UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
GRUPO DE REDES NEURAIS

Aluno: Fábio Daros de Freitas

SEMINÁRIO:
"INTRODUÇÃO ÀS REDES NEURAIS/APRENDIZADO EM REDES NEURAIS"

1 - INTRODUÇÃO

- 1.1 Histórico
- 1.2 Paradigma
  - Características e potencialidades
    - . Aprendizado
    - . Generalização
    - . Abstração
  - Aplicações
  - Limitações

2 - ARQUITETURAS DE REDES NEURAIS

- 2.1 Neurônio artificial
- 2.2 Perceptron
- 2.3 Redes multi camadas
  - Backpropagation
  - Redes de Hopfield
  - Bidirecional Associative Memories
  - Adaptive Ressonance Theory

3 - APRENDIZADO

- 3.1 Introdução
- 3.2 Aprendizado supervisionado
  - Aprendizado em Perceptrons
  - Aprendizado em redes Backpropagation
  - Métodos estatísticos
- 3.3 Aprendizado não supervisionado
  - Hebbian learning
  - Aprendizado competitivo
  - Self-Organization Map

4 - CONCLUSÃO

5 - REFERÊNCIAS BIBLIOGRÁFICAS

1 - INTRODUÇÃO

1.1 Histórico

Apesar de percebermos nos últimos anos um aumento considerável da pesquisa na área das Redes Neurais Artificiais (RNA), o interesse nesta área não é recente. Os primeiros trabalhos foram publicados na década de 40, motivados pelo avanço das ciências biomédicas na elaboração de modelos do aprendizado humano. Em 1949 Hebb propôs um conjunto de regras de aprendizado que foi o ponto inicial para os algorítmos de aprendizado em RNA. Nas décadas de 50 e 60, pesquisadores como Minsky, Rosenblatt e Widrow desenvolveram redes dotadas de uma única camada de neurônios artificiais chamados *perceptrons*.

Os perceptrons foram aplicados a diversos problemas, como previsão do tempo, e visão artificial. Mas em 1969, Minsky e Pappert provaram que as redes de camada única eram teoricamente incapazes de resolver vários problemas bem simples, como por exemplo, implementar uma função OU-EXCLUSIVO. Estes resultados diminuiram o interesse dos pesquisadores, que optaram por áreas mais promissoras.

Alguns dos pesquisadores que permaneceram na área de RNA, como Kohonen, Grossberg e Anderson, foram aos poucos formulando fundamentos teóricos sob os quais puderam ser construidas e treinadas redes multi-camadas. Estas redes se mostraram capazes de superar as limitações dos perceptrons.

Estes resultados somados a outros fatores tecnológicos levaram à retomada, após duas décadas de silêncio, da pesquisa em redes neurais artificiais na a década de 80. Só no ano de 1987 por exemplo, foram realizadas quatro grandes convenções com mais de 500 papers publicados.

Espera-se para o futuro próximo um melhor entendimento das redes neurais artificiais, principalmente das chamadas biologicamente plausíveis, levando a um acréscimo
no número e modalidade de aplicações em que se pode obter vantagens através desta solução.

1.2 Paradigma

As redes neurais artificiais, ou apenas redes neurais, tentam imitar, em algum nível, o comportamento do cérebro humano. Mais restritamente, uma rede neural é a implementação de um algorítmo inspirado no padrão de funcionamento e organização dos neurônios no cérebro humano. Uma rede neural é composta de unidades simples de processamento interligadas através de conexões ponderadas. Na maioria dos casos, as redes neurais não pretendem imitar a operação exata do cérebro humano, mas sim se utilizar de alguma propriedade bem conhecida do seu funcionamento.
Este paradigma também é conhecido como Modelo Conexionista para inteligência artificial.

1.2.1 Características e Potencialidades

As redes neurais se caracterizam, do ponto de vista da sua arquitetura, basicamente por:

- Apresentar um grande número de elementos de processamento muito simples, que realizam na maioria das vezes apenas as operações de soma, multiplicação e busca em tabelas (função de transferência). Estes elementos são os chamados neurônios artificiais, ou apenas neurônios.
- Um grande número de conexões ponderadas entre estes neurônios, imitando as sinapses biológicas. São os pesos destas conexões que codificam o "conhecimento" da rede.
- Controle distribuído e paralelo, sendo um paradigma apropriado para ser implementado em computadores massivamente paralelos.
- Estrutura apropriada para aprendizagem de novos padrões internos de comportamento.

Baseado nestas características, as redes neurais conseguem apresentar habilidades verificadas no comportamento do cérebro humano, levando a possibilidade de se implementar paradigmas de software capazes, entre outros aspectos, de:

Aprendizado: As redes neurais são altamente adaptivas. Isto é, ao se apresentar um conjunto de entradas à rede, ela pode ajustar os pesos das conexões internas de forma a capturar carcaterísticas predominantes deste conjunto de entrada. Este processo é denominado de "aprendizado". Ao término deste processo, se diz que a rede foi "treinada". Esta característica que tem a rede de "aprender" através de dados que lhe são apresentados é a que traz maior interesse na sua utilização. Também é esta a característica que mais a diferencia das técnicas tradicionais de software, que exigem um conhecimento à priori das funções que serão implementadas.

Generalização: Uma vez treinada, uma rede neural é capaz de processar corretamente dados que diferem ligeiramente dos utilizados em seu treinamento. Esta tolerância a ruídos torna as redes neurais apropriadas para o processamento de informações do mundo real, que são geralmente distorcidas, incompletas ou imersas em ruído.

Abstração: Algumas redes neurais são capazes de capturar as características essênciais de um conjunto de entradas que lhes são fornecidas. Ou seja, após serem treinadas com versões distrocidas de um deteminado padrão, estas redes são capazes de abstrair o padrão original do qual as entradas foram originadas.

Não linearidade: Alguns modelos de redes neurais não são lineares. Com isso elas podem capturar interações complexas entre as entradas que lhe são fornecidas e assim modelar sistemas altamente não lineares sem a necessidade da presença de um modelo analítico destes sistemas.

1.2.2 Aplicações

O número e variedade de aplicações candidatas às redes neurais é enorme. Alguns exemplos típicos são:

- Reconhecimento de padrões em geral, entre eles: Reconhecimento ótico de caracteres (OCR), controle de qualidade da manufatura através da inspeção de peças, sistemas de análise de investimentos, controle de processos, etc...

- Estimação de funcões.
- Problemas de otimização combinatória.
- Processamento digital de sinais.
- Controle de processos industriais.

1.2.3 Limitações

As redes neurais não são uma panacea. Elas são inadequadas para um grande número de aplicações, como por exemplo o cálculo de uma integração numérica. A sua potencialidade é alcançada nos mesmos tipos de tarefas que o cérebro humano se mostra eficaz, tais como o reconhecimento de padrões, memorização e lembrança de fatos e processos envolvendo o uso de inteligência.

Uma das maiores limitações das redes neurais é a sua imprevisibilidade. Como no cérebro humano, um rede neural também pode cometer erros. Este é um aspecto fundamental, pois as redes manifestam não só as virtudes do cérebro humano, mas também as suas imperfeições. Uma das principais imperfeições é a dificuldade que a rede tem de "explicar" como ela resolve um determinado problema. A complexidade dos seus padrões internos após o treinamento dificulta bastante qualquer análise que se queira fazer.

2 - ARQUITETURAS DE REDES NEURAIS

2.1 Neurônio Artificial

O neurônio artificial é um dispositivo que implementa as caraterísticas de primeira ordem do neurônio biológico. Ou seja, é um dispositivo que realiza um somatório das suas entradas, multiplicadas por um peso (valor da sinapse), para obter um nível de ativação em sua saída. Cada uma de suas entradas representa a saída de um outro neurônio.
Existem modelos mais sofisticados e mais biologicamente plausíveis para o neurônio artificial, contudo, este é o modelo utilizado na maioria dos paradigmas de redes neurais.
A figura 2.1 mostra o neurônio artificial.

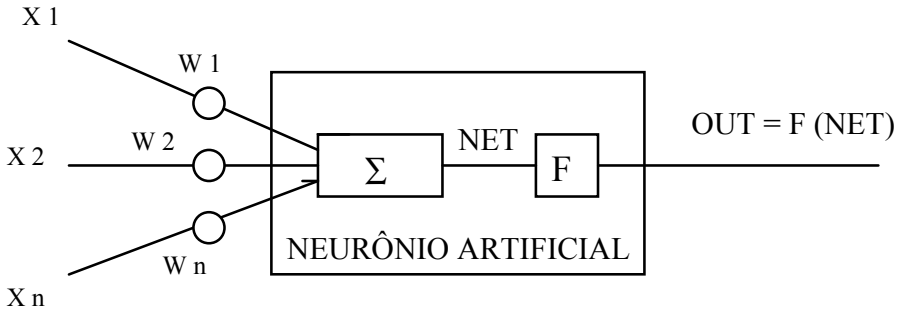

figura 2.1

Observamos pela figura que:

$$NET = W_1X_1 + W_2X_2 + \ldots + W_nX_n$$

Se tratarmos as entradas e os pesos como vetores, veremos NET como o produto escalar entre **X** e **W.** Assim**:**

$$NET = \mathbf{WX}$$

onde:

$$\mathbf{X} = (X_1, X_2, \ldots, X_n) \text{ e } \quad \mathbf{W} = (W_1, W_2, \ldots, W_n)$$

são os vetores de entrada e de pesos respectivamente.
A função F (NET) é chamada *função de ativação* que é aplicada ao sinal NET para gerar o sinal de saída OUT, que é a resposta final do neurônio para os estímulos de entrada. Em alguns modelos, F (NET) é definida como uma função de limiar do tipo:

OUT = *K* (NET), onde *K* é uma função do tipo:

OUT = 1; para NET > T
OUT = 0; caso contrário

onde T a constante de limiar.

Porém, a função de ativação mais utilizada é a função *logística* ou *sigmoidal*, que tem este nome por apresentar um formato no estilo "***S",*** sendo expressa por:

$$F(x) = \frac{1}{(1 + e^{-x})}$$

levando a:

$$OUT = \frac{1}{(1 + e^{-NET})}$$

A figura 2.2 mostra a função sigmoidal.

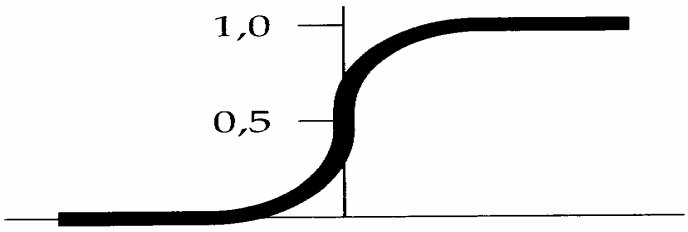

figura 2.2

Outras funcões de ativação também são uilizadas, entre elas a função tangente hiperbólica, que leva a:

OUT = tanh (NET)

A função tangente hiperbólica tem o mesmo formato que a função sigmoidal, porém com imagem entre -1 e 1. Esta função é bastante utilizada pelos neurofisiologistas como modelo matemático da função de ativação dos neurônios.

Um neurônio artificial sozinho é capaz de realizar tarefas simples de reconhecimento de padrões, porém é a interligação de vários neurônios em forma de rede que traz capacidades únicas para a computação neural.

Do ponto de vista da topologia destas redes, elas podem ser de camada única, ou seja, qualquer caminho entre a entrada da rede até a sua saída atravessa apenas um determinado neurônio; ou multi-camada onde o mesmo caminho atravessa mais de um neurônio. Também podem ser do tipo não recorrente, ou *feedforward*, onde um caminho entre a entrada e a saída atravessa um determinado neurônio apenas uma única vez; ou do tipo recorrente, ou *feedback,* onde este caminho pode atravessar um determinado neurônio mais de uma vez através de um tipo de retroalimentação.

A quantidade de neurônios e o número de camadas necessárias para uma rede neural realizar uma tarefa é determinada de forma quase experimental, sujeito basicamante à experiência do projetista. A ausência de uma teoria formal para este dimensionamento é uma das principais deficiências do campo da neuro computação, sendo alvo de pesquisas intensas.

2.2 Perceptron

Os trabalhos pioneiros em redes neurais baseavam-se em um modelo de neurônio como o da figura 2.1 com função de ativação do tipo função de limiar. Estes modelos incluiam sistemas com apenas um neurônio, ou com vários neurônios formando uma rede de camada única como a da figura 2.3. A estes sistemas foi dado o nome de *perceptrons.*

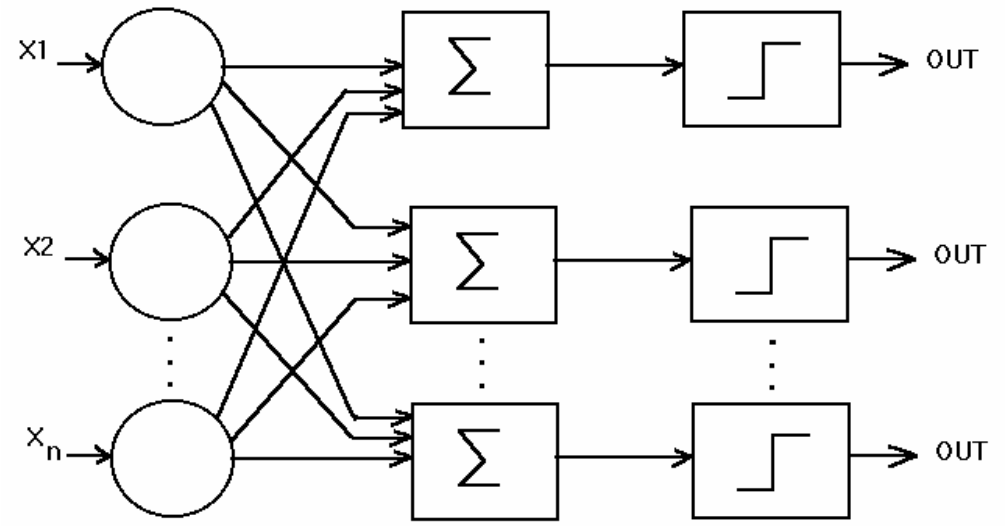

figura 2.3

Durante a década de 60, os perceptrons foram alvo de bastente interesse e geraram um grande otimismo nos pesquisadores. Em 1962 Rosemblatt provou um importante teorema sobre o aprendizado dos perceptrons, que mostra que um perceptron pode aprender qualquer coisa que é capaz de representar; onde "representar" significa a habilidade de simular uma determinada função.

Por algum tempo achou-se ter encontrado a chave para o cérebro artificial, até que a incapacidade dos perceptrons de aprenderem algumas tarefas bem simples culminaram no trabalho de Minsky e Papert, em 1969, provando que há severas restrições nas funções que uma rede de camada única pode representar. O exemplo clássico é o da função OU-EXCLUSIVO, a qual um perceptron é incapaz de representar. O problema é descrito abaixo:

Dado duas entradas binárias x e y, a função OU-EXCLUSIVO é uma função binária cujo resultado é 1 somente quando x =1 ou y = 1 e $x \neq y$, e 0 para os outros casos. Utilizaremos rede neural da figura 2.4 para tentar simular esta função.

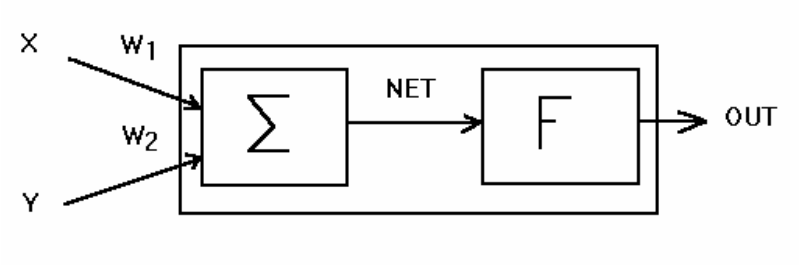

figura 2.4

A funcão F é uma função de limiar em, por exemplo, 0,5. O neurônio então implementa a seguite equação:

$NET = xw_1 + yw_2$.

Igualando esta expressão ao valor de limiar, teremos a seguinte expressão:

$xw_1 + yw_2 = 0{,}5;$

que é uma reta no plano x-y como mostra a figura 2.5

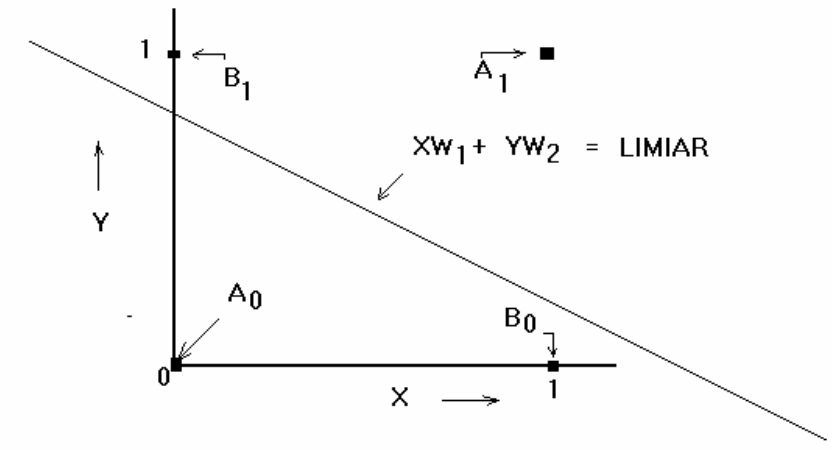

figura 2.5

Onde os pontos A0, A1, B0 e B1 são os estados possíveis das entradas x-y dados por

A0 = (0, 0),
A1 = (1, 1),
B0 = (1, 0),
B1 = (0, 1);

sendo que a função deve retornar 0 para os pontos A_i e 1 para os pontos B_i.
Quando o sinal NET for maior que o valor de limiar, a saída OUT será 1. Isto corresponde, na figura 2.5, ao semi-plano superior à reta de limiar. Analogamente, quando NET for menor que o limiar, OUT estará no semi-plano inferior.
Para o perceptron representar a função OU-EXCLUSIVO, ele deve ser capaz de separar os pontos A_i e B_i, ou seja, a sua reta de limiar deve ser escolhida de forma a manter no seu semi-plano superior os pontos A_i, e no inferior os pontos B_i ou vice versa. Observando mais uma vez a figura 2.5, verifica-se que tal reta não pode ser traçada. Consequentemente este perceptron não consegue representar a função OU-EXCLUSIVO.

Separabilidade linear

O exemplo da função OU-EXCLUSIVO mostra que é impossível se traçar um reta que separe os pontos A_i e B_i no plano x-y de forma que a solução seja correta. Existe um conjunto de funções, como a OU-EXCLUSIVO, que são ditas *lineramente inseparáveis.* Uma função a qual um conjunto de pontos do seu domínio, que corresponde a um determinado valor da sua imagem, pode ser separado geometricamente, é dita *lineramente separável.*
Para os problemas de duas entradas, o separador geométrico é uma reta. Para três um plano. Generalizando, para um espaço de *n* dimensões o separador geométrico é um hiper-plano.

A separabilidade linear restringe as classes de problemas que o perceptron pode resolver. Como é bastante difícil saber se um problema com um número grande de variáveis é linearmente separável, os perceptrons são aplicados somente a problemas simples. A tabela abaixo mostra a variação do número estimado de funções linearmente separáveis para o número de entradas do problema.

Número de entradas	*Número de Funções* 2^{2^N}	*Número de funções L.S.*
1	4	4
2	16	14
3	256	104
4	65.536	1.882
5	$4{,}3 \times 10^9$	94.572
6	$1{,}8 \times 10^{19}$	5.028.134

Fonte: R.O. Windner, *Single-stage Logic* - AIEE 1960

Ao final da década de 60 foi mostrado que o problema da separabilidade linear para os perceptrons pode ser resolvido adicionando-se novas camadas de neurônios às redes. Desta forma as redes podem separar os pontos que estão em uma região convexa, aumentando assim sua capacidade representacional. Resolvendo o problema da função OU-EXCLUSIVO com a rede da figura 2.6 (a) vemos que a reta S1, gerada pelo neurônio superior da primeira camada, divide o plano em dois semiplanos: um contendo o ponto B_1 e outro contendo os demais pontos. Em seguida, o neurônio inferior da primeira camada gera um outra reta, S2, que subdivide o semi-plano inferior de forma a isolar o ponto B_0 dos pontos A_i. A segunda camada realiza um E lógico entre as duas regiões, e assim os pontos A_i ficam agora pertencendo à região convexa aberta limitada pelas retas S1 e S2. A figura 2.6 (b) mostra o plano x-y com o problema OU-EXCLUSIVO devidamente representado.

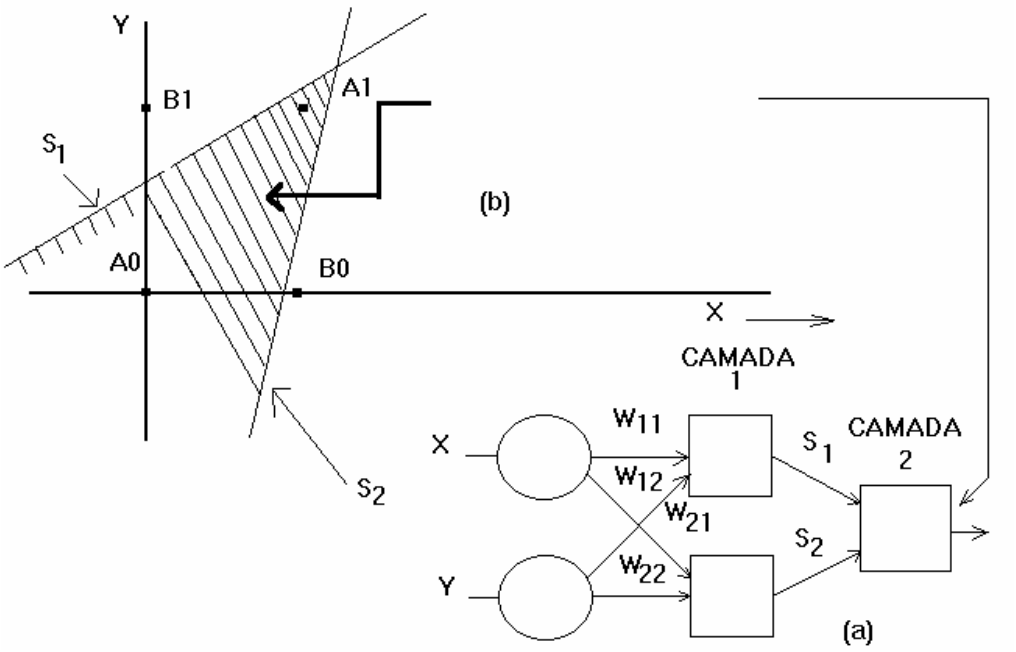

figura 2.6

Desta forma, utilizando-se redes neurais com duas camadas pode-se resolver problemas de classificação em em que os pontos desejados estejam em regiões convexas.

Analogamente é mostrado que utilizando-se três camadas e três variáveis de entrada pode-se representar problemas que envolvam a separação de regiões não convexas, aumentando assim o número de problemas representáveis por estas redes. A figura 2.7 (b) mostra o caso em que duas regiões triangulares, T1 e T2, foram geradas respectivamente pela primeira e segunda camada da rede neural da figura 2.7 (a). O neurônio da terceira camada opera uma função lógica do tipo T1 e não T2, separando assim uma região não convexa no plano.

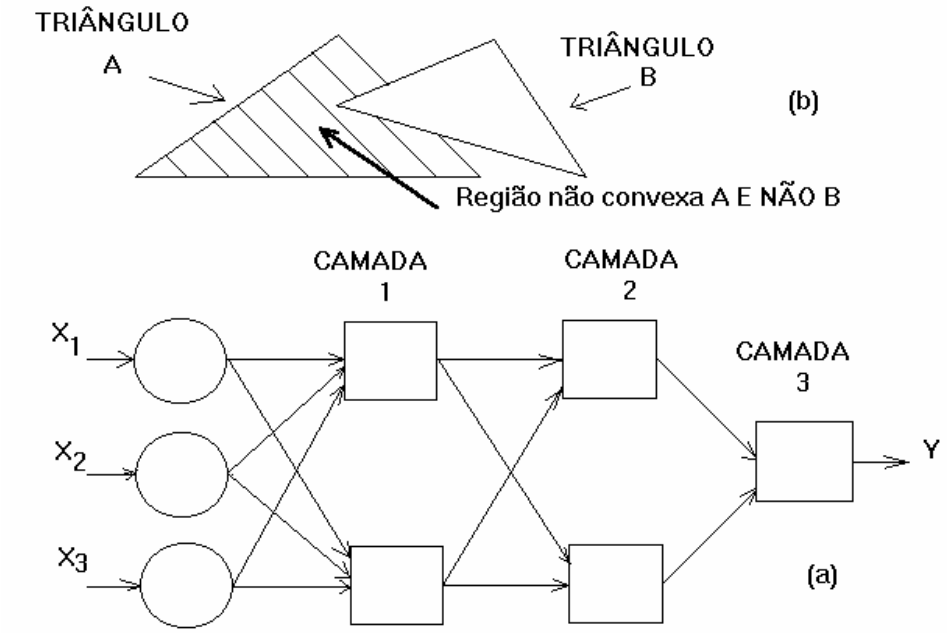

figura 2.7

2.4 Redes Multicamadas

Com o objetivo de se poder resolver problemas mais complexos utilizando as redes neurais, foram formuladas várias arquiteturas de redes multicamadas. Sendo o treinamento de redes multicamadas uma tarefa complexa e fundamental para a utilização desta solução, muitas das arquiteturas de redes multicamadas são projetadas de forma a viabilizar algorítmos de aprendizados específicos. Estas arquiteturas, na maioria das vezes, comprometem a plausibilidade biológica da rede em troca de um treinamento convergente e estável.

Nas redes neurais multicamadas, as camadas que não são a de entrada ou a de saída são chamadas de *ocultas.* A primeira camada é tipicamente uma camada de *fan-in,* e não realiza tarefas computacionais.

As diversas arquiteturas diferem estruturalmente em elementos como o número de camadas, o padrão de interconexão entre as camadas e o modelo de neurônio utilizado. Serão apresentadas em seguida as arquiteturas mais conhecidas de redes neurais.

2.3.1 Backpropagation

As redes do tipo backpropagation são as mais populares entre as redes neurais. Os trabalhos que primeiro descreveram este tipo de redes neurais são de Werbos em 1974, Parker em 1982 e Rumelhasrt, Hinton e Willians em 1986.

Embora o termo backpropagation seja mais utilizado para descrever o algorítmo de aprendizado destas redes, ele também determina sua arquitetura típica. Na figura 2.8 vemos uma rede com três camadas de neurônios: a camada de entrada, cujo os nós não realizam nehuma tarefa computacional, uma camada oculta e a camada de saída. As três camadas são

completamente interligadas com um peso para cada ligação. O neurônio utilizado é equivalente ao da figura 1.1, com função de ativação sigmoidal e disponibilizando o sinal NET para ser utilizado pelo algorítmo de aprendizado.

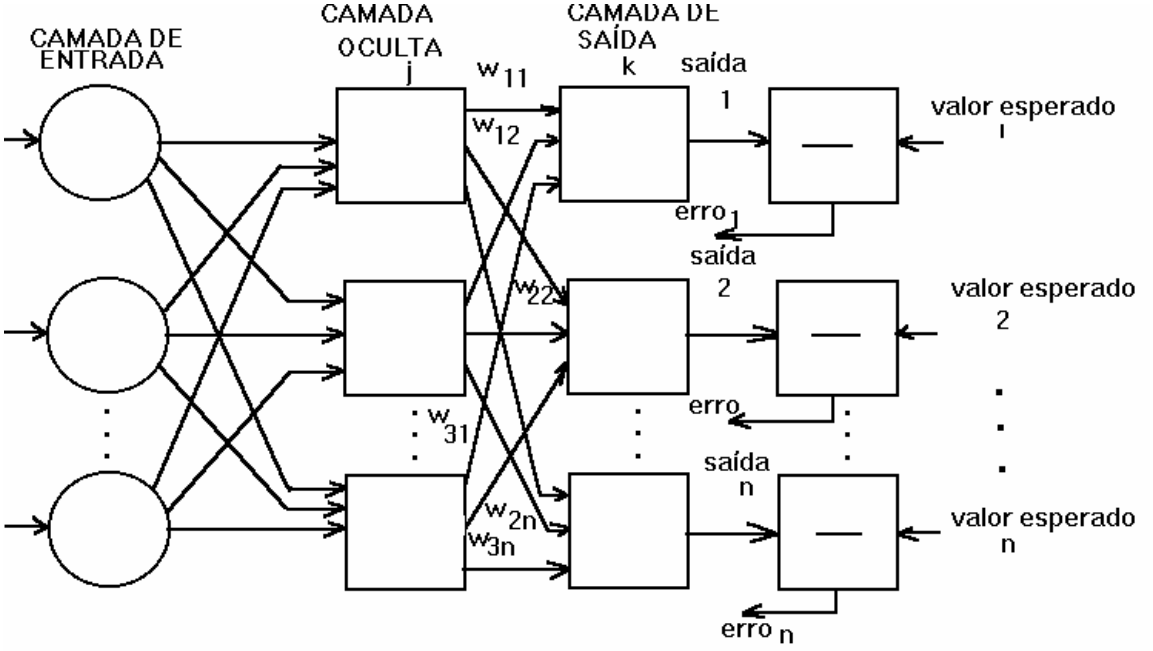

figura 2.8

Durante o aprendizado o vetor de entrada é primeiramente propagado pela rede até a camada de saída. Calcula-se então o erro do vetor de saída comparado ao vetor desejado. Este erro é então retornado à camada oculta através das suas conexões ponderadas. Os pesos da camada oculta são então ajustados segundo algum critério que minimize este erro. Este processo se repete sucessivamente em direção a camada de entrada.

2.3.2 Redes de Hopfield

As redes neurais recorrentes são aquelas em que existem conexões ligando os neurônios da camada de saída aos da camada de entrada. Estas redes apresentam poderosas características de comportamento, que são ausentes nas redes não recorrentes. Por outro lado, elas sofrem de problemas de estabilidade, que não afetam as redes não recorrentes. Como uma rede recorrente é um tipo sistema retroalimentado, ela pode entrar em um modo de oscilação e nunca apresentar um resultado útil em sua saída. Cohen e Grossberg derivaram, em 1983, um teorema que define que se a matriz **W** dos pesos da rede é simétrica e tem diagonal principal nula, então a rede é estável. Também, John Hopfield realizou importantes contribuições na teoria e aplicação destes tipos de redes, levando alguns tipos a serem conhecidos como redes de Hopfield. A figura 2.9 mostra um modelo funcionalmente equivalente aos modelos de Hopfield, e será utilizado para manter compatibilidade representacional com os demais.

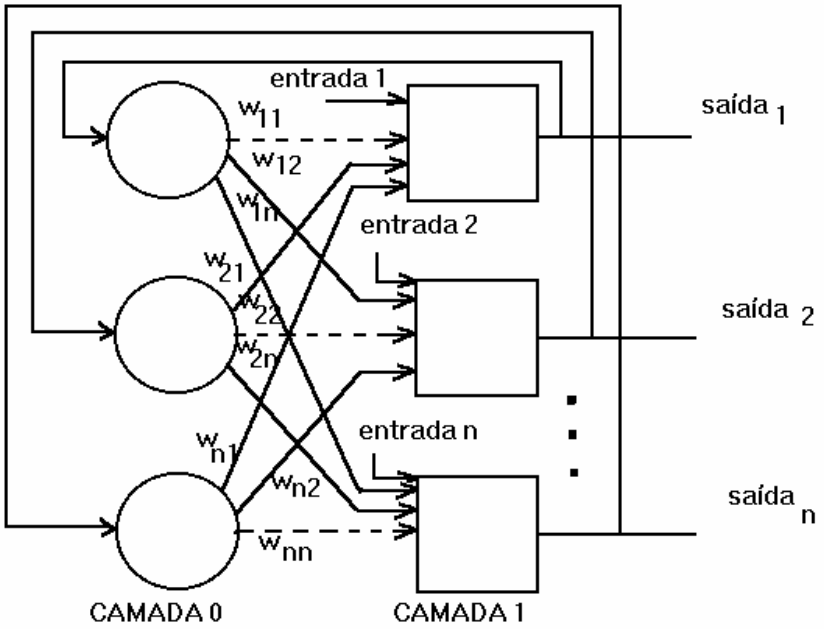

figura 2.9

A camada 0 não realiza tarefas computacionais, servindo apenas para conectar a saída da rede em sua entrada. Cada neurônio ca camada 1 realiza a soma ponderada das suas entradas e aplica o resultado a uma função não linear. Nos primeiros trabalhos de Hopfield, esta função era uma função de limiar. Nos sistemas binários, o estado da rede é dado pelo seu vetor de saída. Assim, uma de rede com dois neurônios na camada de saída, tem seus quatro estados possíveis representados como vértices de um quadrado. Para três neurônios, os estados são representados pelos oito vértices de um cubo. Enfim, para *n* neurônios, os estados estão representrados nos vértices de um hipercubo *n*-dimensional.
Quando um vetor de entrada é aplicado à rede, ela oscila de vértice em vértice até atingir um estado estável. Este vértice final é dado pelos valores dos pesos da rede, o vetor de entrada e o valor de limiar. Se o vetor de entrada estiver parcialmente incompleto ou incorreto, a rede estabilizará no estado mais próximo ao desejado. Este comportamento se assemelha ao comportamento da memória humana, que é uma memória assessável por conteúdo, e não por endereço como nos computadores convencionais. Este tipo de memória chama-se *memória associativa.* Utilizando-se uma função de ativação contínua, por exemplo a sigmoidal, os resultados se assemalham mais aos dos neurônio biológicos.

As redes de Hopfield são de fato memórias *auto-associativas,* isto é, um vetor de entrada pode ser completado ou corrigido, mas não associado a uma memória (vetor) diferente. Isto acontece pelo fato de o vetor de saída aparecer nos mesmos neurônios em que o vetor de entrada é aplicado.

2.3.3 Bidirecional Associative Memories (BAM)

Acresecentando-se mais um camada a uma arquitetura como a da memória associativa vista no ítem anterior, se chega uma memória *hetero-associativa.* Ou seja, o

vetor de entrada é conectado a um conjunto de neurônios e o vetor de saída é obtido em outro conjunto.Como as redes de Hopfield, as BAM também são capazes de generalização e geração de vetores de saída corretos mesmo quando os vetores de entrada estão corrompidos. As versões adaptivas também são capazes de abstração, ou seja, retirar características ideais de padrões imperfeitos ou imersos em ruído. Estes comportamentos se assemalham bastante ao comportamento do cérebro humano.

A figura 2.10 mostra a arquitetura típica de uma BAM.

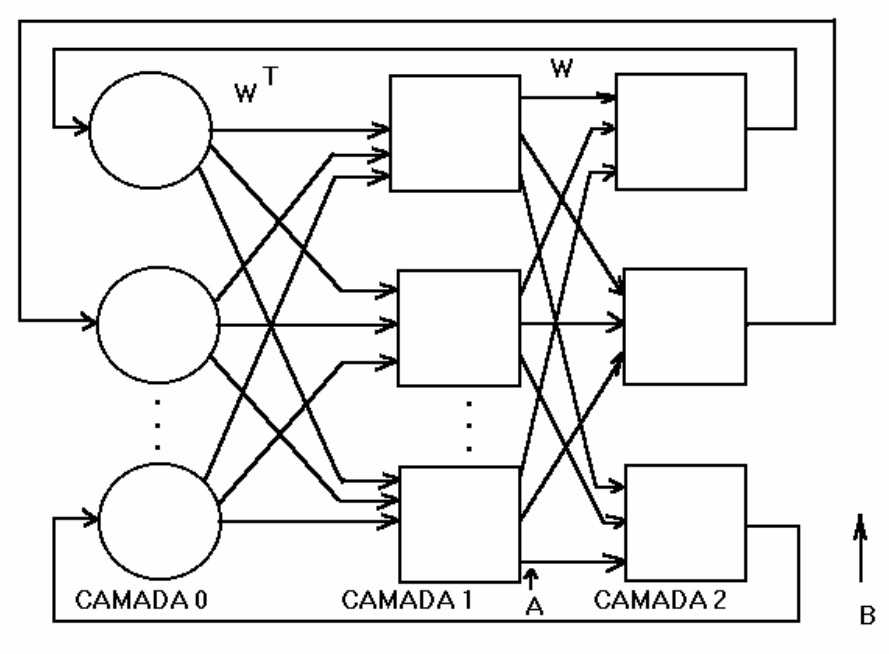

figura 2.10

Um vetor de entrada **A** é aplicado à matriz de pesos **W,** gerando um vetor **B** nos neurônio de saída da camada 2. O vetor **B** é então aplicado à matriz de pesos $\mathbf{W}^T$**,** que é a transposta da matriz **W.** $\mathbf{W}^T$ produz então novas saídas para o vetor **A**, e este processo se repete até a rede atingir um estado estável, onde **A** e **B** não sofrem alterações.

Note que os neurônio das camadas 1 e 2 realizam um somatório ponderado das suas entradas e aplica uma função *F* ao resultado. Assim:

$$bi = F\,(\sum aj\ wij)$$

ou

$$\mathbf{B} = F\,(\mathbf{AW}).$$

Analogamente:

$$\mathbf{A} = F\,(\mathbf{BW}^T)$$

onde $\mathbf{W}^T$ é a matriz transoposta de **W**. A função de ativação é a função sigmoidal.

Cada memória consiste em dois vetores: o vetor **A**, que aparece na saída da camada 1, e o vetor **B**, a memória associada, que aparece na saída da camada 2. Para recuperar uma memória associada, o vetor **A**, ou parte dele, é apresentado momentâneamente na saída da camada 1 e é removido em seguida. A rede começa então a oscilar, produzindo **B** na saída da camada 2. **B** é então processado pela matriz $\mathbf{W}^T$, produzindo uma réplica de **A** na saída da camada 1. A cada ciclo, a rede faz com que os vetores **A** e **B** se tornem cada vez mais

próximos da memória armazenada, levando a rede a atingir um restado estável. Neste ponto, pode-se dizer que se atingiu uma espécie de ressonância, pois a rede oscila sem modificar **A** e **B**. O estados dos neurônios representa a memória de curta duração (*Short Term Memory - STM),* já que eles podem mudar rapidamente quando se aplica um novo vetor **A.** Os valores dos pesos da rede são chamados de memória de longa duração (*Long Term Memory - LTM),* já que eles só mudam através de mecanismos demorados. A estabilidade é garantida de forma análoga às redes de Hopfield.

2.3.4 Adaptive Ressonance Theory (ART)

O sistema ART é um classificador de vetores. Dado um vetor de entrada, a rede o classifica de acordo com a existência de padrões armazenados semalhantes a ele. Se não é encontrado nenhum padrão semelhante, uma nova categoria é criada para este vetor. Quando é encontrado um padrão semelhante, baseado num parâmetro chamado *vigilância*, este é aperfeiçoado. Desta forma, nenhum dos padrões armazenados é modificado quando não reconhecem um vetor de entrada, garantindo a estabilidade da rede.
Uma rede ART típica é composta por duas camadas principais de neurônios, que interagem entre si, e três unidades de controle. A figura 2.11 mostra um diagrama simplificado.

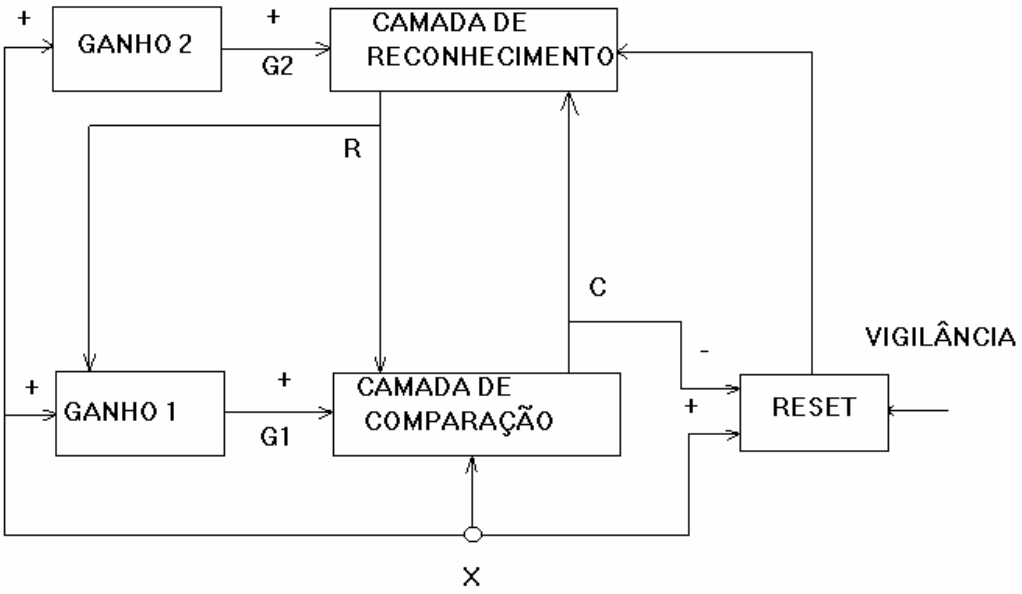

figura 2.11

As duas camadas são chamadas de camada de comparação e camada de reconhecimento. Os controles ganho1 (G1), ganho2 (G2) e reset são utilizados durante o aprendizado e classificação.
Camada de comparação: A figura 2.12 mostra a camada de comparção simplificada com três neurônios. Esta camada recebe como entradas o vetor **X**, o sinal Pj, que é a saída dos neurônios da camada de reconhecimento ponderada pelo vetor **T** e o controle **G1.** Ela produz como saída o vetor **C,** que é a entrada da camada de reconhecimento. Esta camada é regida pela chamada *regra 2/3*, segundo a qual um neurônio só dispara se duas das suas três

entradas forem iguais a 1. Inicialmente G1 é mantido em 1 e **R** = **0**, levando **C** a ser um réplica de **X**. O vetor de pesos **T** é um vetor binário.

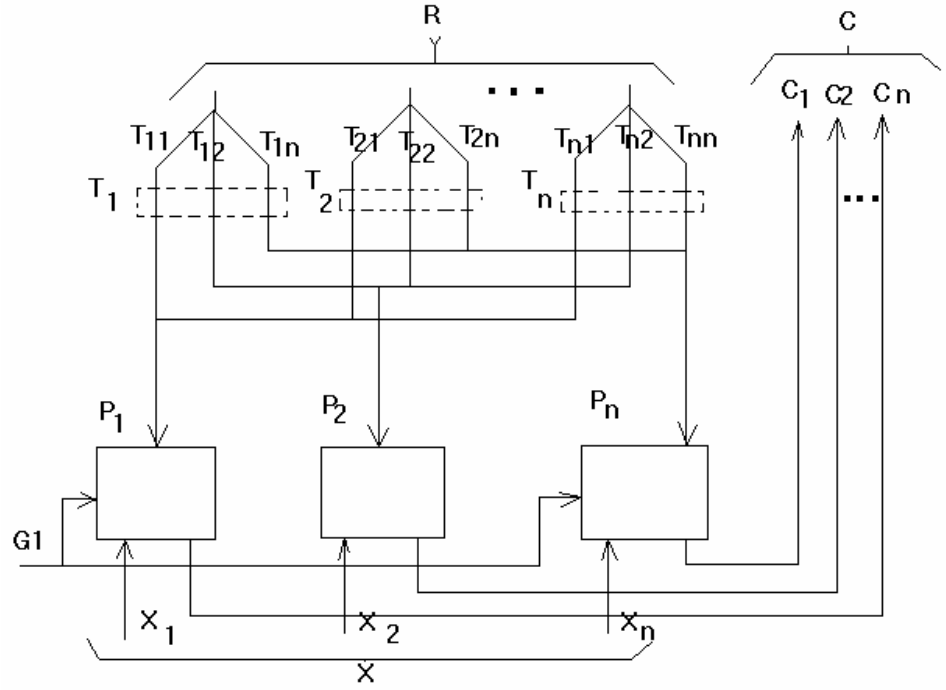

figura 2.12

Camada de reconhecimento: A figura 2.13 mostra uma camada de reconhecimento simplificada com três neurônios. esta camada serve para classificar o vetor de entrada. Cada neurônio tem um vetor de pesos $\mathbf{B}_j$, que é um vetor de valores contínuos. Somente o neurônio cujo vetor **B** mais se assemelha a **C** dispara, isto é, aquele que tem o produto escalar $\mathbf{CB}_j$ maior. Todos os demais são inibidos através de um mecanismo próprio. O vetor $\mathbf{B}_j$ se constitui, então, de um padrão armazenado para um determinada categoria de vetores de entrada. A versão binária deste vetor está no vetor $\mathbf{T}_i$ da camada de conhecimento.

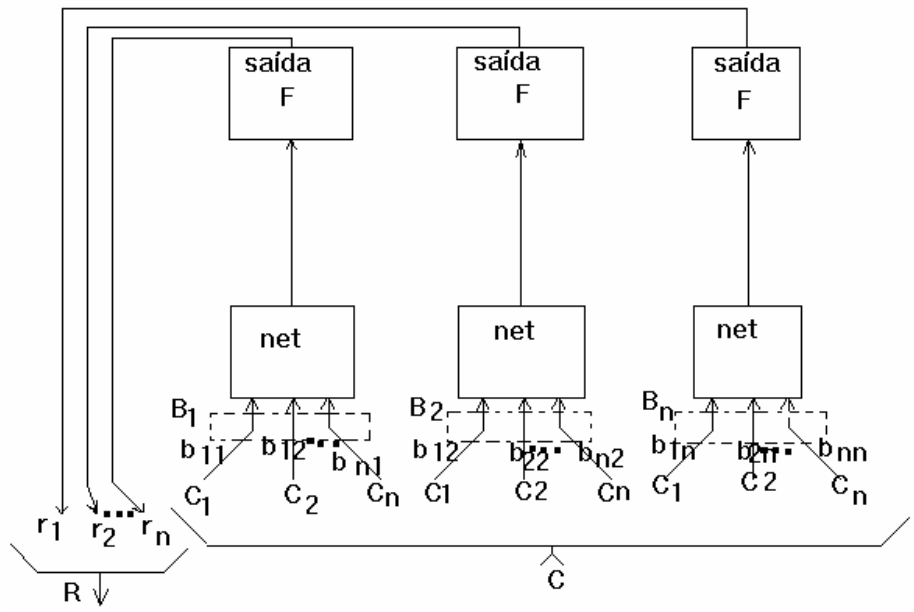

figura 2.13

G2 : O sinal ganho 2 implementa um OU lógico com os componentes do vetor **X**, ou seja, ele vai a 1 se pelo menos um componente de **X** é 1.
G1: O sinal de ganho 1 implementa um OU lógico com os componentes do vetor **X** E um NÃO-OU com os componentes do vetor **R.** Ou seja, ele vai a 1 se existe pelo menos um componente de **X** igual a 1, porém retorna a 0 se algum componente de **R** for igual a 1.
Reset: O sinal de reset mede a similaridade entre **X** e **C.** Se eles diferem abaixo de um parâmetro de vigilância, o sinal de reset inibe o neurônio ativo da camada de reconhecimento. A similaridade entre os vetores é medida dividindo-se o número de componentes igual a 1 do vetor **C** pelo mesmo número correspondente ao vetor **X**.
A operação de classificação dos sistemas ART é dividida em três fases: Reconhecimento, comparação e busca.
1) Reconhecimento: Na ausência de vetor de entrada, **X** = **0**, o sinal G2 vai a 0, desabilitando os neurônio da camada de reconhecimento e fazendo suas saídas serem iguais a 0. O vetor **X** é então aplicado, levando G1 e G2 a 1. O vetor **C** é feito igual a **X**, levando a ativação do neurônio da camada de reconhecimento cujos pesos que mais se assemelham a **C**. Isto faz com que um único componente r_j de **R** seja igual a 1, e o restante igual a 0. Desta forma a camada de reconhecimento tenta "reconhecer" o vetor de entrada **X**.
2) Comparação: O componente r_j gerado ativa então o vetor de pesos $\mathbf{T}_j$, através dos componentes t_{ji}, que gera um sinal p_i para cada neurônio da camada. Como **R** não é mais um vetor nulo, G1 vai a 0. Agora as entradas válidas para a regra de 2/3 na camada de comparação são **X** e **P,** fazendo com que **C** seja um E lógico entre os dois vetores. Se os dois vetores são similares, a rede então "reconheceu"o vetor **X;** se não, o sinal de reset é ativado, desabilitando o neurônio ativo da camada de reconhecimento e a rede entra no modo de busca.
3) Busca: A inibição do neurônio ativo da camada de reconhecimento faz com que **R** volte a ser um vetor nulo, G1 seja igual a 1 e **C** seja igual a **X**. A rede entra em um modo de

aprendizado e o ciclo recomeça, levando um outro neurônio da camada de reconhecimento a disparar e assim sucessivamente passando por todos os outros neurônios. Se ocorre eventualmente um reconhecimento, a rede ajusta os valores de $\mathbf{B}_j$ e $\mathbf{T}_j$ para o neurônio vencedor de modo a "melhorar" o padrão aprendido para o vetor **X;** se não, todos os neurônios da camada de reconhecimento foram verificados e se encontram inibidos, fazendo com que a rede selecione um novo neurônio não utilizado até o momento, e inicialize seus vetores $\mathbf{B}_j$ e $\mathbf{T}_j$ de modo a reconhecer o padrão **X**.

Este sistema ART binário descrito é conhecido como ART-1. Temos ainda os sistemas ART-2, que são capazes de classificar vetores binários e contínuos; os sistemas ART-3, que são sistemas multi-camadas que suportam a hierarquia de representações, e os sistemas Fuzzy ART, que são implementações do ART-1 utilizando fuzzy logic.
A família de sistemas ART foi concebida por Stephen Grossberg e Gail Carpenter.

3 - APRENDIZADO

3.1 Introdução

A importância do aprendizado em redes neurais é tamanha, que foi este o fator responsável pelos quase 20 anos de silêncio na área. Ao final da década de 60, após o trabalho de Minsky e Papert mostrar as limitações dos perceptrons, e apontar as redes multi-camadas como solução, não haviam bons algorítmos para o aprendizado destas redes. O ressurgimento do interesse em redes neurais durante a década de 80 foi graças ao surgimento de poderosos algorítmos de aprendizado.

Entende-se por aprendizado em redes neurais, um processo sistemático através do qual se modifica o valor dos pesos entre as conexões dos neurônios da rede. Visto que a capacidade de uma rede neural de simular uma determinada função (comportamento) é dada pelo estado de sua matriz de pesos **W**, esta matriz é então a responsável pelo correto funcionamento da rede numa dada aplicação. Em aplicações um pouco menos triviais, a matriz **W** assume configurações tão complexas que se torna impossível representa-la por algum processo direto.

O aprendizado em redes neurais pode ser subdividido em dois grandes grupos: supervisionado e não supervisionado. No aprendizado supervisionado, é apresentado à rede além da entrada a saída, desejada; no não supervisionado somente a entrada é apresentada.

Um dos maiores problemas do aprendizado é a sua instabilidade. Uma vez treinada para realizar uma determinada tarefa, a matriz **W** não pode se alterar de forma que a rede não consiga mais realizar a tarefa. Se isto ocorrer, diz-se que o aprendizado é instável.

Nos algorítmos supervisionados, uma das manifetstações da instabilidade no aprendizado é através do "esquecimento" pela rede de um padrão aprendido. Ou seja, alterações realizadas nos pesos da rede durante o aprendizado de um novo padrão cancelam a configuração que reconhecia um padrão anterior. Nos algorítmos não supervisionados, a classificação de um padrão como pertencente a uma categoria em um determinado instante, sendo que anteriormente este padrão pertencia a uma outra categoria, é um dos sintomas da instabilidade do aprendizado.Uma maneira de resolver o problema da instabilidade é parar o aprendizado e salvar os pesos da rede num momento adequado. Desta forma, pode se garantir que o comportamento da rede não mudará com o tempo.

Porém, em muitas aplicações, principalmente aquelas em que a distribuição probabilística das entradas muda muito com o tempo, é desejável que a rede fique continuamente no modo de aprendizado de forma a se adaptar às novas realidades das suas entradas. Desta forma, uma solução mais apropriada para o problema de instabilidade se faz necessária.

É comum a quase todos os algorítmos de aprendizagem, se inicializar os pesos da rede com valores aleatórios num determinado intervalo.

Um outro ponto crítico para o sucesso do aprendizado é a escolha do conjunto de vetores de treinamento. Esta etapa geralmente é a mais longa no desenvolvimento de um sistema utilizando rede neural para uma determinada aplicação. Levantamento e coleta dos dados, análise e seleção de variáveis, pré-processamento dos vetores, etc... Como a rede neural vai "aprender" a modelar o sistema em questão através dos dados, os vetores de treinamento são o ponto crucial para o sucesso de uma aplicação. O número de neurônios das camadas depende do número de variáveis que compões o vetor de entrada. A escolha destas variáveis deve seguir critérios de relevância dentro do sistema, por exemplo, dentre inúmeros sensores disponíveis em uma planta, deve-se selecionar os pontos de medição mais apropriados para se controlar um determinado processo; e descorrelação, ou seja, se duas variáveis escolhidas têm forte correlação estatística, basta utilizar um delas no sistema.

Veremos a seguir vários algorítmos de aprendizado agrupados por suas modalidades.

3.2 Aprendizado Supervisionado

Nesta modalidade de aprendizado, são fornecidos à rede o vetor de entrada e o vetor que se deseja obter na saída quando a rede for estimulada por aquela entrada. Estes dois vetores formam um *par de treinamento.* Após se apresentar o vetor de entrada de um determinado par à rede, verifica-se a diferença existente entre o vetor gerado pela rede e o desejado. Escolhe-se então os pesos que serão modificados de forma a minimizar esta diferença e se produzir o reseltado desejado.

3.2.1 Aprendizado em Peceptrons

Como vimos anteriormente, o aprendizado do perceptron consiste no posicionamento da sua reta ou superfície de separação de forma que as áreas contendo pontos associados a um estado da rede sejam isoladas umas das outras.

O princípio do aprendizado é o seguinte: Se o perceptron dispara quando não devia, diminua de cada peso w_i uma quantidade proporcional a x_i. Caso contrário, se ele não dispara quando deveria, aumente cada peso w_i da mesma quantidade anteior.

Formalizando o algorítmo de treinamento do perceptron, temos:

1 - Aplique o vetor de entrada e calcule a saída.
2 - a) Se a saída estiver correta, vá para 1
b) Se a saída estiver incorreta e o neurônio não disparou, some cada entrada x_i multiplicada por uma constante de aprendizado ao seu peso correspondente w_i; ou
c) Se a saída estiver incorreta e o neurônio disparou, subtraia cada entrada x_i multiplicada pela constante de aprendizado do seu peso correspondente w_i.
3 - Vá para 1

Desta forma temos que:

$$w_i(n+1) = w_i(n) + \delta x_i$$

Uma generalização para vetores contínuos e a chamada *Regra Delta*:
Seja **T** o vetor esperado, e **A** a saída obtida, então calcula-se a diferença:

$$\delta = (\mathbf{T} - \mathbf{A}),$$

então o passo 2a corresponde a $\delta = 0$; o 2b corresponde a $\delta > 0$ e o 2c a $\delta > 0$.
Fazendo

$$\Delta = \eta \delta x_i,$$

teremos:

$$w_i(n+1) = w_i(n) + \Delta x_i$$

onde n é o passo pela rede, e η o coeficiente de aprendizado.
Um dos maiores problemas do aprendizado de perceptrons é saber, à priori, se os dados são linearmente separáveis ou não. As entradas de muitos sistemas reais podem ser linearmente separáveis em um instante e em outro não em outro posterior. Também não se sabe quantos passos o perceptron levará para treinar, pois não há um postulado deste tipo no algorítmo. Devido a estes problemas, em alguns casos prefere-se treinar os pesos diretamente, sem a utilização de um algorítmo.

3.2.2 Backpropagation

O algorítmo de backpropagation é um dos mais populares nas redes neurais. Os passos do algorítmo podem ser resumidos nos seguintes:

1 - Selecione um par do conjunto de treinamento e aplique a entrada à rede.
2 - Calcule a saída da rede.
3 - Calcule o erro entre a saída da rede e o vetor esperado.
4 - Ajuste os pesos da rede de forma a minimizar este erro.
5 - Repita os passos de 1 a 4 para cada par do conjunto de treinamento até que o erro para todo o conjunto esteja em um limite aceitável.

Os passo 1 e 2 são similares ao modo de utilização futuro da rede, e são conhecidos como *passo direto* . O cálculo das saídas é realizado camada a camada.
Os passos 3 e 4 são utilizados para o "aprendizado"da rede, e são chamados de *passo reverso.* Tomemos como referência a figura 2.8
Passo direto: Sejam **X** o vetor de entrada do par, **T** o vetor desejado do par e **Y** o vetor de saída produzido pela rede. Seja ainda **W** a matriz de pesos entre duas camadas da rede e **N** o vetor dos sinais NET dos neurônios. Então temos que:

$\mathbf{N} = \mathbf{XW}$ e
$\mathbf{O} = F(\mathbf{XW})$,

onde **O** é o vetor das saídas OUT dos neurônios da camada mais à direita, e F é a função de ativação do tipo sigmoidal.
O vetor de saída **O** de uma camada é a entrada para a próxima, e o mesmo processo deve continuar até alcançar a camada de saída.
Passo reverso: O passo reverso se subdivide nas etapas de ajuste dos pesos das camadas de saída e das ocultas.

Ajuste dos pesos da camada de saída: A saída do neurônio *k* é subtraída do valor desejado para produzir o erro. Este erro é multiplicado pela derivada da função de ativação calculada par a aquele neurônio, levando a

$\delta_k = OUT_k (1 - OUT_k) (\mathbf{T}_k - OUT_k)$, onde $\mathbf{T}_k$ é o valor desejado para o neurônio *k*.

Então δ_k é multiplicado por OUT_j e o produto é multiplicado por um coeficiente de aprendizado η tipicamente entre 0,01 e 1,0. O resultado é somado ao peso, levando a

$$\Delta w_k = \eta \delta_k OUT_j \text{ , e}$$

$$w_k (n + 1) = w_k (n) + \Delta w_k$$

que é o novo valor do peso w da camada k. Este cálculo é feito para cada neurônio da camada k, utilizando o valor OUT_jdo neurônio da camada j que se conecta a ele.

Ajuste dos pesos das camadas internas: O treinamento das camadas internas é feito propagando-se o erro da camada de saída para trás, camada a camada através da rede, realizando os ajustes nos pesos de cada camada até atingir a camada de entrada.

Primeiro, δ_k é calculado para cada neurônio da camada de saída da mesma forma que é feito para a camada de saída. Então, para um determinado neurônio da primeira camada oculta que se conecta a vários neurônios da camada de saída, temos que os diversos δ_k destes neurônios de saída são propagados pelas conexões ponderadas (multiplicados pelo peso w_{jk} da conexão) e são somados. Em seguida, multiplica-se esta soma pela derivada da saída OUT do neurônio. Assim temos:

$$\delta_j = OUT_j (1 - OUT_j) (\Sigma \delta_k \ w_k),$$

onde δ_k é calculado para cada neurônio de saída e w_k é o peso da conexão entre cada neurônio de saída e o neurônio da camada oculta para o qual se calculou δ_j. Então, os pesos que se conectam ao neurônio da camada oculta pela esquerda (i.e. entre as conexões deste neurônio e neurônios da camada oculta anterior) são ajustados com as seguintes expressões:

$$\Delta w_j = \eta \delta_j OUT_i \ , \text{ e}$$

$$w_j (n + 1) = w_j (n) + \Delta w_j$$

Este processo se repete até que todas as camadas ocultas estejam treinadas.

Existem alguns melhoramentos propostos para este algorítmo. Por exemplo, Stornetta Huberman descreveram, em 1987, alterações para o método. São elas:
Mudar o range de entrada para $\pm 1/2$, e polarizar a função sigmoidal para ter saída entre $\pm 1/2$. Estas simples alterações melhoraram o tempo de convergência do algorítimo em 30 a 50% em média.

3.2.3 Métodos Estatísticos

Os métodos estatísticos são uma alternativa para o treinamento de redes neurais que apresentam algumas vantagens, como por exemplo a habilidade de escapar de um mínimo local durante a convergência do treinamento. Estes métodos fazem pseudo-mudanças nos valores dos pesos da rede, retendo aquelas que resultam em melhorias.
Os passos do método podem ser resumidos nos seguintes:

1 - Aplique um conjunto de entradas e calcule as saídas da rede.
2 - Compare estas saídas com as saídas desejadas e calcule uma medida das diferenças. Uma medida utilizada é a soma dos quadrados das diferenças entre cada componente dos vetores. O objetivo do treinamento então, é minimizar esta diferença, chamada de *função objetivo.*
3 - Selecione um peso aleatoriamente, e altere seu valor de uma quantidade aleatória.
Se este ajuste melhorou a função objetivo, retenha-o, se não desfaça o ajuste.
4 - Repita os passos 1 a 3 até que a rede esteja treinada dentro do desejável.

Este processo tende a encontrar uma solução, mas pode ser enganado ao atingir um *mínimo local.* Quando se esta minimizando uma função, e se chega a um determinado valor cuja as alterações nas variáveis de controle, dentro de um limite definido, não melhoram a solução, diz-se que chegou a um mínimo para esta função. Porém, se existe um valor ainda menor que este resultado na imagem da função, este mínimo é local, e não global. Assim, a função não foi minimizada, e o algorítmo foi "enganado"por este resultado.

Uma forma de resolver este problema é variar a amplitude das alterações. Primeiro, alterações aleatórias de grande amplitude são realizadas, retendo as melhorias. Diminui-se gradativamente a amplitude das alterações, e desta forma se alcança, eventualmente, um mínimo global. Este processo é chamado de *simulated annealing*, pois ele imita o comportamento do resfriamento de metais.

A aplicação este método no treinamento de redes neurais é conhecido como *Boltzmann Training*. Para cada alteração dos pesos que melhore a função objetivo, faça:

1 - Calcule a probabilidade de se aceitar esta alteração pela expressão

$P(c) = \exp(-c / kT)$, onde

.k é uma constante análoga à constante de Boltzmann escolhida para o problema.

. T é a temperatura artificial.

2 - Selecione um número aleatório dado por uma distribuição uniforme. Se P (c) for maior que este número, retenha o valor; se não, desfaça a alteração.

Este mecanismo permite que o sistema se direcione ocasionalmente rumo ao mínimo global, e também que escape de eventuais mínimos locais.

Existem outros métodos utilizam outras distribuições de probabilidade, apresentando vantagens em algumas aplicações.

3.3 Aprendizado Não Supervisionado

Esta modalidade de treinamento caracteriza-se pela ausência do par de treinamento. Somente o padrão de entrada é apresentado à rede, que consegue extrair as caraterísticas relevantes do conjunto de padrões de treinamento e classifica-los de forma adequada. Como não é apresentado nenhum padrão de saída durante o treinamento, não se sabe, à priori, qual neurônio de saída será selecionado para uma determinada categoria.

3.3.1 Hebbian Learning

O trabalho de D.O. Hebb ao final da década de 40 serviu de inspiração para a maioria dos algorítmos de aprendizado em redes neurais. Seu estudo apresentou uma hipótese sobre como ocorriam as alterações locais entre os neurônios sem o auxílio de nenhum outro mecanismo externo. Hebb propôs que uma sinapse que conecta dois neurônios é fortalecida sempre que estes dois neurônios disparam.

O algorítmo Hebbian Learning realiza alteraçòes nos pesos das conexões entre neurônios segundo a expressão:

$$w_{ij}(t+1) = w_{ij}(t) + NET_i\ NET_j$$

Para as redes multi-camadas obterem vantagens representacionais sobre as redes de camada única, uma função de ativação não linear deve ser introduzida nos neurônios. Uma variação do algorítmo de Hebb para esta situação é:

$$w_{ij}(t+1) = w_{ij}(t) + OUT_i\, OUT_j$$

para

$$OUT = \frac{1}{(1 + e^{-NET})}$$

Existe também uma variação diferencial do algorítmo de Hebb que é dada por:

$$w_{ij}(t+1) = w_{ij}(t) + [OUT_i(t) - OUT_i(t-1)]\,[OUT_j(t) - OUT_j(t-1)]$$

onde as alterações dos pesos é em função das variações prévias das saídas.

3.3.2 Aprendizado competitivo

O aprendizado competitivo é um algorítmo de categorização que se comporta como um detetor de carcaterísticas de padrões. A idéia básica é fazer com que diante de um padrão apresentado à rede, os neurônios compitam entre si de levando o vencedor a ser o representante da categoria.

O aprendizado competitvo se dá em redes multicamadas. O número de neurônios da camada de saída geralmente é dado pelo número de categorias que se pretende classificar.
O princípio de funcionamento do algorítmo é o seguinte: Aplica-se um padrão à entrada da rede, este padrão é propagado para a camada de saída. Entre os neurônios da camada de saída, aquele que tiver a maior excitação NET será o vencedor e terá 1 na sua saída. Todos os outros neurônio permanecem com a saída em 0. Este comportamento é chamado de WTA (*Winner Take All)*. O neurônio vencedor terá seus pesos alterados de forma a reforçar sua habilidade para reconhecer este padrão de entrada.

Especificando o algorítmo temos:

1 - INICIALIZAÇÃO
Os pesos são inicializados com valores aleatórios de forma que para cada neurônio $\sum w_{ij} = 1$

2 - ENTRADA
O vetor **X** é apresentado à entrada da rede.

3 - COMPETIÇÃO
Os neurônio de saída competem entre si. Apenas o neurônio com NET máximo vence e tem saída igua a 1. Os demais ficam com saída 0. Ou seja:
$OUT_i = 1$, e $OUT_j = 0$, para $i \neq j$.

4 - ADAPTAÇÃO
Os pesos do neurônio vencedor *j* são alterados da seguinte forma:

$$w_{ij}(t+1) = w_{ij} + \eta \left(\frac{OUT_J}{N_k} w_{ij}(t) \right)$$

onde η é a constante de aprendizado e N_k é o número de neurônios ativos. Esta divisão é para se ter alterações proporcionais a ativação relativa dos neurônios e garantir que $\sum w_{jj} = 1$

5 - LOOP
Repita os passos 2 a 5 até que se classifique o conjunto de treinamento.

A interpretação geométrica deste algorítmo é a seguinte:
Alterando a propriedade $\sum w_{jj} = 1$ para $\sum (w_{jj})^2 = 1$, para redes com três variáveis, teremos os vetores de pesos normalizados. Podemos então ver estes vetores como vetores unitários em um esfera unitária. No início do aprendizado, estes vetores estão indicando pontos distribuídos aleatóriamente na superfície da esfera. Um vetor de entrada que indica um determinado ponto na esfera. Quando se treina os pesos, o efeito é o deslocamento do ponto indicado pelos pesos do neurônio vencedor para o centro da cateogoria que o vetor de entrada representa. A figura 3.1 ilustra esta interpretação.

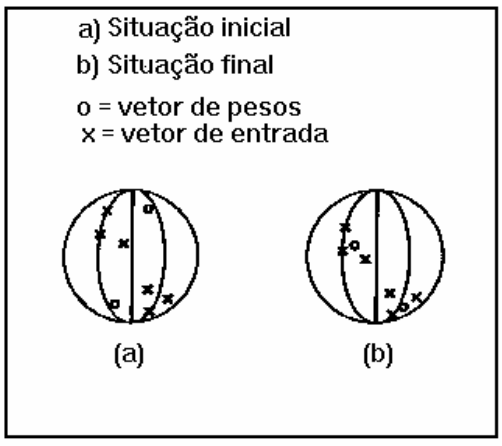

figura 3.1

3.3.2 Self Organization Map

Este algorítmo de classificação foi inventado por Kohonen em 1984, e é utilizado no reconhecimento de padrões. O princípio de funcionamento é parecido com o aprendizado competitivo. Após se apresentar um padrão à rede, os neurônios da camada de saída competem entre si calculando qual deles tem a menor distância geométrica com relação ao padrão (já que os vetores são normalizados). O vetor de pesos do neurônio vencedor se torna o centro de uma região que será treinada para reconhecer o padrão de entrada.
Uma arquitetura típica destas redes, e caracterizada por um arranjo bidimensional dos neurônios de saída como mostra a figura 3.2, e a figura 3.3 mostra uma possível região treinada para um neurônio i.

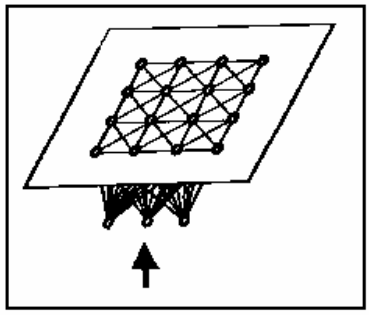

figura 3.2

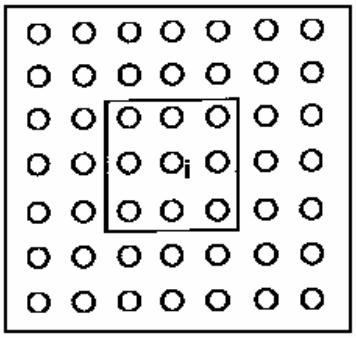

figura 3.3

O algorítmo pode ser formalizado da seguinte forma:

1 - Aplique o vetor **X** à entrada da rede.
2 - Calcule a distância entre o vetor **X** e os vetores de pesos de cada neurônio através da seguinte expressão:

$$D_j = \sqrt{\sum_i (x_i - w_{ij})^2}$$

onde x_i é o componente i do vetor x, e w_{ij} é o peso da conexão da entrada i para o neurônio j.

3 - O neurônio com a menor distância D_j de **X** é o vencedor. O vetor de pesos $\mathbf{W}_c$ então se torna o centro do grupo de vetores que estão a uma distância *D* de $\mathbf{W}_c$.

4 - Treine este grupo de vetores com de acordo com a expressão:

$$\mathbf{W}_j (t + 1) = \mathbf{W}_j (t) + \alpha [\mathbf{X} - \mathbf{W}_j (t)]$$

para todos os vetores de pesos a uma distância *D* de $\mathbf{W}_c$, onde α é a constante de aprendizado.

5 - Repita os passos 1 a 4 para todo o conjunto de treinamento.

Kohonen recomenda que se comece com *D* sendo a maior distância entre os vetores de pesos e termine pequeno o suficiente para que somente um neurônio seja treinado; e α comece próximo a 1 e seja reduzido até 0,1. O número de ciclos de treinamento deve ser da ordem de 500 vezes o número de neurônio de saída.
Após o treinamento, o reconhecimento é feito aplicando-se o padrão de entrada, calculando as excitações e selecionando o neurônio com o maior valor de excitação.
Algumas características topográficas são verificadas, entre elas:
1) As relações de distância entre os vetores **X** são preservadas no mapa, ou seja, categorias mais pareceidas estão localizadas mais próximas umas das outras no mapa.
2) Se os diferentes vetores **X** aparecem com frequências diferentes, os mais frequêntes são mapeados em regiões maiores, e os menos frequêntes em regiões menores.

4 - CONSIDERAÇÕES FINAIS

As redes neurais artificiais demostram ser uma tecnologia bastante poderosa. Os novos modelos de redes e neurônios têm buscado cada vez mais a plausibilidade biológica. Verifica-se, nestes modelos, comportamentos cada vez mais parecidos com os do cérebro humano, por exemplo, alguns tipos de redes parecem "adormecer", ou sofrer de "ataques epiléticos" diante de situações específicas.
A integração destes modelos de I.A. conexionista com os modelos simbólicos, e o surgimento de hardware específico (i.e. massivamente paralelo) permitem a criação de sistemas bastante poderosos, com mercados potenciais da ordem de bilhões de dólares.

Embora a construção do cérebro artificial ainda esteja distante, o entendimento gradual de algumas das suas funções podem acelerar bastante esta caminhada. E isto tem ocorrido cada vez mais. Desta forma, espera-se para o futuro próximo grandes avançosa nesta área de conhecimento.

5 - BIBLIOGRAFIA

- Wasserman, Philip D., *Neural Computing - Theory and Prctice*, VAN NOSTRAND REINHOLD, 1989
- Rich, E. e Knigth, K., *Inteligência Artificial*, 2ª edição, Makron Books, 1993
- Hammerstom, D., "*Neural Networks at Work"*, *IEEE Spectrum*, pp 26-32, junho 1993
- Hammerstom, D., "*Working with Neural Networks"*, *IEEE Spectrum*, pp 46-53, julho 1993
- Pessoa, Luiz Adauto F.C. e Mendes, Sueli B.T., "*Estudo e Avaliação de Algorítmos de Aprendizado Não Supervisionado em Redes Neuronais"*, *Revista Brasileira de Computação* pp 3-22, vol. 6 n. 4, junho 1991.