas as entradas sinápticas mais a polarização) do neurônio j, e y_j é a saída do neurônio.

2. O MLP contém uma ou mais camadas de neurônios escondidos que não são parte da camada de entrada ou da camada de saída da rede. Estes neurônios escondidos possibilitam que a rede aprenda tarefas complexas, extraindo progressivamente mais características significativas dos padrões de entrada (vetores de entrada).
3. A rede MLP exibe um alto grau de conectividade, determinado pelas sinapses da rede. Uma mudança na conectividade da rede requer uma mudança na população de conexões sinápticas, ou pesos sinápticos.

Estas mesmas características, entretanto, são também responsáveis pelas dificuldades encontradas na análise de tais redes. Por exemplo, a presença das não-linearidades distribuídas e a alta conectividade tornam difícil a análise teórica das redes MLPs. Em uma rede MLP, o conhecimento aprendido sobre o ambiente é representado pelos valores assumidos pelos pesos sinápticos da rede. A natureza distribuída deste conhecimento ao longo da rede a torna de difícil interpretação. Além disso, o uso de neurônios escondidos torna o processo de aprendizado mais difícil de ser "visualizado" na estrutura da rede.

Observe, na Figura 4.1 que o sinal flui através da rede MLP no sentido direto, da esquerda para a direita e de camada a camada. A Figura 4.2 apresenta um detalhe parcial de uma rede MLP. Dois tipos de sinais são identificados nesta rede:

1. Sinais funcionais: São estímulos que chegam aos nós de entrada da rede, se propagam de forma direta (neurônio a neurônio) através da rede e emergem da camada de saída da rede como sinais de saída. Cada neurônio de um MLP tem aplicado às suas entradas um conjunto de sinais funcionais que gera um sinal funcional na saída do respectivo neurônio . Na camada de entrada de um MLP o conjunto de sinais funcionais aplicado a cada neurônio é o próprio conjunto de sinais de entrada (vetor de entrada). A denominação sinal funcional decorre do fato de que estes sinais são obtidos na saída de cada neurônio como uma função dos sinais de entrada do respectivo neurônio.

2. Sinais de Erro: Um sinal de erro se origina em um neurônio de saída da rede MLP e se propaga de volta (camada a camada) através da rede. Este sinal é referido como sinal de erro porque seu cálculo, a cada neurônio da rede, envolve algum tipo de função de erro.

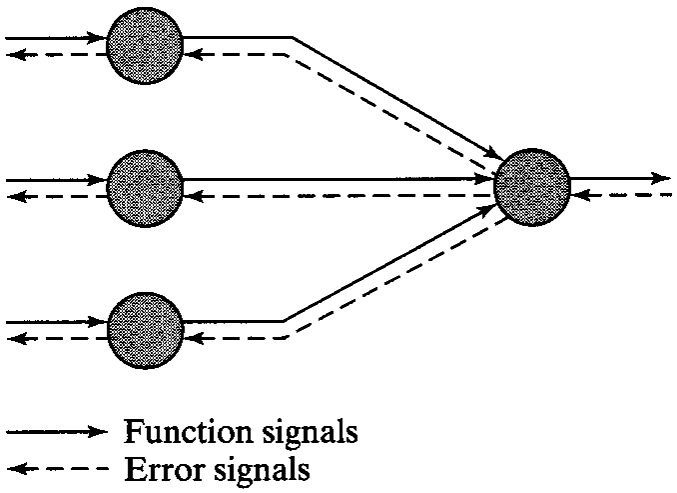

Figura 4.2: Ilustração das direções dos dois fluxos básicos de sinal em uma rede *multilayer perceptron*: propagação direta dos sinais e retro-propagação dos sinais de erro.

Cada neurônio de cada camada escondida ou da camada de saída de uma RNA MLP desempenha duas operações computacionais:

1. A computação do sinal funcional na saída de cada neurônio, o qual é expresso como uma função contínua não-linear do sinal funcional de entrada e dos pesos sinápticos associados com aquele neurônio.

2. A computação de uma estimativa do vetor gradiente (isto é, os gradientes da superfície de erro com respeito aos pesos conectados às entradas de um neurônio), cálculo este que é necessário para o passo reverso através da rede MLP.

4.1 O Algoritmo *Backpropagation*

Assim como o algoritmo LMS é considerado o mais renomado dos algoritmos utilizados em filtragem linear adaptativa, o algoritmo *backpropagation* foi estabelecido como o mais popular algoritmo utilizado no contexto do aprendizado de RNAs MLP.

A popularidade do algoritmo *backpropagation* resulta de sua relativa simplicidade de implementação e do fato de ser um poderoso dispositivo para armazenar o conteúdo de informação (adquirido pela rede MLP a partir do conjunto de dados) nos pesos sinápticos da rede.

Na medida em que o conjunto de dados usado para treinar uma RNA MLP seja grande o suficiente para ser representativo do ambiente no qual a rede está inserida, a rede MLP treinada através do algoritmo *backpropagation* desenvolverá a capacidade de generalizar. Especificamente, esta capacidade permite à rede MLP apresentar um desempenho satisfatório quando é alimentada com dados de teste retirados do mesmo espaço de entrada que os dados de treino, mas não previamente apresentados ao MLP.

Antes de passarmos à descrição do algoritmo *backpropagation*, é conveniente fazermos algumas considerações quanto à notação que será utilizada:

- Os índices i, j e k se referem a diferentes neurônios no MLP. Os sinais funcionais se propagam através da rede, da esquerda para a direita, sendo que o neurônio j está na camada à direita do neurônio i, e o neurônio k está na camada à direita do neurônio j, quando o neurônio j é uma unidade escondida.
- Na iteração n, o n-ésimo padrão de treino (vetor-exemplo) é apresentado ao MLP.
- O símbolo $\varepsilon(n)$ se refere à soma instantânea dos erros quadráticos nos nós de saída do MLP (ou energia do erro) na iteração n. A média de $\varepsilon(n)$ sobre todos os valores de n (isto é, o conjunto de treino inteiro) representa a energia média do erro ε_{av}.
- O símbolo $e_j(n)$ se refere ao sinal de erro na saída do neurônio j para a iteração n.

- O símbolo $d_j(n)$ se refere à resposta desejada para o neurônio *j* e é usado para computar $e_j(n)$.
- O símbolo $y_j(n)$ se refere ao sinal funcional encontrado na saída do neurônio *j*, na iteração *n*.
- O símbolo $w_{ji}(n)$ denota o peso sináptico que conecta a saída do neurônio *i* à entrada do neurônio *j*, na iteração *n*. A correção aplicada a este peso na iteração *n* é denotada por $\Delta w_{ji}(n)$.
- O potencial de ativação (isto é, a soma ponderada de todas as entradas sinápticas mais a polarização) do neurônio *j* na iteração *n* é denotado por $v_j(n)$ e constitui o sinal aplicado à função de ativação associada ao neurônio *j*.
- A função de ativação que descreve a relação funcional entrada-saída da não-linearidade associada ao neurônio *j* é denotada por $\varphi_j(\cdot)$.
- A polarização aplicada ao neurônio *j* é denotada por b_j; seu efeito é representado por uma sinapse de peso $w_{j0} = b_j$ conectada a uma entrada fixa igual a (+1). Alternativamente, a polarização pode ser gerada por uma sinapse de peso $w_{j0} = \theta_j$ conectada a uma entrada de valor fixo e igual a (–1), quando recebe o nome de *threshold*. A nível de operação do MLP, para todos os fins práticos as duas alternativas apresentam os mesmos resultados. Neste estudo consideraremos apenas o nome genérico "polarização", a qual pode ser originada de um valor fixo positivo (+1) ou negativo (-1).
- O *i*-ésimo componente do vetor de entrada do MLP é denotado por $x_i(n)$.
- O *k*-ésimo componente do vetor de saída do MLP é denotado por $o_k(n)$.
- O parâmetro razão de aprendizado é denotado por η .

Tendo estabelecido a notação, inicialmente apenas descreveremos as equações de definição do algoritmo *backpropagation* e sua forma de operação. Posteriormente, deduziremos as equações que regem sua operação.

Seja o sinal de erro na saída do neurônio *j* da camada de saída na iteração *n* (isto é, na apresentação do *n*-ésimo vetor de treinamento) definido por

$$e_j(n) = d_j(n) - y_j(n) \tag{4.2}$$

Define-se o valor instantâneo do erro quadrático para o neurônio *j* como $\frac{1}{2}e_j^{\,2}(n)$.

Correspondentemente, o valor instantâneo da soma dos erros quadráticos $\varepsilon(n)$ é obtida somando $\frac{1}{2}e_j^{\,2}(n)$ sobre todos os neurônios da camada de saída. Estes são os únicos neurônios "visíveis" para os quais os sinais de erro podem ser calculados de forma direta. A soma instantânea dos erros quadráticos na camada de saída do MLP é então escrita como

$$\varepsilon(n) = \frac{1}{2}\sum_{j\in C} e_j^{\,2}(n) \tag{4.3}$$

onde o conjunto C inclui todos os neurônios na camada de saída. Seja N o número total de padrões (vetores-exemplo) contidos no conjunto de treino. O erro médio quadrático (MSE) é obtido somando $\varepsilon(n)$ sobre todo *n* e então normalizando com respeito ao tamanho *N* do conjunto de treino, conforme

$$\varepsilon_{av} = \frac{1}{N-1}\sum_{n=0}^{N-1}\varepsilon(n) \tag{4.4}$$

O valor instantâneo da soma dos erros quadráticos $\varepsilon(n)$, e portanto o MSE denotado por ε_{av}, é função de todos os parâmetros livres (isto é, pesos sinápticos e níveis de polarização) do MLP. Para um dado conjunto de treino, ε_{av} representa a Função de Custo do processo de minimização do erro de aprendizado, constituindo uma medida inversa do desempenho do processo de aprendizado a partir do conjunto de treino. Para minimizar ε_{av} os pesos sinápticos são atualizados a cada apresentação *n* de um novo padrão ao MLP através do vetor de entrada até o término de uma **Época**. Uma Época consiste no intervalo correspondente à apresentação de todos os N vetores-exemplo do conjunto de treino à camada de entrada do MLP. O ajuste dos pesos é feito de acordo com os respectivos erros computados para cada padrão apresentado ao MLP. A média aritmética destas alterações

individuais nos pesos sobre o conjunto de treino é portanto uma estimativa da verdadeira alteração que resultaria a partir da alteração de pesos baseada na minimização da função custo ε_{av} sobre todo conjunto de treino.

Considere a Figura 4.3, a qual descreve o neurônio *j* sendo alimentado por um conjunto de sinais produzidos na saída dos neurônios da camada à sua esquerda.

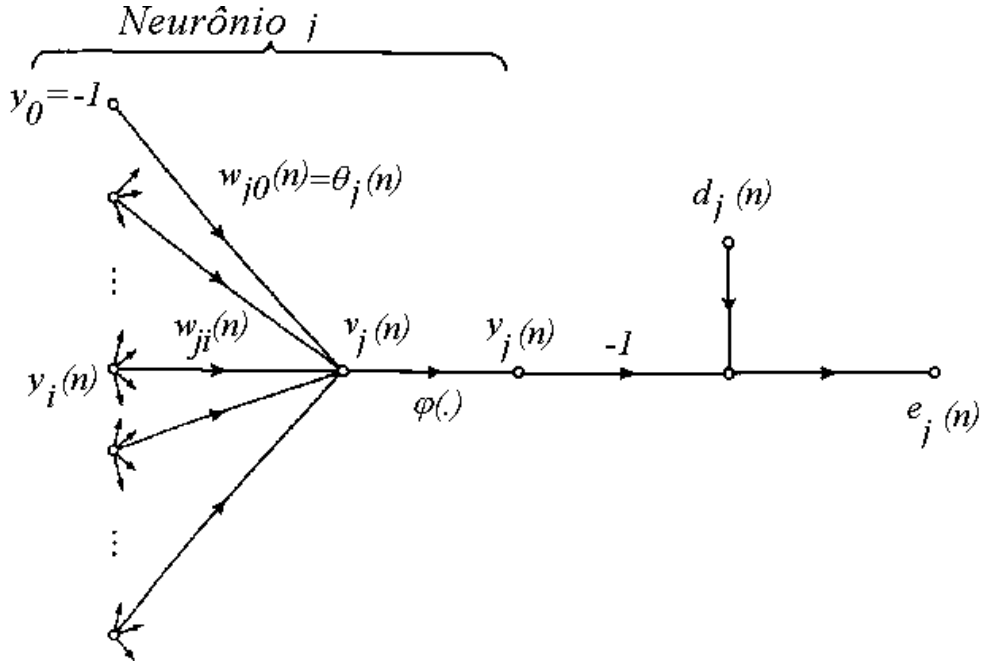

Figura 4.3: Grafo de fluxo de sinal no neurônio *j*.

O potencial de ativação $v_j(n)$ aplicado na entrada da não-linearidade associada ao neurônio *j* é, portanto

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{4.5}$$

onde *m* é o número total de entradas (excluindo a polarização) aplicadas ao neurônio *j*. O peso sináptico w_{j0} (correspondente à entrada fixa $y_0 = -1$) define a polarização θ_j aplicada ao neurônio *j*. $w_{ji}(n)$ é o peso sináptico conectando a saída do neurônio *i* ao neurônio *j* e

$y_i(n)$ é o sinal no *i*-ésimo nó de entrada do neurônio *j*, ou equivalentemente, o sinal na saída do neurônio *i*. Portanto o sinal $y_j(n)$ resultante na saída do neurônio *j* na iteração *n* é:

$$y_j(n) = \varphi_j\left(v_j(n)\right) \tag{4.6}$$

De maneira similar ao algoritmo LMS, o algoritmo *backpropagation* aplica a correção $\Delta w_{ji}(n)$ ao peso sináptico $w_{ji}(n)$, tendo como base a direção contrária do gradiente local da superfície de erro $\varepsilon(w)$ relativo ao peso sináptico.

Se, para uma dada variação no peso sináptico, o algoritmo movimenta-se em uma trajetória ascendente na superfície $\varepsilon(w)$, então significa que esta variação deve ser aplicada com o sinal invertido sobre o peso sináptico, já que houve um aumento do erro, e objetiva-se uma diminuição do erro.

Por outro lado, se para uma dada variação no peso sináptico o algoritmo movimenta-se em uma trajetória descendente na superfície $\varepsilon(w)$, então significa que esta variação deve ser aplicada com o sinal positivo sobre o peso sináptico, já que houve uma diminuição do erro e, portanto, o movimento deve ser encorajado naquela direção.

Este método de correção dos pesos sinápticos é denominado de Regra Delta. No algoritmo LMS, estudado em capítulo anterior, a Regra Delta é definida pela já conhecida expressão $\Delta\underline{w}(n) = -\eta\underline{\nabla}\mathrm{J}(\underline{w}(n))$, onde $\underline{\nabla}\mathrm{J}(\underline{w}(n)) = \dfrac{\partial \mathrm{J}(\underline{w}(n))}{\partial \underline{w}(n)} = \dfrac{\partial\left\{\frac{1}{2}e^2(n)\right\}}{\partial \underline{w}(n)}$ é o gradiente local da superfície de erro gerada pela função de custo $\mathrm{J} = \mathrm{J}(\underline{w}(n)) = \frac{1}{2}e^2(n)$ a ser minimizada no instante *n*.

No caso do MLP, o gradiente local da superfície de erro $\varepsilon(w)$ relativo ao peso sináptico w_{ji} representa, portanto, um fator de sensibilidade, determinando a direção de movimento no espaço de pesos sinápticos para o valor do peso sináptico w_{ji} que minimiza $\varepsilon(w)$.

A correção $\Delta w_{ji}(n)$ aplicada a $w_{ji}(n)$, ditada pela Regra Delta, é definida por

$$\Delta w_{ji}(n) = w_{ji}(n+1) - w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{4.7}$$

onde η é a constante que determina a razão de aprendizado do algoritmo *backpropagation*. O uso do sinal negativo em (4.7) impõe a movimentação contrária à direção apontada pelo gradiente na superfície de erro definida no espaço de pesos sinápticos.

O algoritmo *backpropagation* estabelece (ver Seção 4.1.5 para a demonstração das equações que seguem) o aprendizado de um MLP através da Regra Delta como sendo a correção efetuada em suas sinapses através de

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{4.8}$$

onde $\Delta w_{ji}(n)$ é a correção aplicada à *i*-ésima sinapse do neurônio *j*, $y_i(n)$ é o sinal de entrada no *i*-ésimo nó de entrada do neurônio *j* (= sinal na saída do neurônio *i*, pertencente à camada à esquerda da que pertence o neurônio *j*, se este não estiver na primeira camada escondida – se o neurônio *j* estiver na primeira camada escondida então $y_i(n)$ corresponde ao *i*-ésimo nó de entrada $x_i(n)$ do MLP) e $\delta_j(n)$ é o gradiente local do neurônio *j*, definido por

$$\delta_j(n) = \begin{cases} \varphi'_j(v_j(n)) e_j(n) & \text{, neurônio } j \text{ é de saída} \quad (4.9a) \\ \varphi'_j(v_j(n)) \sum_k \delta_k(n) w_{kj}(\mathrm{n}) & \text{, neurônio } j \text{ é escondido} \quad (4.9b) \end{cases}$$

De acordo com (4.9a) o gradiente local $\delta_j(n)$ para o neurônio de saída j é igual ao produto do correspondente sinal de erro $e_j(n)$ pela derivada $\varphi'_j(v_j(n))$ da função de ativação associada. Neste caso o fator chave necessário envolvido no cálculo do ajuste dos pesos $\Delta w_{ji}(n)$ é o sinal de erro $e_j(n)$ na saída do neurônio *j*.

Quando o neurônio *j* está localizado em uma camada escondida, conforme mostra a Figura 4.4, mesmo não sendo diretamente acessíveis, tais neurônios dividem a responsabilidade pelo erro resultante na camada de saída. A questão, no entanto, é saber como penalizar ou recompensar os pesos sinápticos de tais neurônios pela sua parcela de responsabilidade, já que não existe resposta desejada especificada neste local do MLP e, portanto, não há como calcular o sinal de erro.

A solução, dada pela equação (4.9b), é computar o sinal de erro recursivamente para o neurônio escondido *j* retro-propagando os sinais de erro de todos os neurônios à direita do neurônio *j* aos quais a saída deste encontra-se conectado.

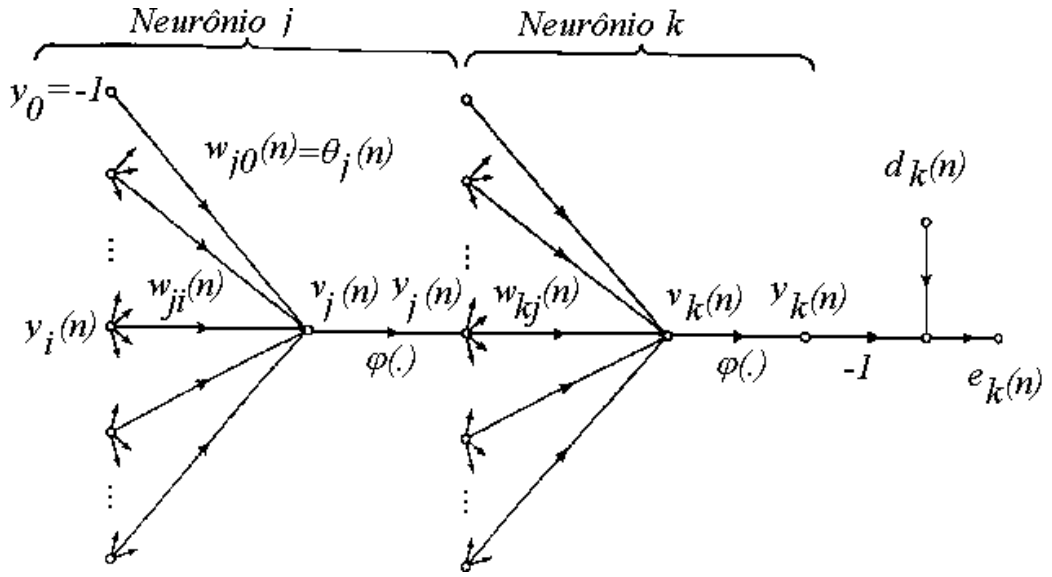

Figura 4.4: Grafo de fluxo de sinal mostrando os detalhes do neurônio de saída *k* comectado ao neurônio escondido *j*.

O fator $\varphi_j'(v_j(n))$ envolvido na computação do gradiente local $\delta_j(n)$ na equação (4.9b) depende somente da função de ativação associada com o neurônio escondido *j*. Os demais fatores envolvidos no somatório sobre k em (4.9b) dependem de dois conjuntos de termos. O primeiro, $\delta_k(n)$, requer conhecimento dos sinais de erro $e_k(n)$ recursivamente retro-propagados, conforme veremos adiante, a partir de todos aqueles neurônios localizados na camada imediatamente à direita do neurônio escondido *j* e que estão

diretamente conectados a ele (ver Figura 4.4). O segundo conjunto de termos, $w_{kj}(n)$, consiste dos pesos sinápticos dos neurônios à direita do neurônio *j* e que com ele estabelecem conexão.

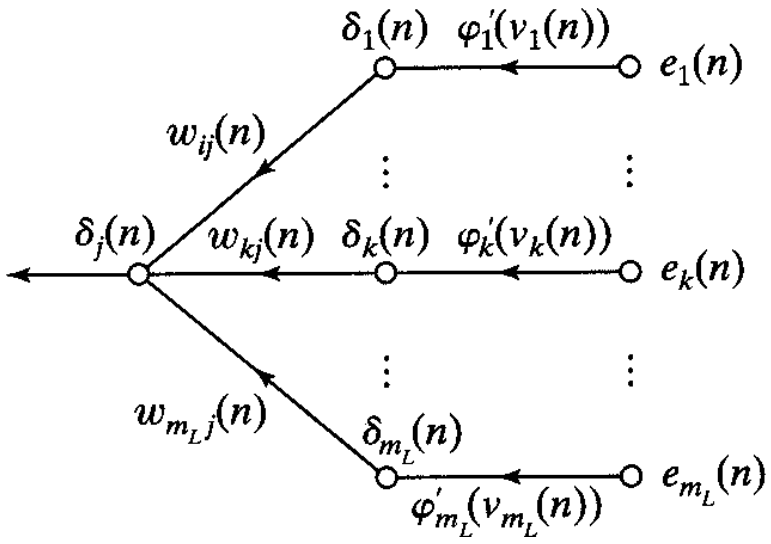

Figura 4.5: Grafo de fluxo de sinal mostrando o processo de retro-propagação dos sinais de erro na camada de saída para um neurônio *j* da camada escondida imediatamente à esquerda. m_L é o número de neurônios da camada de saída.

4.1.1 Os Dois Passos Computacionais do Algoritmo *Backpropagation*

Na aplicação do algoritmo *backpropagation*, dois passos computacionais distintos podem ser identificados, um passo direto e um passo reverso.

No passo direto (*forward pass)* os pesos sinápticos permanecem inalterados em todo MLP e os sinais são propagados da entrada da rede para a saída, de neurônio a neurônio.

O sinal que resulta na saída do neurônio *j* é computado por

$$y_j(n) = \varphi(v_j(n)) \tag{4.10}$$

onde $v_j(n)$ é o potencial de ativação do neurônio *j*, definido por

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \tag{4.11}$$

sendo *m* o número total de entradas (excluindo a polarização) aplicadas ao neurônio *j*, $w_{ji}(n)$ é o peso sináptico conectando a saída do neurônio *i* ao neurônio *j* e $y_i(n)$ é o sinal de entrada do neurônio *j*, ou equivalentemente, o sinal na saída do neurônio *i*. Se o neurônio *j* está na primeira camada escondida do MLP, então o índice *i* refere-se ao *i*-ésimo nó de entrada do MLP, para o qual escreve-se

$$y_i(n) = x_i(n) \tag{4.12}$$

onde $x_i(n)$ é o *i*-ésimo componente do vetor de entrada do neurônio *j*. Se, por outro lado, o neurônio *j* está na camada de saída, o índice j refere-se ao *j*-ésimo nó de saída do MLP, para o qual escreve-se

$$y_j(n) = o_j(n) \tag{4.13}$$

sendo $o_j(n)$ o *j*-ésimo componente do vetor de saída.

Esta saída é comparada com a resposta desejada $d_j(n)$ sendo obtido o sinal de erro $e_j(n)$ para o *j*-ésimo neurônio de saída.

Portanto, o passo direto começa na primeira camada escondida pela apresentação do vetor de entrada a ela e termina na camada de saída com a determinação do sinal de erro para cada neurônio desta camada.

O passo reverso (*backward pass)* começa na camada de saída, propagando os sinais de erro na direção contrária através do MLP (de volta para a entrada – retro-propagando), de camada em camada, e recursivamente computando os gradientes locais para cada neurônio.

Este processo recursivo de determinação dos gradientes locais permite que sejam executadas correções nos pesos sinápticos do MLP de acordo com a Regra Delta (Equação (4.8)).

Para um neurônio localizado na camada de saída, o gradiente local é simplesmente o sinal de erro daquele neurônio multiplicado pela primeira derivada de sua não-linearidade (equação (4.9a)).

A partir do gradiente local de cada neurônio da camada de saída, usa-se a equação (4.8) para computar as mudanças em todas as sinapses (conexões) que alimentam a camada de saída.

Obtidos os gradientes locais para os neurônios da camada de saída, usa-se a equação (4.9b) para computar o gradiente local de cada neurônio na camada à esquerda.

A partir do gradiente local de cada neurônio da camada à esquerda, usa-se a equação (4.8) para computar as mudanças em todas as sinapses (conexões) que alimentam esta camada.

Este procedimento é continuado recursivamente, propagando correções nos pesos sinápticos camada por camada, até a camada de entrada.

Note que durante cada ciclo passo direto - passo reverso ao longo da apresentação do conjunto de treino ao MLP, o vetor de entrada para aquele ciclo é mantido fixo.

4.1.2 A Derivada da Função de Ativação

A determinação do gradiente local para cada neurônio do MLP requer o conhecimento da derivada $\varphi'(\cdot)$ da função ativação $\varphi(\cdot)$ associada com o neurônio, conforme se infere de (4.9). Para que esta derivada exista, é necessário que a função de ativação $\varphi(\cdot)$ seja contínua. Uma função de ativação não-linear continuamente diferenciável, comumente aplicada em redes MLP é a função sigmoidal, já descrita no Capítulo 1. Duas formas da função sigmoidal são aqui tratadas:

a) Função Logística:

Esta forma de não-linearidade sigmoidal é definida por:

$$\varphi_j(v_j(n)) = \frac{1}{1+\exp(-av_j(n))}, \quad a>0 \text{ e } -\infty < v_j(n) < \infty \tag{4.14}$$

onde $v_j(n)$ é o potencial de ativação do neurônio *j*. De acordo com esta não-linearidade, a amplitude da saída fica restrita ao intervalo $0 \le y_j \le 1$. Omitindo os índices *n* e *j* por simplicidade, e derivando a função de ativação expressa em (4.14) com respeito a $v_j(n)$, temos

$$\varphi'(v) = \frac{d}{dv}\varphi(v) = \frac{d}{dv}\left\{\frac{1}{1+\exp(-av)}\right\} = \frac{a\ \exp(-av)}{[1+\exp(-av)]^2} = \tag{4.15}$$

$$= a\varphi^2(v)\left(\frac{1}{\varphi(v)} - 1\right) = a\varphi(v)(1-\varphi(v))$$

e como $y_j(n) = \varphi(v_j(n))$,

$$\varphi'(v_j(n)) = \frac{d}{dv}\varphi(v_j(n)) = a\ y_j(n)[1-y_j(n)] \tag{4.16}$$

Note na Equação (4.16) que a derivada atinge valor máximo em $y_j(n) = 0.5$, e seu valor mínimo (= zero) em $y_j(n) = 0$, ou $y_j(n) = 1.0$.

Já que a quantidade de mudança em um peso sináptico do MLP é proporcional à derivada, segue que, para uma função de ativação sigmoidal, os pesos sinápticos sofrem a maior alteração para aqueles neurônios no MLP onde os sinais assumem valores no meio de seu intervalo de variação. Esta é uma característica que contribui para a estabilidade do algoritmo de aprendizagem.

b) Função Tangente Hiperbólica:

Esta forma de não-linearidade sigmoidal é definida por:

$$\varphi_j(v_j(n)) = a\tanh(bv_j(n)) = a\left\{\frac{1-\exp(-2bv_j(n))}{1+\exp(-2bv_j(n))}\right\}, \quad a,b>0 \tag{4.17}$$

De acordo com esta não-linearidade, a amplitude da saída fica restrita ao intervalo $-a \le y_j \le a$. Omitindo os índices n e j por simplicidade, a derivada da função ativação pode ser obtida através de

$$\varphi'(v) = \frac{d}{dv}\varphi(v) = \frac{d}{dv}a\tanh(bv) = ab\,\mathrm{sech}^2(bv) = ab\left(1-\tanh^2(bv)\right) = \tag{4.18}$$

$$= ab\left(1-\left(\frac{a\tanh(bv)}{a}\right)^2\right) = ab\left(1-\frac{(a\tanh(bv))^2}{a^2}\right) = ab\left(1-\frac{\varphi^2(v)}{a^2}\right) =$$

$$= ab\left(1-\frac{y^2}{a^2}\right) = ab\left(\frac{a^2-y^2}{a^2}\right) = \frac{b}{a}\left(a^2-y^2\right) = \frac{b}{a}(a+y)(a-y)$$

Portanto

$$\varphi'(v_j(n)) = \frac{d}{dv}\varphi(v_j(n)) = \frac{b}{a}(a+y_j(n))(a-y_j(n)) \tag{4.19}$$

A Figura 4.6 mostra o gráfico da função tangente hiperbólica e de sua derivada para $a = 1.7159$ e $b = 2/3$.

Observe que, ao utilizarmos a Equação (4.16) como derivada da função logística e a Equação (4.19) como derivada da função tangente hiperbólica, o gradiente local δ_j dado por (4.9) pode ser calculado sem o uso explícito da definição analítica da função de ativação.

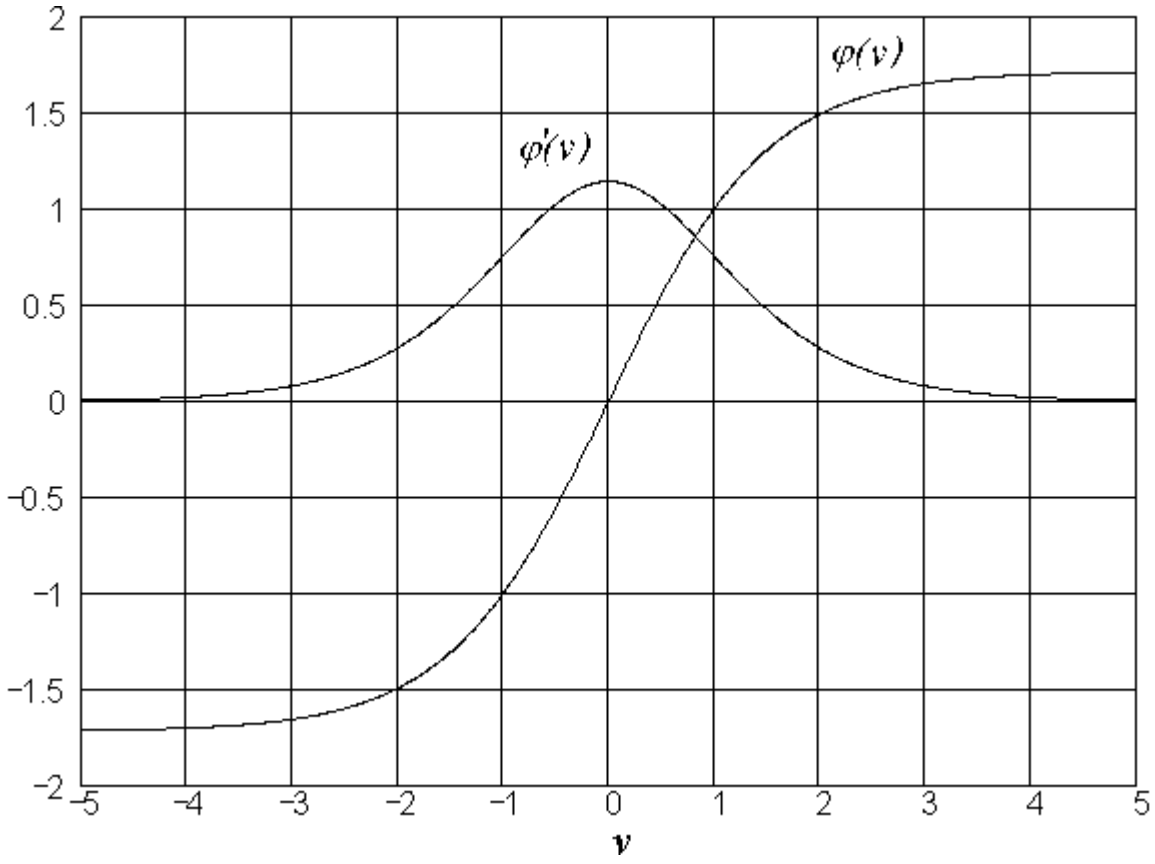

Figura 4.6: Gráfico de $\varphi(v) = a\tanh(bv)$ e $\varphi'(v) = ab\left(1 - \tanh^2(bv)\right)$ p/ $a = 1.7159$ e $b = 2/3$.

4.1.3 Razão de Aprendizado e Fator de Momento

O algoritmo *backpropagation* provê uma aproximação da trajetória de movimento sobre a superfície de erro no espaço de pesos sinápticos a qual, a cada ponto da superfície, segue a direção de descida mais íngreme.

Quanto menor for feita a razão de aprendizado η, menores serão as correções aplicadas aos pesos sinápticos do MLP de uma iteração para a próxima e mais suave será a trajetória no espaço de pesos. Isto é obtido sob o custo de uma lenta convergência do algoritmo até um valor de erro pequeno o suficiente para ser aceitável.

Se, por outro lado, a razão de aprendizado η é feita grande, de modo a acelerar a convergência do algoritmo, as correções feitas nos pesos sinápticos podem resultar demasiadamente grandes, de modo que o algoritmo se torna instável (oscilatório).

Um método simples utilizado para acelerar a convergência e manter a trajetória estável é o acréscimo do chamado Fator de Momento à Regra Delta (mostrada na Equação 4.7). Assim, teremos

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n) \tag{4.20}$$

onde a constante α é denominada de Constante de Momento com $0 < \alpha < 1$. Seu efeito é aumentar a velocidade da trajetória no espaço de pesos na direção da descida mais íngreme.

Da equação (4.20) nota-se que se a correção aplicada em determinado peso sináptico mantém o mesmo sinal algébrico durante várias iterações consecutivas, situação que ocorre quando a trajetória na superfície de erro desenrola-se ao longo de um caminho em descida íngreme, a correção do peso sináptico é acelerada pelo fator de momento, já que, sendo o caminho uma descida íngreme, o mínimo deve estar longe ainda. Um eventual mínimo local encontrado ao longo desta descida acelerada pode, então, ser facilmente transpassado. Isto ocorre porque, imaginando que a trajetória das coordenadas do vetor de pesos sinápticos $\underline{W}_j$ de um neurônio *j* qualquer seja a trajetória de um móvel de grande massa descendo uma ladeira irregular (i.e., com vários mínimos locais), em conseqüência do alto momento de inércia (energia cinética) do móvel devido à sua massa, as irregularidades (mínimos locais) não conseguem parar o movimento do móvel.

Por outro lado, se a correção aplicada em determinado peso sináptico troca o sinal algébrico durante várias iterações consecutivas, situação esperada ocorrer quando a trajetória na superfície de erro desenrola-se ao longo de um caminho próximo ao mínimo global, a correção do peso sináptico é freada pela redução do valor absoluto médio do fator de momento acrescentado, já que um mínimo (provavelmente global) está próximo e uma alta velocidade poderia desestabilizar o algoritmo em torno do mínimo.

4.1.4 Sumário do Algoritmo *Backpropagation* e Sugestões Operacionais

I - Inicialização:

1.Define-se o número de camadas do MLP. Em geral, sob o ponto de vista de rapidez de redução do MSE, é preferível utilizar poucas camadas escondidas com muitos neurônios por camada do que muitas camadas escondidas com poucos neurônios por camada. Isto porque o uso de muitas camadas escondidas "dilui" o efeito corretivo da retro-propagação dos sinais de erro sobre as sinapses ao longo do *backward pass*. Em conseqüência, o MLP demorará mais Épocas para atingir um MSE suficientemente baixo. Por outro lado, um número maior de camadas escondidas habilita o MLP a captar melhor as estatísticas de ordem superior do processo a ser aprendido, melhorando, assim, a capacidade de generalização do MLP. Isto ocorre porque um maior número de camadas escondidas torna o mapeamento $\Re^{m_1} \to \Re^{m_L}$ realizado pelo MLP, sendo m_1 e m_L respectivamente o número de nós de entrada e saída do MLP, um mapeamento com "maior não-linearidade recursiva". A informação sobre o processo a ser aprendido pelo MLP fica armazenada nas sinapses dos neurônios de cada camada, e as saídas de cada camada recursivamente alimentam as entradas da camada seguinte durante a fase de treino. Cada camada executa uma operação não-linear devido a função de ativação, portanto, a medida que uma camada alimenta a seguinte uma nova instância da operação não-linear é efetuada. A operação não-linear efetuada pela função de ativação é definida pela função exponencial e^x (ou por uma combinação de exponenciais no caso da Tangente Hiperbólica) , sendo e^x passível de ser expandida na série de potências $e^x = 1 + x + \frac{1}{2}\cdot x^2 + \frac{1}{6}\cdot x^3 + \frac{1}{24}\cdot x^4 + \frac{1}{120}\cdot x^5 + \ldots$. Ora, como a informação é recursivamente acumulada nas sinapses do MLP, sendo processada através de várias instâncias recursivas de uma série de potências durante o treino, fica implícito que o MLP acumula informação na forma de "estruturas de correlação estatística de ordem superior", isto é, após a fase de treino do MLP a informação armazenada no conjunto de sinapses está associada à

$E\{\underline{x}_i \otimes \underline{x}_j\} + E\{\underline{x}_i \otimes \underline{x}_j \otimes \underline{x}_k\} + E\{\underline{x}_i \otimes \underline{x}_j \otimes \underline{x}_k \otimes \underline{x}_l\} + \cdots$ onde $\underline{x}_i, \underline{x}_j$, ... representam individualmente todos os possíveis N vetores existentes no conjunto de treino, $E\{\cdot\}$ é o operador média estatística; $\underline{x}_i \otimes \underline{x}_j$ representa a matriz $m \times m$ formada pelos m^2 produtos entre os m componentes do vetor $\underline{x}_i$ pelos m componentes do vetor $\underline{x}_j$, isto é, $\underline{x}_i \otimes \underline{x}_j = \underline{x}_i \underline{x}_j^T$; $\underline{x}_i \otimes \underline{x}_j \otimes \underline{x}_k$ representa a estrutura cúbica em $\Re^3$ formada pelos m^3 produtos entre os m^2 elementos da matriz $\underline{x}_i \otimes \underline{x}_j$ e os m componentes do vetor $\underline{x}_k$; e assim sucessivamente.

1- Subtrai-se o vetor média do conjunto de N vetores de treino.

2- Normaliza-se a *i*-ésima componente de cada vetor de treino pelo desvio padrão do conjunto de N valores formado pela *i*-ésima componente de todos os N vetores de treino.

3- Normaliza-se o conjunto de N saídas desejadas para o intervalo [-1,+1].

4- Definem-se os parâmetros *a* e *b* da função de ativação. Em geral, $a = 1.7159$ e $b = 2/3$ são valores adequados para $\varphi(v) = a\tanh(bv)$, de modo que $\varphi'(0) = ab = 1.14 \approx 1$.

5- Inicializam-se os pesos sinápticos com valores aleatórios de distribuição uniforme. Uma possível heurística é adotar uma inicialização randômica com valores compreendidos no intervalo [$-2.4/F_i$, $+2.4/F_i$] onde F_i é o *fan-in* ou o número total de nós de entrada (sinapses) do neurônio. Outra possível heurística é adotar uma inicialização randômica com conjunto de valores de média zero e variância definida por $1/F_i$.

6- Definem-se o momento $0<\alpha<1$ e a razão de aprendizado $0<\eta<1$ por camada do MLP.

7- Visto que os neurônios próximos da camada de saída tendem a ter maiores gradientes locais, atribui-se a eles usualmente razões de aprendizado menores. Outro critério a ser considerado simultaneamente é que neurônios com muitas entradas devem ter η menores.

II - Treinamento:

1- Apresenta-se cada exemplo (vetor de entrada) do conjunto de treino ao MLP. Definindo como $\Gamma : \Re^{m_1} \rightarrow \Re^{m_L}$ o mapeamento ou processo a ser aprendido pelo MLP, sendo m_1 e m_L respectivamente o número de nós de entrada e saída do MLP, o conjunto de treino deve conter uma parcela suficientemente significativa do universo de vetores-exemplo que descrevem o processo Γ, caso contrário, após o treino o MLP não terá condições de inferir um resultado correto quando a ele for apresentado um vetor de Γ que não encontrava-se no conjunto de treino. Em outras palavras, o conjunto de treino deve conter uma parcela suficientemente significativa do universo de vetores-exemplo que descrevem o processo Γ para não prejudicar a capacidade de generalização do MLP.

2- Para cada exemplo executa-se completamente um ciclo passo direto - passo reverso, mantendo-se o vetor de entrada aplicado à entrada do MLP.

3- O final da apresentação de todos os exemplos do conjunto de treino define uma Época . A cada determinado número de Épocas em que for observado uma significativa queda no MSE, aumenta-se o momento α e/ou a razão de aprendizado η.

4- Prossegue-se o treino do MLP de Época em Época, eventualmente ajustando α e η, até que se atinja o Critério de Parada.

III - Critério de Parada:

O critério de parada no treino de uma rede MLP é subjetivo, já que não existe prova de que o algoritmo *backpropagation* tenha convergido para o mínimo global da superfície de erro (se é que existe o mínimo global).

Sugere-se como critério de parada o seguinte procedimento:

As iterações de treino são terminadas se:

1- O valor do MSE atingiu um valor suficientemente baixo **e/ou**

2- A razão de variação do MSE atingiu um valor suficientemente baixo em valor absoluto e negativo.

Quando qualquer uma das condições acima é atingida, considera-se que o MLP não necessita mais ser treinado. Note que o critério 2 pode significar que o *backpropagation* ficou preso em um mínimo local e não global.

É importante observar que um MSE baixo ao final do treino não necessariamente implica em uma alta capacidade de generalização. Se o conjunto de treino escolhido para representar o processo Γ a ser aprendido pelo MLP constituir um sub-conjunto cujas propriedades estatísticas não correspondem às de Γ, então o MLP falhará em inferir o resultado correto quando um vetor de Γ que não pertence ao conjunto de treino for apresentado ao MLP.

4.1.5 A Determinação da Expressão do Gradiente Local do Algoritmo *Backpropagation*

Conforme já discutido, o algoritmo *backpropagation* aplica a correção $\Delta w_{ji}(n)$ ao peso sináptico $w_{ji}(n)$, tendo como base a direção contrária do gradiente local $\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$ da superfície de erro $\varepsilon(w)$ relativo ao peso sináptico. Em última análise, o gradiente local $\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$ representa a variação resultante $\Delta\varepsilon(w)$ no erro quadrático instantâneo $\varepsilon(w)$ do MLP quando é aplicada uma pequena variação ao peso sináptico que liga a saída do neurônio *i* ao *i*-ésimo nó de entrada do neurônio *j*.

De acordo com a regra da cadeia do cálculo diferencial, o gradiente local $\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)}$ pode ser expresso por

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{4.21}$$

De (4.3) temos

$$\varepsilon(n) = \frac{1}{2} \sum_{j=0}^{m_L - 1} e_j{}^2(n) = \frac{1}{2} \left\{ e_0{}^2(n) + e_1{}^2(n) + \cdots + e_j{}^2(n) + \cdots + e_{m_L-1}{}^2(n) \right\} \tag{4.22}$$

onde m_L é o número de neurônios da camada de saída.

Derivando (4.22) em relação à $e_j(n)$ resulta em

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = \frac{1}{2} \frac{\partial}{\partial e_j(n)} \left\{ e_0{}^2(n) + e_1{}^2(n) + \cdots + e_j{}^2(n) + \cdots + e_{m_L-1}{}^2(n) \right\} = e_j(n) \tag{4.23}$$

Derivando (4.2) em relação à $y_j(n)$ obtemos

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{4.24}$$

Derivando (4.6) em relação à $v_j(n)$ temos

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi'_j(v_j(n)) \tag{4.25}$$

valor já determinado por (4.16) ou (4.19), dependendo da função de ativação utilizada no MLP.

De (4.5) temos

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) = \begin{Bmatrix} w_{j0}(n) y_0(n) + w_{j1}(n) y_1(n) + \cdots \\ \cdots + w_{ji}(n) y_i(n) + \cdots + w_{jm}(n) y_m(n) \end{Bmatrix} \tag{4.26}$$

Derivando (4.26) em relação à $w_{ji}(n)$ resulta em

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = \frac{\partial}{w_{ji}(n)} \left\{ \begin{matrix} w_{j0}(n)y_0(n) + w_{j1}(n)y_1(n) + \cdots \\ \cdots + w_{ji}(n)y_i(n) + \cdots + w_{jm}(n)y_m(n) \end{matrix} \right\} = y_i(n) \tag{4.27}$$

Substituindo (4.23), (4.24), (4.25) e (4.27) em (4.21), temos

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} = e_j(n)(-1)\varphi_j'(v_j(n))y_i(n) \tag{4.28}$$

Substituindo (4.28) em (4.7)

$$\Delta w_{ji}(n) = w_{ji}(n+1) - w_{ji}(n) = \eta e_j(n)\varphi_j'(v_j(n))y_i(n) \tag{4.29}$$

Note de (4.28) que o termo $e_j(n)\varphi_j'(v_j(n))$ em (4.29) origina-se da cadeia de operações

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = e_j(n)(-1)\varphi_j'(v_j(n)) \tag{4.30}$$

isto é

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -e_j(n)\varphi_j'(v_j(n)) \tag{4.31}$$

ou ainda, simplificando os diferenciais intermediários em (4.31), obtemos o denominado gradiente local j

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n)) \tag{4.32}$$

Note que (4.32) é equivalente à (4.9a). Substituindo (4.32) em (4.29) obtemos

$$\Delta w_{ji}(n) = w_{ji}(n+1) - w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{4.33}$$

Observe que (4.33) é a Equação (4.8).

No entanto, (4.32) não pode ser explicitamente determinada exceto para neurônios na camada de saída, porque na camada de saída existe um erro $e_j(n)$ associado a cada neurônio *j* no instante *n*.

Como proceder, então, para determinar o gradiente local $\delta_j(n)$ de neurônios pertencentes a camadas escondidas, onde não existe um erro explícito associado a cada neurônio? Para solucionar este problema, inicialmente vamos expandir (4.32) em

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = e_j(n)\varphi_j'(v_j(n)) = -\frac{\partial \varepsilon(n)}{\partial y_j(n)}\frac{y_j(n)}{\partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial y_j(n)}\varphi_j'(v_j(n)) \tag{4.34}$$

Reescrevendo (4.22), temos

$$\varepsilon(n) = \frac{1}{2}\sum_{k=0}^{m_L-1} e_k^{\,2}(n) = \frac{1}{2}\sum_k e_k^{\,2}(n) \tag{4.35}$$

A Equação (4.35) é idêntica à (4.22) apenas com o índice do somatório substituído por *k* para caraterizar que (4.35) refere-se à erros quadráticos de neurônios da camada de saída. Isto é feito para evitar confusão com neurônios da camada escondida imediatamente à esquerda da camada de saída, os quais, segundo a convenção aqui adotada devem ser indexados por *j*. É conveniente relembrar que estamos buscando determinar o gradiente local $\delta_j(n)$ de neurônios pertencentes a camadas escondidas, onde não existe um erro explícito associado a cada neurônio.

De (4.35) temos

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \frac{1}{2}\frac{\partial}{\partial y_j(n)}\sum_k e_k^{\,2}(n) = \sum_k e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)} \tag{4.36}$$

que pode ser re-escrita como

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k(n)\frac{\partial e_k(n)}{\partial y_j(n)} = \sum_k e_k(n)\frac{\partial e_k(n)}{\partial v_k}\frac{\partial v_k}{\partial y_j(n)} \tag{4.37}$$

Mas, da Figura 4.4 temos para o neurônio *k* na camada de saída

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)) \tag{4.38}$$

e, portanto, derivando (4.38) em relação à $v_k(n)$

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)) \tag{4.39}$$

De (4.5), temos que o potencial de ativação para um neurônio *k* situado na camada à direita da que estão situados os neurônios de índice *j* a ele conectado é dado por

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) = \begin{Bmatrix} w_{k0}(n)y_0(n) + w_{k1}(n)y_1(n) + \cdots \\ \cdots + w_{kj}(n)y_j(n) + \cdots + w_{km}(n)y_m(n) \end{Bmatrix} \tag{4.40}$$

onde *m* é o número total de entradas (excluindo a polarização) aplicadas ao neurônio *k*.

Derivando (4.40) em relação a $y_j(n)$ temos

$$\frac{\partial v_k(n)}{\partial y_j(n)} = \frac{\partial}{\partial y_j(n)} \begin{Bmatrix} w_{k0}(n)y_0(n) + w_{k1}(n)y_1(n) + \cdots \\ \cdots + w_{kj}(n)y_j(n) + \cdots + w_{km}(n)y_m(n) \end{Bmatrix} = w_{kj}(n) \tag{4.41}$$

Substituindo (4.39) e (4.41) em (4.37) obtemos

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k} \frac{\partial v_k}{\partial y_j(n)} = -\sum_k e_k(n) \varphi_k'(v_k(n)) w_{kj}(n) \tag{4.42}$$

Mas de (4.34) com *j* substituído por *k* temos

$$\delta_k(n) = e_k(n) \varphi_k'(v_k(n)) \tag{4.43}$$

Substituindo (4.43) em (4.42),

$$\frac{\partial \varepsilon(n)}{\partial y_j(n)} = -\sum_k \delta_k(n) w_{kj}(n) \tag{4.44}$$

Substituindo (4.44) em (4.34) resulta em

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial v_j(n)} = -\frac{\partial \varepsilon(n)}{\partial y_j(n)} \varphi_j'(v_j(n)) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{4.45}$$

Observe que (4.45) é a Equação (4.9b), válida quando o neurônio *j* encontra-se em uma camada escondida.

Desta forma, fica demonstrada a consistência das Equações (4.8)/(4.20) e (4.9) utilizadas na atualização das sinapses do MLP durante o *backward pass*.

4.2 Referências Bibliográficas do Capítulo 4

[1] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press, Massachusetts, 1995.

[2] R. D. Strum e D. E. Kirk, *First Principles of Discrete Systems and Digital Signal Processing*, Addison-Wesley, 1989.

[3] S. Haykin, *Adaptive Filter Theory*, 3rd ed., Prentice Hall, Upper Saddle River, New Jersey, 1996.

[4] S. Haykin, *Neural Networks*, 2nd ed., Prentice Hall, Upper Saddle River, New Jersey, 1999.

[5] Z.L.Kovács, *Redes Neurais Artificiais*, Editora Acadêmica São Paulo, São Paulo, 1996.