

## INTELIGÊNCIA ARTIFICIAL VOLTADA AO PARADIGMA REDES NEURAIIS

Segundo RABUSKE (1995), “representa-se o conhecimento para posteriormente recuperá-lo, para raciocinar com ele e para adquirir mais conhecimento”.

### 1.1 VISÃO GERAL SOBRE INTELIGÊNCIA ARTIFICIAL E SISTEMAS ESPECIALISTAS

Mesmo não concordando com a definição ou conceito de IA, o certo é que os grandes autores do assunto normalmente citam alguns trabalhos e autores como cruciais para esta evolução como Alan Turing com a tese Church-Turing, tida por muitos como princípio fundamental da computação, a dupla McCulloch-Pitts com o artigo *A logical calculus of the ideas immanent in nervous activity* (1943), passando por Noam Chomsky que em 1959 criou a Hierarquia de Chomsky que é até hoje muito utilizada em compiladores e interpretadores, atravessando décadas com apoio de tecnologias desenvolvidas pela IBM, Raymond Reiter, logicamente que ainda poderiam ser citados vários outros como Goldber, Minsky e Zadeth, na verdade seria necessário todo um trabalho para discorrer com justiça sobre os autores que contribuíram para o avanço desta área.

Mais especificamente na área computacional, é conhecido o termo especialista, que faz referência a um sistema que substitui o pensamento racional na resolução de um problema específico, antigamente eram baseados em sistemas algorítmicos, ou lógicos (SABBATINI, 1993).

Atualmente a aplicação de sistemas especialistas cientificamente é quase que inexistente, segundo BARRETO (2002), muitas vezes não é mais considerado como uma técnica de IA.

Na última década, ao nível de implementação informatizada WIVES (2001) comenta que agora existem, além dos sistemas clássicos uma série de novas implementações como máquina de inferência os métodos connexionista, bayes, vetores e árvores de decisão, lógica fuzzy e lógica bayesiana.

O método connexionista ou rede neural foi escolhido para a implementação da série de testes cognitivos e será detalhado neste capítulo.

## 1.2 REDES NEURAI

Dentre as diferentes formas de implementação, a rede neural foi considerada a mais adequada, pois, embora os exames que compõe a Avaliação Geriátrica Ampla tenham estudos dos quais podem ser extraídas algumas regras, estas não são suficientes para criar a máquina de inferência de um sistema especialista clássico, premissas para uma abordagem utilizando lógica fuzzy ou uma matriz de possibilidades bayesiana, uma mantenha para demonstrar isto seria necessário conhecê-la melhor, portanto, nas próximas páginas mostraremos maiores detalhes sobre a rede neural, o neurônio artificial, suas características e formas de implementação.

Os sistemas especialistas procuram imitar ou modelar a maneira de resolver problemas de um especialista humano. Mas eles não podem modelar a inteligência humana, programar soluções ou apontar para soluções de problemas específicos. Os projetistas de redes neurais procuram colocar inteligência no hardware ou software na forma de uma capacidade de generalização para aprender (JAIN, MAO e MOHIUDDIN, [1996](#)). As RNAs (Redes Neurais Artificiais) que classificam padrões colocam a inteligência em seus pesos sinápticos. Com sistemas especialistas é possível saber porque uma determinada solução foi escolhida, entretanto, isto não é possível com as redes neurais. Elas agem como uma caixa preta, porém são muito úteis na classificação de padrões, agrupamento de dados, previsões e análises segundo HAYKIN (1999).

A primeira rede neural data de 1943, entretanto somente tem chamado atenção nos últimos 20 anos. Recentemente, foi verificado que era possível resolver com vantagens interessantes problemas cotidianos com a mesma linha de raciocínio do cérebro humano. Obtendo resultados satisfatórios sem ter exata noção de como a solução é processada, o que não ocorre na solução trivial onde é necessário detalhar todo o problema de uma forma estruturada (BARRETO, 2000).

É difícil encontrar uma definição global para uma rede neural, uma delas considera que a rede é um agregado de pequenos e simples processadores, cada um dotado de uma parcela local de memória e processamento conectados através de uma rede de transmissão (MARRONE, 2005).

De acordo com HAYKIN (1999), uma rede neural é um processador paralelo distribuído que tem uma tendência natural para armazenar conhecimento e torna-o passível de reuso. Este processador assemelha-se ao cérebro em dois aspectos. Primeiramente que o conhecimento é assimilado pelo cérebro em um processo de aprendizagem e o segundo são as conexões internas entre os neurônios conhecidas como sinapses que são utilizadas para transmitir o conhecimento, como no cérebro humano.

Para entender como funciona uma rede neural primeiramente temos de fazer uma explanação sobre como é o funcionamento do neurônio humano, o original.

Cada neurônio pode disparar pulsos elétricos através da rede sináptica que é recebido por outros neurônios. Quando um neurônio recebe uma quantidade determinada de energia através de pulsos ele libera esta energia para os neurônios seguintes em um processo que é chamado de ativação. A rede de neurônios muda durante sua existência e a quantidade e intensidade de pulsos também, o que faz com que a rede tenha um comportamento diferente, isto ocorre através de mudanças internas e externas motivadas pelos 5 sentidos humanos. A este processo chamamos de aprendizado (TETTAMANZI E TOMASSINI,2001).

Não é possível, no ponto atual de evolução humana, criar um cérebro artificial, mas já existem muitas implementações que consideram uma rede de neurônios para realizar tarefas simples. Estas implementações que são chamadas de Redes Neurais Artificiais (ou ANN, *Artificial Neural Network*) não são de fato inteligentes, mas conseguem realizar tarefas como reconhecimento de padrões ou extração de regras, entre outras possibilidades. Por outro lado um dos grandes pontos a que se propõe a tecnologia conexionista consiste no fato que teoricamente as associações lógicas que são criadas têm a capacidade de “aprender”, o que poderia ser traduzido de melhor forma como capacidade de se adaptar a uma situação.

Ao término, ou seja, após o sinal percorrer todo o conjunto de neurônios da rede, gera um sinal de saída em função da entrada (SOARES & NASCIMENTO,1997), que poderia ser considerado como a “solução” para o problema.

## 1.2.1 Classificação de Redes Neurais

### 1.2.1.1 Topologia

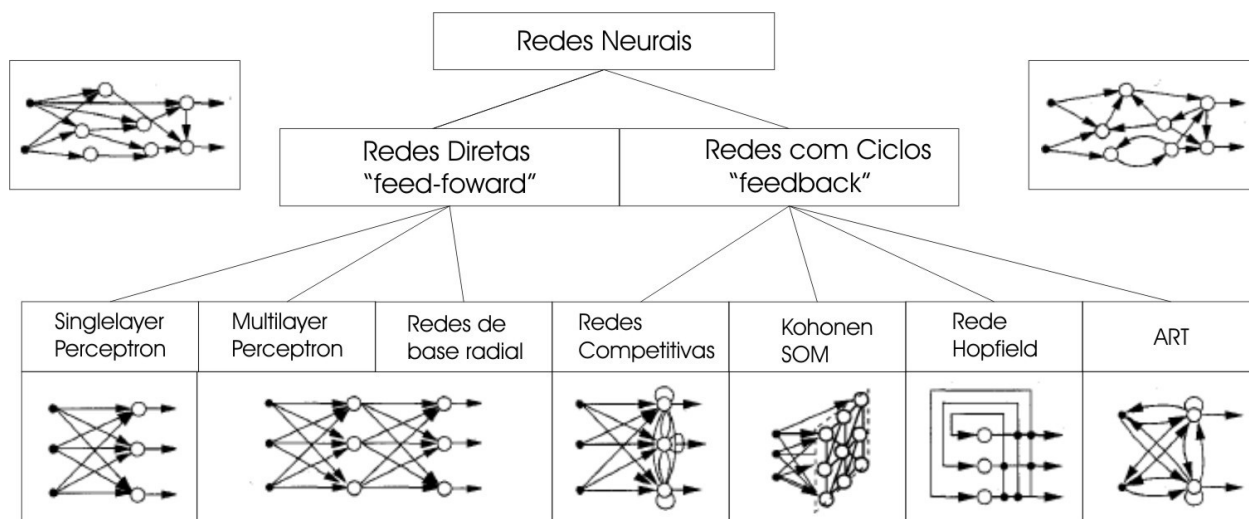
As redes neurais podem ser vistas como grafos, nos quais os neurônios são vistos como nodos e as sinapses são as conexões segundo JAIN, MAO, MOHIUDDIN (1996).

Segundo esta classificação as redes neurais podem ser vistas como:

- **Diretas ou *feed-foward***: Quando os grafos não têm *loops*.

- **Recorrentes:** Quando existem *loops* devido principalmente às conexões de retroalimentação.

Na Figura 3 percebe-se que, nas redes diretas a estrutura é disposta em camadas, onde a saída de um neurônio é diretamente conectada ao neurônio da camada seguinte, não existe retroalimentação, como nas redes com ciclos.



onde:

*Singlelayer Perceptron:* Perceptron Única Camada

*Multi layer Perceptron:* Perceptron Multi-camadas

SOM: *Self-Organized Map* (Mapa Auto-organizável)

ART: *Adaptative Resonance Theory* (Teoria da Ressonância Adaptativa)

**Figura 1** Topologia de Redes Neurais (Fonte: JAIN, MAO, MOHIUDDIN, 1996).

### 1.2.1.2 Aprendizado

A classificação através do aprendizado é normalmente definida pela existência ou não de realimentação explícita de dados partindo do universo exterior à rede neural durante o período de treinamento, ou seja, os dados são apresentados a rede uma ou várias vezes durante o treinamento. No caso em que não existe esta realimentação de dados o aprendizado ocorre sem “professor”. Por este conceito temos, segundo HAYKIN (1999):

**Aprendizado supervisionado:** No caso em que existe a correção no resultado obtido. Normalmente ocorre da seguinte maneira, convencionase o comportamento da saída da rede para um

resultado e corrige-se este resultado à medida que se aproxima ou se afasta do resultado desejado, esta correção é feita normalmente alterando-se as conexões sinápticas e rerepresentando os dados.

**Aprendizado não-supervisionado:** Neste caso não se corrige a saída, utiliza-se um esquema de forma a fazer que para entradas semelhantes à rede tenha respostas semelhantes.

Na Tabela 14, pode-se observar então a taxonomia das mais conhecidas redes neurais, classificada pelo tipo de aprendizado.

Ταβέλα 1: Classificação de Redes Neurais por Aprendizado (Fonte: JAIN; MAO; MOHIUDDIN, 1996)

Paradigma	Regra de Aprendizado	Arquitetura	Algoritmo de Aprendizado	Utilização
Supervisionado	Correção de erro	<i>Perceptron</i> simples e multi-camada	<i>Perceptron</i> com retropropagação de erro. <i>Adaline</i> e <i>Madaline</i>	Classificação de padrões, aproximação de funções e controle.
	<i>Boltzmann</i>	Recorrente	Aprendizado Boltzmann	Reconhecimento de padrões.
	<i>Hebbian</i>	Multi-camada com retropropagação	Análise linear	Análise de dados e classificação de padrões
	Competitivo	Competitivo	Vetor de aprendizado	Compressão de dados.
		<i>ART Network</i>	<i>ARTMap</i>	Classificação de padrões.
Não-supervisionado	Correção de erro	Multi-camada com retropropagação	Sammon's projection	Análise de Dados
		Retropropagação	<i>PCA</i>	Análise de dados
	<i>Hebbian</i>	Hopfield		Compressão de Dados
		Competitivo	Quantização	Categorização
	Competitivo			Compressão de dados
		<i>Kohonen SOM</i>	<i>Kohonen SOM</i>	Categorização, análise de dados.
		Rede <i>ART</i>	<i>ART1, ART2</i>	Categorização

onde:

*ADALINE*: *Adaptive Linear Element* (Elemento Adaptativo Linear)

*MADALINE*: *Multiple Adaptive Linear Element* (Elemento Adaptativo Linear Múltiplo)

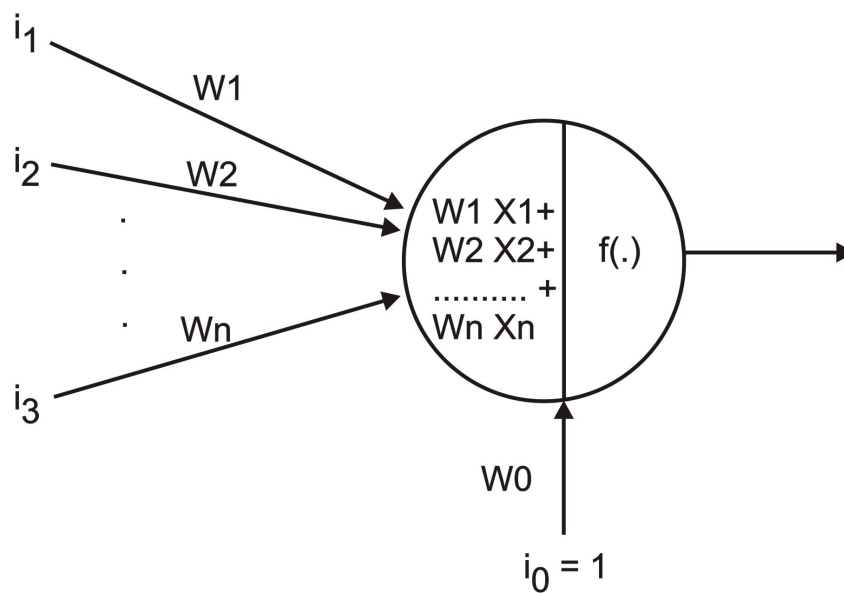
A Tabela 14 serve como referência ao leitor sobre as principais características de algumas redes neurais utilizadas usualmente, mais informações podem ser encontradas, por exemplo, em HAYKIN (1999) ou FAUSSET (1996).

Agora estudaremos as características de uma delas em específico, a rede *Perceptron* multi-camadas, uma das redes mais usuais e que por ter características de ser forte para classificação de padrões conforme listado na Tabela 14, é interessante para utilização no problema de gerar indicativos para demência e depressão proposto neste trabalho.

### 1.3 REDE *PERCEPTRON* MULTI-CAMADAS

O conceito básico do uso de um *perceptron* foi introduzido por Rosenblatt em 1958. É uma das formas mais conhecidas e exploradas de rede neural, e apresenta ainda hoje uma grande possibilidade de aplicações. Apesar de ter uma dificuldade para treinamento similar a outras redes neurais, a literatura acerca do *perceptron* multi-camadas é vasta e apresenta uma boa base para soluções de problemas segundo JAIN, MAO, MOHIUDDIN (1996).

O *perceptron* utiliza os valores contínuos de entrada, retira o peso e passa adiante o resultado na forma de um ou dois valores, especificamente em um *perceptron* o fator de aprendizado consiste em determinar um valor que pode ser discreto como  $\{0,1\}$  ou  $\{-1,0,1\}$  ou contínuo  $[0,1]$  ou  $[-1,+1]$  HAYKIN (1999).

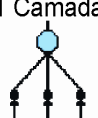
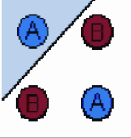
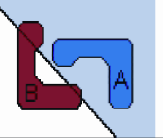

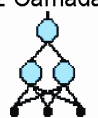
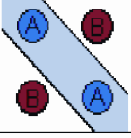
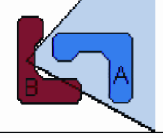
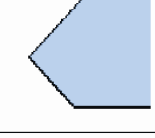






**Figura 2** Modelo clássico do neurônio *Perceptron*.

Onde  $W$  e  $I$  são peso e respectivos valores no vetor de entrada,  $W_0$  e  $I_0$  formam um tipo especial de entrada conhecido como bias, o resultado final é o nível de ativação do neurônio, dado pela soma de todos os pesos e respectivos valores de entrada incluindo o valor de bias de acordo com a equação 1 (FAUSSET, 1996):

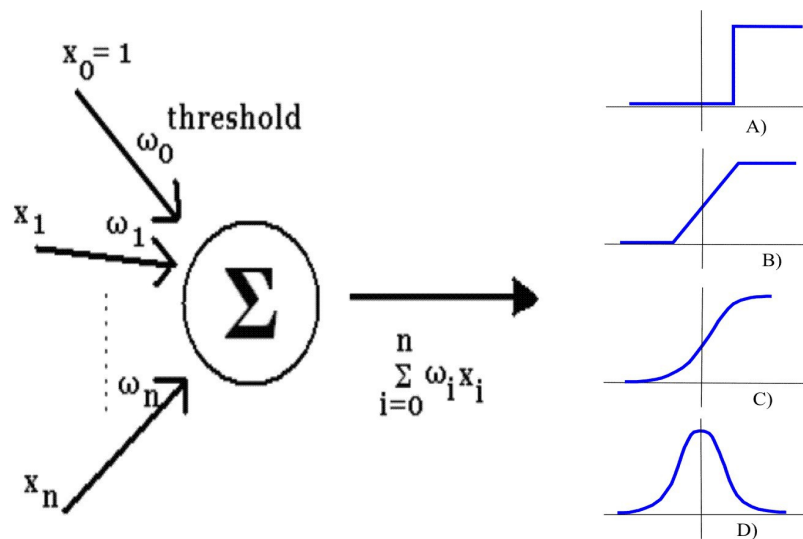
$$a = w_0 + w_1 i_1 + w_2 i_2 + ..... + w_n i_n \quad (1)$$

Ao unir vários neurônios do tipo perceptron em uma única camada MINSKY (1969) conseguiu um modelo útil para classificar valores contínuos de entrada, mas nada além disto, com experimentação verificou que o modelo de uma camada de neurônios é limitado, como foi provado por que podia somente resolver problemas lineares. Então, como evolução deste modelo, surge a estrutura de multi-camadas (*Multi Layer Perceptron*) que passa a utilizar uma função de transferência baseada em um gradiente e que passa então a permitir decisões mais complexas baseadas em um aprendizado mais aperfeiçoado baseado no uso do *perceptron* em “camadas”. A Figura 5 demonstra este raciocínio, onde com uma camada a região de decisão é criada através de um plano, passando por regiões fechadas e convexas conseguidas através de 2 camadas e de complexidade arbitrária para 3 camadas ou mais.

Estrutura	Regiões de Decisão	Problema de XOR	Classes com Regiões mescladas	Formas de Regiões mais Gerais
1 Camada 	Médio Plano limitado por um hiperplano			
2 Camadas 	Regiões fechadas e convexas			
3 Camadas 	Complexidade arbitrária limitada pelo número de neurônios			

**Figura 3** Demonstração da relação estrutura formada pelo número de camadas de neurônios versus a região de decisão formada e conseqüente capacidade de decisão (Fonte: JAIN, MAO, MOHIUDDIN, 1996).

Fundamentado sobre a mesma estrutura de funcionamento do *perceptron* foram criadas várias outras redes neurais como a Adaline ou Madaline. Nelas é utilizada a chamada regra Delta na qual um algoritmo gera um Gradiente Descendente que é utilizado para procura dos pesos que melhor se enquadram, ou seja, os pesos nos quais a diferença entre a raiz da saída encontrada e a observada é menor. Na Figura 6 segue comparação entre os diferentes tipos de função ou métodos de ativação, entre eles a função sigmoidal que é mais usualmente utilizada com o *perceptron*:



**Figura 4** Métodos de ativação: a) *Threshold*, b) *Linear*, c) *Sigmóide* e d) *Gaussiano* (Fonte: JAIN, MAO, MOHIUDDIN, 1996)

Baseado na evolução da regra Delta foi desenvolvido o algoritmo chamado de *BackPropagation* (retropropagação) que é hoje o mais usado para o treinamento de redes multi-camadas *perceptron*. Como até o início dos anos 70 nenhum algoritmo de aprendizado para estas redes multi-camadas havia sido desenvolvido, as pesquisas na área de redes neurais acabaram caindo em descrédito, até que nos anos 80, um algoritmo chamado Retropropagação ou Backpropagation, veio fazer renascer o interesse geral pelas redes neurais (JAIN, MAO, MOHIUDDIN, 1996).

### 1.3.1 Aprendizado Utilizando BackPropagation

O algoritmo provê um aprendizado supervisionado, isto é, ele procura achar iterativamente a mínima diferença entre as saídas desejadas e as saídas obtidas pela rede neural, segundo um erro mínimo. Desta forma, ajustando os pesos entre as camadas através da retropropagação do erro encontrado em cada iteração.

Uma solução para superar o problema do aprendizado da classificação de padrões não-linearmente separáveis é a utilização de uma camada intermediária de neurônios, chamada Camada Escondida ou oculta ("*Hidden Layer*"), de modo a poder implementar superfícies de decisão mais complexas. A característica principal da camada escondida é que seus elementos se organizam de tal



forma que cada elemento aprenda a reconhecer características diferentes do espaço de entrada, assim, o algoritmo de treinamento deve decidir que características devem ser extraídas do conjunto de treinamento. A desvantagem em utilizar camada escondida é que o aprendizado se torna muito mais difícil (FAUSSET, 1996).

O algoritmo Backpropagation foi desenvolvido de maneira independente por vários pesquisadores trabalhando em diferentes áreas aproximadamente na mesma época. Em 1974, WERBOS apresentou o algoritmo enquanto desenvolvia sua tese de doutorado em estatística e o chamou de "Algoritmo de Realimentação Dinâmica". Parker redescobriu o algoritmo e chamou-o de "Algoritmo de Aprendizado Lógico". Porém, foi com o trabalho de Rumelhart, Hinton e Williams do grupo PDP ("*Parallel Distributed Processing*") do MIT (*Massachusetts Institute of Technology*) em 1986 divulgou e popularizou o uso do Backpropagation para o aprendizado em redes neurais. O algoritmo Backpropagation é hoje em dia a técnica de aprendizado supervisionado mais utilizada para redes neurais unidirecionais multi-camadas. Uma descrição sumária da operação da rede é apropriada neste instante para ilustrar como o Backpropagation é utilizado para o aprendizado de problemas de mapeamento complexo. Basicamente, a rede aprende um conjunto pré-definido de pares de exemplos de entrada/saída em ciclos de propagação/adaptação. Depois que um padrão de entrada foi aplicado como um estímulo aos elementos da primeira camada da rede, ele é propagado por cada uma das outras camadas até que a saída seja gerada. Este padrão de saída é então comparado com a saída desejada e um sinal de erro é calculado para cada elemento de saída.

O sinal de erro é então retro-propagado da camada de saída para cada elemento da camada intermediária anterior que contribui diretamente para a formação da saída. Entretanto, cada elemento da camada intermediária recebe apenas uma porção do sinal de erro total, proporcional apenas à contribuição relativa de cada elemento na formação da saída original. Este processo se repete, camada por camada, até que cada elemento da rede receba um sinal de erro que descreva sua contribuição relativa para o erro total. Baseado no sinal de erro recebido, os pesos das conexões são então atualizados para cada elemento de modo a fazer a rede convergir para um estado que permita a codificação de todos os padrões do conjunto de treinamento (FAUSSET, 1996).

O Backpropagation utiliza o mesmo princípio da Regra Delta, qual seja, a minimização de uma função custo, no caso, a soma dos erros médios quadráticos sobre um conjunto de treinamento, utilizando a técnica de busca do gradiente-descendente. Por esta razão, o algoritmo *Backpropagation* também é chamado muitas vezes de Regra Delta Generalizada ("*Generalized Delta-Rule*"). A

modificação principal em relação à Regra Delta foi à utilização de funções contínuas e suaves como função de saída dos neurônios ao invés da função de limiar lógico. Como as funções de saída passaram a ser deriváveis, isto permitiu a utilização da busca do gradiente-descendente também para os elementos das camadas intermediárias (FAUSSET,1996).

O algoritmo para implementação em linguagem de alto nível pode ser visto na Figura 7:

**Figura 5** Algoritmo de Retropropagação de Erro – Regra Delta

```
Inicializar todos os pesos de forma randômica
Até terminar a condição de execução fazer{
  Propagar a entrada para a rede e computar as saídas
  observadas
  Retropropagar o erro encontrado{
    Para cada saída k calcular o erro como  $S_k$ .
    Para cada unidade oculta calcular o erro  $Sh$ .
    Atualizar cara peso de cada neurônio
  }
}
```

### 1.3.2 Formalização do Algoritmo *Backpropagation*

Neste trabalho iremos dar apenas uma descrição resumida sobre como funciona a matemática do algoritmo backpropagation, para saber mais detalhes ver (HAYKIN, 1994).

A formulação propõe um mapeamento entre na forma de um conjunto de coordenadas cartesiandas no qual, para cada  $X_i$  de entrada existe um valor  $Y_i$  de saída.

Considerando este mapeamento, se for escolhida uma função qualquer de pontos  $P(X_i, Y_i)$ , a rede será capaz, depois de treinada, de interpolar entre eles para reconhecer pontos não apresentados no conjunto de exemplos, desta forma aproximando a função.

De acordo com a descrição de Haykin (1994), temos que os elementos de entrada distribuem os valores entre os neurônios das camadas escondidas, então como resultado temos que o valor para a saída da camada escondida vale:

$$net_{kj}^h = \sum_{i=1}^n w_{ji}^h x_{ki} + \theta_j^h \quad (2)$$

onde:

$w_{ji}^h$  é o peso da conexão entre o  $i$ -ésimo elemento da camada de entrada e o  $j$ -ésimo elemento da camada escondida;

$\theta_j^h$  é o termo *bias* que provê um valor fictício de entrada igual a 1, dando um grau de liberdade maior para a função de saída do neurônio pois permite a ativação ou não em determinadas condições.

Portanto, se isolarmos somente um neurônio, teremos que seu valor de saída valerá:

$$i_{kj} = f_j^h(net_{kj}^h) \quad (3)$$

Considerando o cálculo anterior das equações 2 e 3, e estendendo o raciocínio para as camadas de saída temos as equações 4 e 5:

$$net_{kp}^o = \sum_{j=1}^l w_{pj}^o x_{ji} + \theta_p^o \quad (4)$$

$$o_{kp} = f_p^o(net_{kp}^o) \quad (5)$$

O objetivo é minimizar o erro, sendo que a diferença entre o valor desejado (saída) e o calculado ( $net$ ), portanto, teremos que o erro  $E_{kp}$  é dado pela equação 6:

$$E_{kp} = y_{kp} - o_{kp} \quad (6)$$

E para todos os neurônios podemos formular um erro total utilizando o erro médio quadrático sendo assim, teremos:

$$E_k = \frac{1}{2} \sum_{p=1}^m E_{kp}^2 \quad (7)$$

Como, segundo ROISENBERG (1996) o que interessa é encontrar o menor erro quadrático, a busca na superfície de erros se dá pelo gradiente negativo, ou seja, a direção de modificações dos pesos será dada de acordo com a direção que o vetor gradiente seguir na superfície.

Finalmente, temos que a atualização dos pesos dos neurônios da camada de saída dada pela fórmula 8:

$$w_{pj}^o(t+1) = w_{pj}^o(t) + \Delta_k w_{pj}^o(t) \quad (8)$$

Onde,

$$\Delta_k w_{pj}^o = \eta (y_{kp} - o_{kp}) f_p^{o'}(net_{kp}^o) i_{kj}$$

Ainda segundo ROISENBERG (1996), o vetor gradiente pode oscilar bastante entre pontos da superfície, dificultando e diminuindo a velocidade de convergência para o erro médio quadrático desejado. Então foi idealizado um termo conhecido como *momentum*, que será visto com mais detalhes em seguida, e que, na prática define o quão importante é a variação naquele “momento”. O termo momentum é definido por  $\alpha$  e é inserido como na equação 9:

$$w_{pj}^o(t+1) = w_{pj}^o(t) + \alpha \Delta_k w_{pj}^o(t) \quad (9)$$

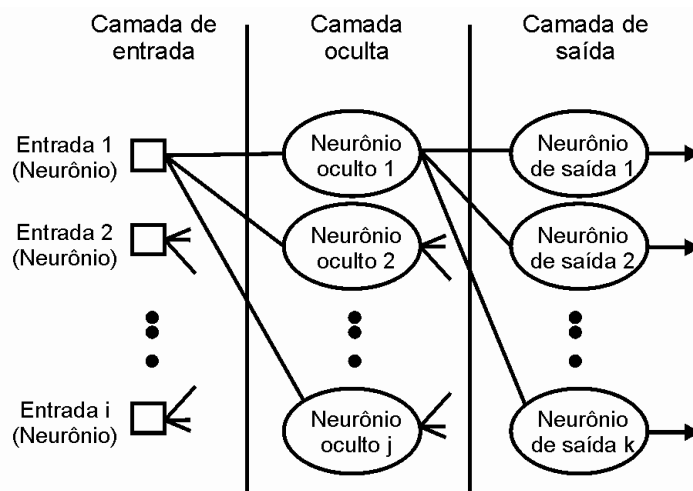
onde:  $\alpha$  é conhecido como momentum.

### Parâmetros de Retropropagação de Erro

As redes neurais multi-camadas são normalmente contadas da segunda camada em diante. A camada de entrada não tem nenhuma unidade de processamento. Como a camada de entrada não é considerada pode-se então dizer que na Figura 8 a primeira camada é a oculta onde ocorre o processamento e a camada de saída fornece o resultado final da rede.

Um dos principais fatores a ser considerado ao se desenhar uma rede é o número de camadas a utilizar, em tese a idéia é que a existência de duas camadas ocultas seja suficiente para resolver qualquer problema, entretanto existem considerações que demonstram que somente uma camada oculta

é suficiente para implementar problemas como Fourier com a mesma qualidade de resolução segundo KASAPIS (2003).



**Figura 6** Estrutura de um *perceptron* multi-camadas demonstrando a camada oculta onde ocorre o processamento, a camada de entrada e a camada de saída. (Fonte: KASAPIS, 2003).

Na prática não existe um número exato de camadas para cada problema, e a resposta para encontrar a melhor relação entre o número de camadas, e a melhor resposta é um tanto empírica, consiste em muita experimentação. Outra tarefa de difícil resolução é a de determinar o número de neurônios ocultos a serem distribuídos nas camadas ocultas de uma rede MLP.

Pode ser utilizada a fórmula conhecida dada por  $O = \sqrt{m \times n}$  dada por FAUSSET (1996) onde,  $m$  é o número de entradas,  $n$  o número de saídas e o resultado o número de neurônios para a camada oculta dado por  $O$ . Entretanto, neste caso também a fórmula não é precisa, a própria literatura indica que a experimentação continua sendo a melhor metodologia para encontrar o número de neurônios ocultos da rede.

De um modo geral pode-se dizer que se existem poucas unidades, existirá um grande erro durante o treinamento que poderá ser gradualmente diminuído com a inclusão de mais neurônios na camada intermediária, este erro diminuirá gradualmente com a inclusão de cada neurônio até que a variação do erro não seja mais relevante no resultado final, então um modelo suficientemente robusto para a solução do problema foi atingido.

Durante o período de treinamento existem vários atributos da rede que devem ser considerados:

**Épocas:** Número de execuções da rede durante o treinamento.

**Taxa de Aprendizado:** A taxa com a qual é corrigido o erro durante a execução da rede.

Durante a execução do algoritmo a taxa de aprendizado normalmente é mantida de forma constante. A performance do algoritmo normalmente é muito relacionada com o uso de uma taxa de aprendizado correta. De uma forma geral, se a taxa de aprendizado for muito elevada o algoritmo pode rapidamente encontrar uma solução, entretanto esta pode não ser a melhor, inclusive pode não ser sequer a correta. Se for muito baixa pode fazer com que o algoritmo leve muito tempo para convergir para um ótimo, aumentando em muito o tempo de execução e tornando o programa inviável.

**Critério de Parada:** Embora nem sempre seja considerado como um item de treinamento, deve-se considerar, ao construir o algoritmo como o mesmo determina a forma de parar a execução.

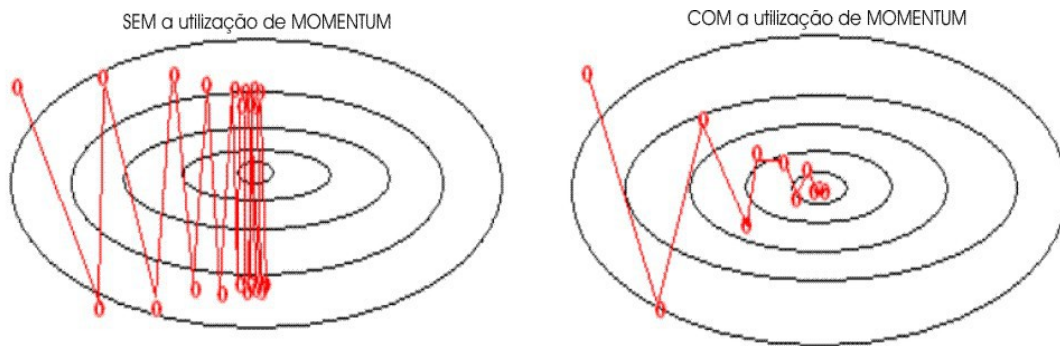
Geralmente não é uma tarefa simples determinar a melhor taxa de aprendizado, mas normalmente esta varia entre 0 e 1. Normalmente é comum experimentar diversas tentativas de diferentes valores para a taxa de aprendizado. Uma metodologia muito utilizada e conhecida é a de utilizar taxas maiores no princípio e diminuí-las ao passo que se aproxima do máximo global desejado.

Exemplificando: Quando existe uma base de dados vasta e confiável é comum que se deseje que o algoritmo encontre o resultado mais rapidamente, por isso a taxa normalmente é alta dado que o sistema pode encontrar facilmente uma resposta correta já que a base é confiável, entretanto, quando a leitura dos dados é mais complexa é desejável que o algoritmo “passeie” por todos os máximos locais de forma a encontrar a melhor solução, portanto a taxa de aprendizado deve ser baixa e a execução mais longa, caso contrário a chance de uma solução em um máximo local e não global torna-se grande.

**Momentum:** Indica a taxa de dependência com que os resultados na execução da rede são influenciados pelo erro ocorrido com a execução anterior.

O termo momentum refere-se a um parâmetro típico do algoritmo de retropropagação do erro. Evidências empíricas mostram que o uso do atributo momentum na execução de uma rede pode apressar a convergência e aumenta a possibilidade de evitar um mínimo local. A idéia é dar ao algoritmo uma certa taxa de “inércia”; baseando-se na variação do erro a cada execução de forma a evitar uma mudança mais drástica nos pesos de cada neurônio se a solução for melhor.

Na Figura 9 é mostrada uma comparação entre a execução utilizando *momentum* durante o aprendizado:



**Figura 7** Utilização do parâmetro momentum no aprendizado da rede neural (Fonte: KASAPIS, 2003).

Importante mencionar que o uso deste atributo melhora o desempenho da execução, mas normalmente resultados similares são conseguidos simplesmente utilizando taxa de aprendizado, mas de forma mais lenta.

De forma contrária por desconsiderar algumas variações de erros o uso do Momentum pode levar a um máximo local.

**Crítério de Parada:** Como todo algoritmo deve ser definido um critério de parada. Várias formas já foram tentadas das quais se sobressaem as seguintes 4:

- Fixar um número de execuções, 20, 200 ou 2000, não existe uma regra exata e sim experiência.
- Configurar um valor aceitável para o erro e parar assim que tiver sido atingido.
- Configurar um valor aceitável para a variação de gradiente, ou seja, entre cada época, verificar se a variação comparando com a época anterior foi inferior a um certo valor e parar ao ser atingido.
- Dividir a base existente em duas, uma de treinamento e uma de validação, parar no momento em que conseguir que a taxa de erro de treinamento seja a mesma de validação. Isto é feito ou em duas execuções ou variando o treinamento quando o erro aumentar mais que um ponto considerado normal.

A melhor forma de validar o funcionamento da rede é testá-la em um terceiro conjunto de dados que não foi utilizado durante o processo.

**Generalização:** Importante característica da rede neural que demonstra a capacidade da rede, após treinada, de resolver novos problemas ainda desconhecidos.