Universidade Federal do Rio Grande do Sul
Instituto de Informática
Curso de Pós-Graduação em Ciência da Computação

Introdução Às Redes Neurais Artificiais

André Cardon
Daniel Nehme Müller

Philippe Navaux
Orientador

Porto Alegre, novembro de 1994

Sumário

Capítulo 1

Introdução

1.1 Histórico

O primeiro trabalho sobre Redes Neurais foi realizado no ano de 1943, por McCullock e Pitts [EBE 90]. Neste, eles desenvolveram um estudo sobre o comportamento do neurônio biológico, com o objetivo de criar um modelo matemático para este. As conclusões desta pesquisa foram de extrema importância para a futura implementação computacional do neurônio formal:

- a atividade do neurônio é *tudo ou nada*;
- a atividade de qualquer sinapse inibitória previne a excitação do neurônio naquele instante.

A primeira afirmação significa que o neurônio estará no estado ativado se a sua saída ultrapassar um valor limite, caso contrário, ficará no estado de repouso (este princípio originou a função limiar). Entende-se por estado ativado transmitir a saída (transmissão) a outros neurônios da rede.

Já a segunda afirmação teve importância na construção do neurônio formal a partir do conceito de pesos, ou seja, cada entrada do neurônio terá um valor associado; caso este seja positivo, tenderá a excitar a célula; e caso ele seja negativo, tenderá a inibir.

Em 1949, Donald O. Hebb no livro The Organization of Behavior definiu o conceito de atualização de pesos sinápticos. Hebb deixou com seu estudo quatro pontos importantes [EBE 90]:

1. numa rede neural a informação é armazenada nos pesos;
2. o coeficiente de aprendizagem é proporcional ao produto dos valores de ativação do neurônio;
3. os pesos são simétricos (o peso da conexão de A para B é igual ao da de B para A);

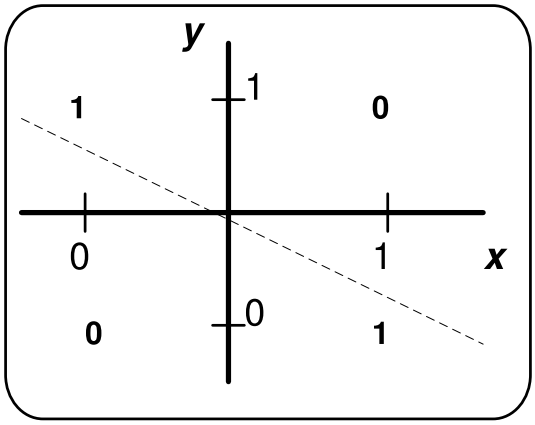

Figura 1.1: **Função XOR:** uma representação gráfica.

4. quando ocorre o aprendizado os pesos são alterados.

Porém, apesar de todos os estudos feitos até então, o primeiro modelo de rede neural implementado foi o **perceptron**, por Frank Rosenblatt em 1958.

O perceptron é uma rede neural simples: constitui-se de uma camada de entrada e uma camada de saída. A cada entrada existe um peso relacionado, sendo que o valor de saída será a soma dos produtos de cada entrada pelo seu respectivo peso. Como definido por McCullock e Pitts, o neurônio possui um comportamento *tudo ou nada*, logo, será necessário estabelecer uma função limiar que defina quando o neurônio estará ativo ou em repouso.

O trabalho de Rosenblatt também estabeleceu a base para os algoritmos de treinamento de modelos não su pervisionados como o de Kohonen e para modelos supervisionados como o backpropagation [EBE 90]. Um modelo é dito supervisionado quando treina-se um modelo para uma saída pré-determinada, que define um dado padrão de entrada.

Apesar do impacto que teve o perceptron na comundade de Inteligência Artificial, este modelo foi fortemente criticado no livro de Minsky e Papert, ***Perceptrons*** . No livro os autores citam o exemplo de que o Percetron com uma simples camada não poderia simular o comportamento de uma simples função XOR (ou-exclusivo) [WAS 89]. Após a publicação desta obra, iniciou o período conhecido como *anos negros* das redes neurais, pelo fato desta ter desencorajado novas pesquisas e desestimulado o estudo mais profundo deste campo. A seguir segue uma breve explicação do problema do XOR.

O perceptron é capaz de aprender tudo aquilo que ele consegue representar [WAS 89]; a partir disso surge a questão: o que ele consegue representar? O perceptron com somente duas camadas consegue representar toda função linearmente separável.

Suponhamos um plano x**o**y onde x e y são as entradas da rede e o ponto cartesiano (x,y) é o valor da respectiva saída, como mostra a figura 1.1.

Como é possível perceber na figura 1.1, não podemos traçar uma única reta (função linear) tal que divida o plano de maneira que as saídas com valor 0 ficam situadas de um lado da reta e as com valor 1 do outro. Entretanto, este problema pode ser solucionado com a criação de uma camada intermediária na rede e graficamente com uma estrutura em três (ou mais) dimensões.

Em 1972, Teuvo Kohonen da Universidade de Helsinky definiu um novo modelo de rede neural, conhecido como **mapa auto-organizável de características** . Ao contrário do perceptron, ele não era limitado a valores binários, mas os valores das entradas, dos pesos e das saídas poderiam ser contínuos. Além disso, ao invés da saída da rede ser representada pela atuação de um simples neurônio *vencedor* (ativado), Kohonen estabeleceu um grande número de neurônios para representar o padrão de entrada, sendo que este número é alcançado pela influência do neurônio *vencedor* aos seus vizinhos. Em outras palavras, não há apenas um neurônio responsável pela representação do padrão de entrada, mas sim um conjunto de neurônios que interagem entre si.

Todavia, a grande importância do trabalho de Kohonen foi ter introduzido um novo paradigma no estudo das Redes Neurais: o aprendizado não-supervisionado, segundo o qual a rede não precisa ficar sendo comparada constantemente a um valor de saída durante a fase de aprendizado. Até o início da década de 80, os modelos de redes neurais implementados poderiam ser classificados em dois tipos: *feedback*, onde a rede, partindo de um estado inicial , chega ao estado final de aprendizado através de iterações, na qual as saídas são canalizadas para as entradas e *feedforward*, a qual transforma um conjunto de sinais de entrada em sinais de saída. O modelo de Kohonen veio a ser um novo paradigma, alternativo, para as redes neurais.

Em 1982 houve uma nova evolução nos trabalhos das redes neurais iniciado pela criação do **modelo de Hopfiled**, desenvolvido pelo físico John Hopfield. Este modelo se caracteriza por ser do tipo *feedback*, isto é, há uma conexão das entradas com as saídas. Por este motivo, estas redes dificilmente chegam a um estado instável, ou seja, chegará um momento em que a saída, após oscilar entre alguns valores binários, será sempre a mesma para um mesmo padrão de entrada. Hopfield, para demonstrar este fato, utilizou o exemplo de uma função matemática que decresce a cada vez que a rede troca de estado, conseqüentemente, chegará um momento em que esta função alcançará um valor mínimo e não será mais alterada. Este é o momento em que a rede alcançou a estabilidade.

A rede de Hopfield pode ser comparada a um modelo físico, onde a cada troca de estado da rede a energia da mesma diminui, portanto, a fase de aprendizado chegará ao fim no momento em que a rede tiver a sua energia minimizada.

O perceptron teve sua validade provada no **modelo backpropagation**, que possibilitou a implementação da terceira camada necessária para o aprendizado do XOR. Utilizando uma rede de neurônios como os utilizados no perceptron, o backpropagation realiza uma retro-propagação do erro da saída para as camadas anteriores. O erro é o resultado da comparação entre a saída desejada (pré-definida) e a saída real da rede. Com esta retro-propagação, juntamente com uma função limiar de valores fracionários (fugindo ao *tudo ou nada*), possibilita-se a representação de funções não-lineares, permitindo o treinamento da função

XOR. Por este fato, o backpropagation tornou-se uma das redes mais utilizadas, dado seu amplo espectro de representações possíveis.

Vários outros modelos foram criados desde então, embora estes citados anteriormente foram os mais destacados por terem sido os precursores e por terem definidos os paradigmas de modelos de redes neurais existentes hoje.

1.2 Características Básicas de Redes Neurais Artificiais

Quando falamos em Redes Neurais Artificiais (RNA's), podemos distinguir pelo menos dois componentes físicos: **conexões e elementos de processamento**. A combinação desses dois elementos cria uma RNA. Podemos fazer uma analogia com um grafo, onde os nodos são os elementos de processamento e as arestas são as conexões.

Há ainda outros componentes (não-físicos) das redes neurais: **padrões e funções** [SIM 90]. Padrões são dados de entrada da rede. Eles são uma codificação que corresponde a certa modelagem da realidade de uma aplicação definida para a RNA. Funções são modelos matemáticos utilizados no treinamento e reconhecimento de padrões.

Conexões, elementos de processamento, padrões e funções são os componentes básicos de qualquer RNA, mas infelizmente esta terminologia não é uma regra, uma vez que não há uma norma que defina as denominações e utilização de cada um deles. Mas, para que o leitor tenha uma visão de como funcionam estes componentes, cada um deles é definido a seguir.

1.2.1 Padrões

As RNA's não podem operar sem dados. Esses dados são os padrões apresentados a uma RNA. Podem ser valores numéricos ou mesmo caracteres que serão posteriormente transformados em números. A um determinado padrão de entrada corresponde um sinal de saída, sendo que a dimensão (tipo) do padrão de entrada pode ser diferente do padrão de saída.

Uma das principais aplicações de RNA's é o reconhecimento (classificação) de padrões. Por exemplo, em reconhecimento da fala existem tipos diferentes de características envolvidas. Nesta aplicação, o padrão de entrada pode ser uma matriz que contém o caracter, e a saída apenas um número que indica se o caracter foi reconhecido ou não. Assim, temos um padrão de entrada com dimensão diferente do padrão de saída. Além disso, neste caso torna-se essencial um *pré-processamento* de informações. Isso significa que a seleção e representação correta das características do padrão a ser aplicado pode afetar a performance da rede. Criar o melhor conjunto possível de características como padrão é o primeiro passo para o sucesso de qualquer aplicação em RNA's.

1.2.2 Funções

Das funções utilizadas em RNA's, podemos distinguir basicamente em dois tipos: funções para transferência de sinais entre neurônios; e funções para aprendizado de padrões. As funções de transferência, também chamadas *funções de limiar*, são aquelas responsáveis por determinar a forma e a intesidade de alteração dos valores transmitidos de um neurônio a outro.

As funções de limiar são basicamente quatro: a *linear*, a *hard-limiter* ou *step*, a *em rampa*, a *sigmoid*, e a *gaussiana*.

A função *linear* é uma equação linear da forma:

$$f(x) = \alpha x$$

sendo que x é um número real e α um escalar positivo (inclinação da reta). Veja figura 1.2.

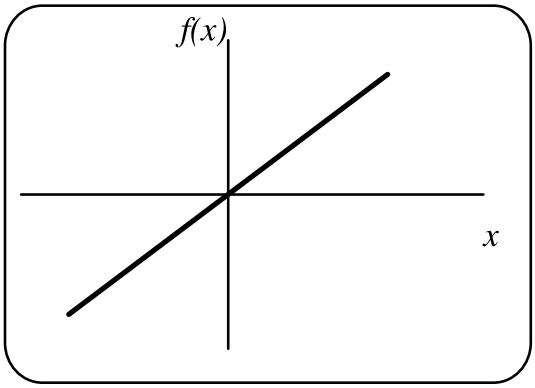

Figura 1.2: **Linear:** reta resultante da função linear de transferência.

A *hard-limiter* é uma equação que pode receber dois valores, uma vez que é utilizada para valores binários. Tem a forma:

$$f(x) = \begin{cases} \beta & \text{se } x \geq \theta \\ -\delta & \text{se } x < \theta \end{cases}$$

onde β e δ são os valores que valerão para f(x) caso x ultrapasse ou não o limiar θ. Veja figura 1.3.

A função de limiar *em rampa* tem este nome por ser uma *hard-limiter* modificada: ela possui não uma transição direta entre dois valores, mas sim uma **fase** de transferência:

$$f(x) = \begin{cases} \gamma & \text{se } x \geq \gamma \\ x & \text{se } |x| < \gamma \\ -\gamma & \text{se } x \leq -\gamma \end{cases}$$

onde γ é o valor de saturação da função, ou seja, durante a transição o valor de f(x) irá variar dentro do intervalo $(\gamma, -\gamma)$. Veja a figura 1.4.

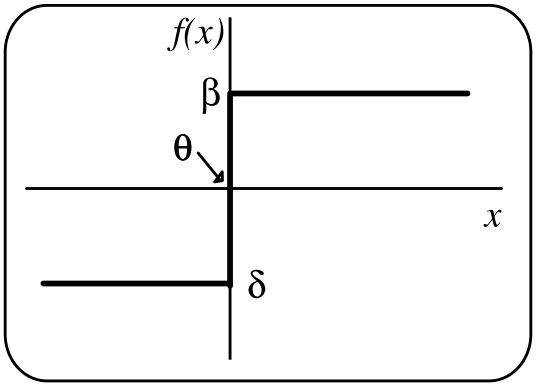

Figura 1.3: **Hard Limiter:** o coeficiente de limiar determina onde será o limite de transferência.

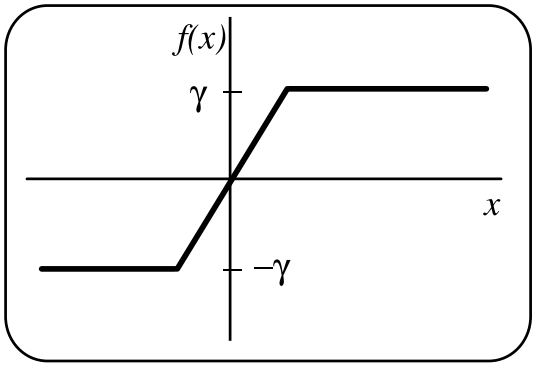

Figura 1.4: **Em rampa:** esta função permite a delimitação de uma área de transição durante a variação da transferência.

A função *sigmoid* é a versão contínua da *em rampa*. Ela permite uma transição gradual e não linear entre dois estados:

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

onde α é um real positivo. Quanto maior o valor de α, mais detalhada será a transição de um estado a outro. Este tipo de função é utilizada também em outras áreas como sociologia e química. Veja figura 1.5.

A função de transferência *gaussiana* é conhecida pelo seu uso em estatística. Aqui ela é usada quando há a definição de um ponto médio em x e uma variância a este ponto:

$$f(x) = e^{(-x^2/v)}$$

onde v é uma variância pré-definida. Veja figura 1.6.

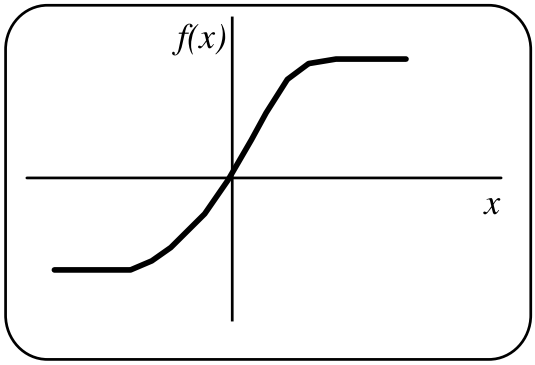

Figura 1.5: **Sigmoid:** uma transição mais detalhada.

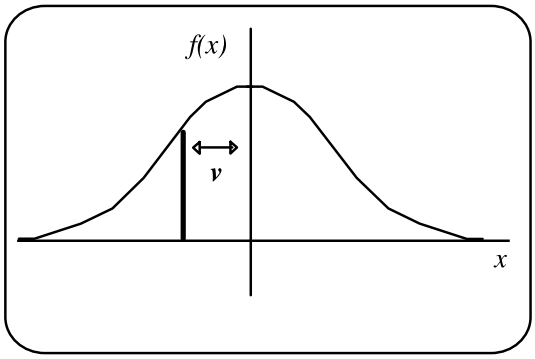

Figura 1.6: **Gaussiana:** distribuição uniforme baseada na variância.

Vamos passar a analisar agora outro tipo de função: a de aprendizado. As funções de aprendizado em geral dependem do modelo de RNA que é escolhido. Estas funções servem para realizar a alteração dos valores dos pesos da rede, possibilitando assim o aprendizado de um determinado padrão.

As diversas funções de aprendizado estão descritas nos respectivos modelos, que serão analisados posteriormente.

1.2.3 Conexões

Uma RNA é equivalente a um grafo orientado (dígrafo). Um dígrafo tem arestas (conexões) entre nodos (elementos de processamento) com um só sentido. Isso faz com que a informação flua numa direção definida (denotada por uma flexa no desenho do dígrafo). Esta informação flui através das arestas e é coletada pelos nodos. RNA's estendem a representação de dígrafo por incluir um peso em cada aresta (conexão) que modula o sinal de saída que é passado para o nodo adjacente (ver figura 1.7).

Os pesos são os responsáveis pela memorização do padrão, pois são ajus-

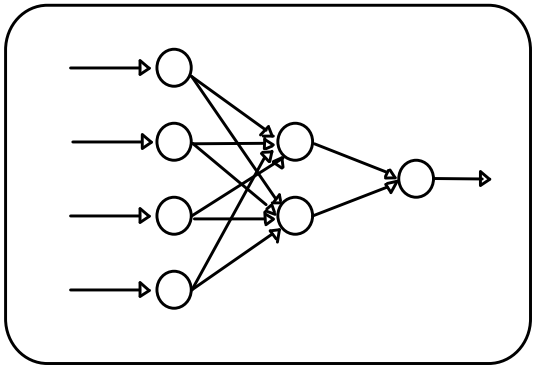

Figura 1.7: **Diagrama:** dígrafo de uma RNA.

tados durante o processo de aprendizado que captura as informações. Desta forma, uma conexão define tanto um fluxo de informação quanto a modulação da informação que é passada.

Desta forma, pesos e conexões positivos (possuem um valor numérico positivo) são excitatórios e aqueles que são negativos são inibitórios. Caso tenhamos um peso igual a zero, é como se a conexão não existisse.

1.2.4 Elementos de Processamento

O elemento de processamento (também chamado de *neurônio* ou *neuronodo*) é a parte da RNA onde é realizada todo o processamento. Um elemento de processamento (que é um nodo no dígrafo) de uma camada de entrada recebe apenas um valor do padrão de entrada correspondente, mas possui diversas conexões com os neurônios das camadas seguintes (que podem ter várias entradas).

Cada elemento de processamento coleta a informação que foi mandada para ele e produz um único valor de saída. Existem duas importantes qualidades que um elemento de processamento deve ter:

1. Elementos de processamento necessitam apenas de informações locais. A saída do elemento de processamento é uma função dos pesos e das entradas;

2. elementos de processamento produzem apenas um valor de saída. Este valor único é propagado através das conexões do elemento emissor para o receptor, ou para fora da rede, quando for um elemento da camada de saída.

Estas duas qualidades permitem que RNA's operem em paralelo.

Existem vários mecanismos para se obter a saída de um elemento de processamento. Geralmente temos uma função das saídas da camada anterior e os pesos das conexões entre a camada anterior e a atual. Matematicamente, temos que a saída de um elemento de processamento é uma função das entradas e dos

pesos:

$$Entrada_i = F(Padrão, Peso_j)$$

O modo mais comum de função de um elemento de processamento é a combinação linear. Desta forma temos:

$$Entrada_i = f(\sum(padrão_{ij} peso_{ij})) = f(PadrãoPeso_j)$$

com i variando de 1 a n (número de conexões).

1.3 Aplicações

São inúmeras as aplicações possíveis para as redes neurais, mas elas vieram de encontros às necessidades de modelagem de aplicações que não representáveis através da programação convencional, como, por exemplo, o reconhecimento de imagens. Neste sentido, atualmente encontramos outros modelos matemáticos como a lógica *fuzzy* e os algoritmos genéticos, que também contribuem para aplicações que exigem uma certa margem de erro para serem executadas. Neste sentido, a seguir são apresentados alguns exemplos de aplicações típicas das redes neurais, que dificilmente seriam viáveis na programação convencional.

1.3.1 Reconhecimento de Fala

Máquina de Escrever Fonética

Utiliza o modelo SOM (Self-Organizing Map, de Kohonen) para aprender fonemas, que posteriormente serão transformados em palavras, através de regras gramaticais aprendidas automaticamente [KOH 90]. O aprendizado é feito com amostras do espectro de curta duração (da gravação), que são apresentados à rede na ordem natural da fala, o que contribui para a auto-organização na rede SOM.

Para a adaptação desta rede a um novo locutor, basta o ditado de 200 a 300 palavras para que seja reorganizado o aprendizado. Nesta fase, para uma melhor fidelidade do aprendizado, podem ser usados modelos supervisionados como LVQ1, LVQ2 e LVQ3.

A transformação dos fonemas em palavras dá-se através de uma gramática denominada *Contexto Dinamicamente Expansível*, onde podem ser usadas regras ou produções para a correção de erros de transformação.

1.3.2 Identificação de Sinais de Radar

Utilizando os modelos perceptron e backpropagation, Sigillito & Hutton realizaram a classificação de sinais de radar [SIG 90]. O radar transmite múltiplos padrões de sinais para a ionosfera e aguarda o retorno. A função da rede neural aqui é classificar os sinais de retorno em bons ou maus. Os testes apontaram o reconhecimento dos sinais de retorno ao nível de um especialista humano.

1.3.3 Mercado Financeiro

A partir do modelo backpropagation, Zaremba construiu uma rede para análise do mercado financeiro [ZAR 90]. Com base em valores mensais de ativos, são definidos padrões construídos a partir de *janelas* para cada 4 meses, com cinco valores por mês. Para cada ativo, há um aprendizado e, portanto, um mapeamento de pesos específico. A função da rede neural nesta aplicação é aprender as flutuações do mercado em sua história para posterior reconhecimento de tendências do mercado futuro. O êxito deste tipo de aplica ção depende enormemente dos dados ensinados, ou, em outras palavras, do modelo adotado para o pré-processamento destes dados.

1.3.4 Composição Musical

Eberhart & Dobbins utilizaram uma variante do modelo backpropagation para realizar o reconhecimento de notas musicais, permitindo a composição automática [EBEa 90]. As notas são diferenciadas por sua freqüência e duração, permitindo sua codificação e treinamento da rede neural com estes padrões. A representação das notas pode-se dar por valoração direta, onde cada nota possui um valor, ou por transição, onde se ensina para a rede uma transição de notas, exigindo assim um conjunto harmonioso de entrada para um reconhecimento preciso. Para esta última forma, cada uma das saídas da rede é um conjunto de notas que farão a composição final.

Capítulo 2

Modelos

2.1 Classificação

Apesar de não haver uma norma taxonômica para os modelos de redes neurais, há diferenças claras entre eles [LIP 87] [KOH 90]. Assim, podemos realizar uma distinção entre os modelos através de suas características básicas, como o tipo de entradas, a forma de conexão e o tipo de aprendizado.

Por tipos de entrada entendemos a entrada ser binária ou intervalar:

- Chamamos de **binários** aqueles modelos que aceitam entradas discretas, ou seja, somente na forma de 0 e 1. Neste caso, encontram-se modelos como o de Hopield e ART;
- os **intervalares** são os que aceitam qualquer valor numérico como entrada (forma contínua). Estes são modelos como o Backpropagation e Kohonen.

Por forma de conexão definimos a maneira como os elementos da rede estão conectados e, por conseguinte, como o modelo matemático representa a transmissão dos sinais na rede. Há três formas de conexão:

1. **alimentação à frente**, onde os sinais de entrada são simplesmente transformados em sinais de saída;
2. **retro-alimentação**, no qual os sinais ficam sendo alterados em diversas transições de estado, sendo a saída também alimentadora da entrada;
3. **competitiva**, que realiza a interação lateral dos sinais recebidos na entrada entre os elementos dentro de uma zona de vizinhança.

O tipo de aprendizado (treinamento) refere-se à existência ou não de um sinal de saída pré-definido para a rede.

No *aprendizado supervisionado*, há uma definição sobre qual a saída que se deseja para a rede, o que leva a forçar o ajuste dos pesos de modo a representar o sinal desejado.

Por outro lado, há o *auto-aprendizado* (não-supervisionado), que limita-se a fazer uma representação da distribuição de probabilidade dos padrões de entrada na rede. Este tipo de treinamento está intimamente ligado com a conexão competitiva.

Tendo uma visão taxonômica dos modelos, melhor nos situaremos nos modelos que veremos a seguir. As classificações que foram citadas anteriormente são complementares, como mostra a figura 2.1.

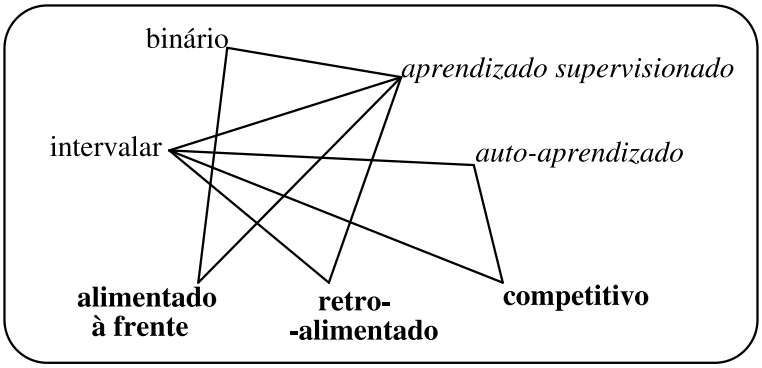

Figura 2.1: **Relações:** classificações para modelos de redes neurais.

Como podemos perceber, modelos com entrada binária muitas vezes podem ser compostos ainda com características como o aprendizado supervisionado e a conexão alimentada à frente. A entrada intervalar pode ser composta também com o aprendizado supervisionado ou ainda com o auto-aprendizado.

Modelos com um aprendizado supervisionado, por sua vez, podem compor com entrada binária ou intervalar, e conexões alimentadas à frente ou retro-alimentadas. Com auto-aprendizado tem-se apenas entradas intervalares e conexões competitivas.

No caso de ser utilizada a conexão alimentada à frente, é possível também o uso de apenas entradas binárias e de aprendizado supervisionado. Conexões retro-alimentadas são comumente combinadas com entradas binárias e aprendizado supervisionado. E conexões competitivas são encontradas junto a entradas intervalares e auto-aprendizado.

Essas combinações de classificações não são definições estáticas, elas são apenas um elemento de auxílio na compreensão das características dos modelos mais comumente usados na atualidade. E são estes modelos que veremos a seguir.

2.2 Perceptron / Adaline

2.2.1 Características

- Possui entrada intervalar, aprendizado supervisionado e alimentação à frente.
- Utiliza o *Combinador Adaptativo Linear*, onde a saída de um elemento processador (ep) é a combinação linear das entradas [WID 90]. Estas são um vetor que é multiplicado por pesos pré-existentes no **ep**, assumindo a forma:

  $$S(t) = \sum_{i=0}^{n-1} p_i(t) e_i(t)$$

  onde P[n] é o vetor de pesos, E[n] o vetor de entradas, S a saída do ep e t o número da amostra (de entrada).
- Possui como regra de adaptação dos pesos, o algoritmo *perceptron* ou α-*LMS* (também conhecido como ***adaline***). A diferença entre ambos está na forma do cálculo. O perceptron tem a seguinte formulação:

  $$P(t+1) = P(t) + \alpha(\varepsilon(t)/2)E(t)$$

  onde α é o coeficiente de aprendizado e ε é o erro da saída *quantizada* do **ep**. A saída é quantizada através da função de limiar aplicada à saída. O erro tem a forma:

  $$\varepsilon(t) = d(t) - f_{lim}(S(t))$$

  ou seja, o erro é fruto da aplicação da função de limiar sobre a diferença entre a saída desejada d e a resultante S. A saída desejada é o valor -1 ou 1 que se espera como resultado do treinamento.
- O α-LMS possui como diferencial a fórmula de ajuste de pesos e do erro. A atualização de pesos é da forma:

  $$P(t+1) = P(t) + \alpha(\varepsilon(t)E(t)/|E(t)|^2)$$

  e o erro passa a ter uma forma sem quantização:

  $$\varepsilon(t) = d(t) - S(t)$$

- Ambas as regras necessitam a saída quantizada Sq.

2.2.2 Topologia

A figura 2.2 apresenta um simples **ep** segundo as fórmulas do perceptron e α-LMS discutidas anteriormente. O **ep** recebe o padrão representado no vetor **E**, que é multiplicado pelo vetor de pesos proveniente da **Regra de Adaptação**. O cálculo resultante é somado, perfazendo assim a *soma ponderada*, que é a saída

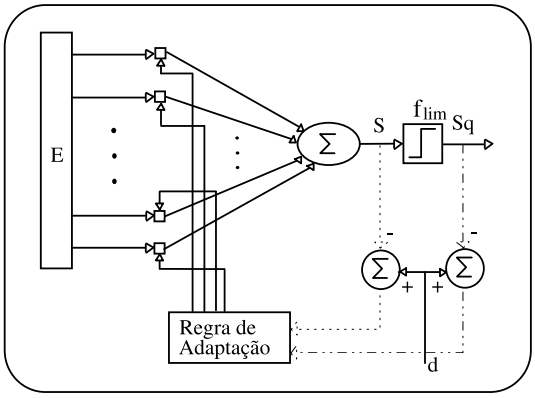

Figura 2.2: **Topologia:** combinador adaptativo linear.

não-quantizada **S**, utilizada na regra α-LMS. A saída quantizada é representada pelo **Sq**, utilizada na regra perceptron. Ambas as regras calculam os erros a partir da resposta desejada **d**.

Cada um desses elementos pode ser combinado com outros formando uma rede de duas camadas, ampliando a capacidade de armazenamento de padrões, como mostra a figura 2.3.

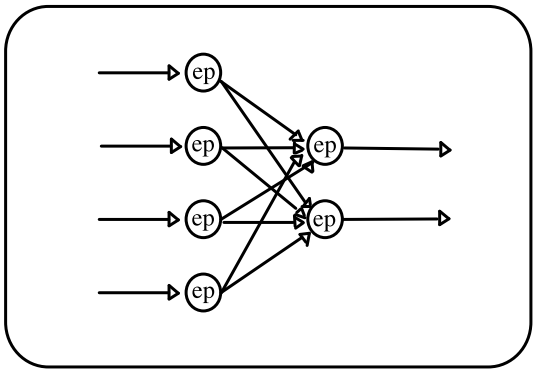

Figura 2.3: **Rede:** diversos elementos processadores conectados.

Reunindo diversos nodos numa rede, é possível a caracterização de padrões com mais valores ou em maior quantidade. A saída de um **ep** é conectada com a entrada de outro, distribuindo o poder de processamento e armazenamento dos padrões.

2.2.3 Algoritmo

1. Inicialize os pesos e o coeficiente de limiar θ com valores randômicos e dentro do intervalo (0;1]. O coeficiente de limiar é aqui utilizado para

ajuste interno da função de limiar definida. Como função limiar, neste caso, é utilizada a *hard-limiter*.

2. Apresente o padrão de entrada, na forma do vetor **E**, e a saída desejada **d** para o padrão.

3. Calcule a saída:
$$S(t) = \sum_{i=0}^{n-1} P_i(t)E_i(t) - \theta$$

4. Se for a regra perceptron, aplica-se a função limiar:
$$Sq(t) = f_{lim}(S(t))$$

5. A atualização de pesos segue as regras que já foram apresentadas:

   - para o perceptron:
$$P(t+1) = P(t) + \alpha(\varepsilon(t)/2)E(t)$$
$$\varepsilon(t) = d(t) - Sq(t)$$
   - para o α-LMS:
$$P(t+1) = P(t) + \alpha(\varepsilon(t)E(t)/|E(t)|^2)$$
$$\varepsilon(t) = d(t) - S(t)$$

6. Voltar ao passo 2 até que a saída quantizada **Sq** esteja respondendo corretamente com a saída desejada **d**.

2.2.4 Uso

Aprendizado

Para realizar o treinamento (aprendizado) de um padrão, é necessário primeiramente passá-lo para a forma de **1** e **-1**. É comum encontrarmos arquivos de padrões em binários, resultantes, por exemplo, de processamento de sinais digitais. Cada **0** do padrão binário deve ser colocado na forma de **-1** para que não hajam erros na execução das equações do algoritmo. Caso o padrão seja contínuo, este processamento torna-se desnecessário.

A forma em que são apresentados os padrões também é de grande importância. Se há mais de um, devem ser apresentados aleatoriamente, ou de forma alternada. Nunca se deve treinar um padrão para somente depois treinar outro, isso deixaria os pesos da rede sempre tendendo a apenas um dos padrões.

Um outro passo é o ajuste dos parâmetros do algoritmo, que são o coeficiente de aprendizado α e o coeficiente de limiar θ. O coeficiente α determina o intervalo de ajuste dos pesos, quanto maior seu valor, maior será a diferença dos valores de uma iteração a outra. Quanto maior o valor, maior o intervalo

de ajuste, e portanto maior será a rapidez de convergência. Por outro lado, se o valor de α for sempre alto, poderá ocorrer o caso do sistema nunca chegar numa convergência, entrando em laço infinito. Isto porque os valores nunca serão pequenos o suficiente para garantir uma saída confiável, bem próxima a 0 ou a 1, dependendo da saída desejada. O valor de α pode ser arbitrado (e, de certa forma, terá que ser inicialmente arbitrado) pelo projetista da aplicação, devendo ficar dentro do intervalo [0.01;1.0]. O coeficiente poderá ser variável, seguindo uma função decrescente, partindo de um valor maior para um menor quando próximo da convergência.

Já o coeficiente de limiar θ pode seguir um ajuste conjuntamente com os pesos ou ser definido pelo projetista. Ele é responsável pelo deslocamento, na função limiar, do ponto central de decisão da quantização da saída. Em outras palavras, um coeficiente θ valendo 0 significa que não há deslocamento em relação ao ponto onde a função limiar cruza o eixo das abscissas (veja figura 1.3).

Enquanto o eixo das abscissas indica a saída linear **S** resultante da soma ponderada, o eixo das ordenadas corresponde à saída quantizada **Sq**, e o coeficiente de limiar θ determina qual valor será o limite para a transferência de -1 para 1, ou vice-versa. Se θ for definido como sendo 0,5, os valores de saída abaixo (ou igual) a este coeficiente passarão a valer -1, caso contrário 1.

Uma vez realizado o ajuste dos parâmetros, obtém-se, após um determinado número de iterações, uma saída como a desejada.

Reconhecimento

Para efetuar o reconhecimento de um padrão, ou seja, saber se ele pertence ou não ao conjunto de padrões treinados, basta que o algoritmo seja executado uma vez, sem o ajuste de pesos. Isso basta para que obtamos uma saída quantizada que indique a que padrão treinado pertence o padrão atual.

Vantagens e Desvantagens

A grande vantagem da implementação do algoritmo perceptron/adaline é a simplicidade. São poucos parâmetros a ajustar e o padrão de entrada não necessita de um pré-processamento muito elaborado, dependendo da aplicação.

Por outro lado, ele tem sua aplicação restrita a padrões não muito complexos, que sejam linearmente separáveis (veja capítulo introdutório).

Aplicações

A utilização do perceptron/adaline está restrita à quantidade de padrões envolvidos e sua complexidade. Ele pode ser usado em reconhecimento de sinais digitais ou, quando utilizado em grandes redes, pode ser aplicado à caracteriza ção de apenas um padrão por neurônio.

2.3 Backpropagation

2.3.1 Características

- É um modelo com entrada intervalar, aprendizado supervisionado e com alimentação à frente.
- O backpropagation deriva-se do modelo perceptron/adaline [RUM 86] [WID 90]. Seus neurônios são compostos por três ou mais camadas de adalines interconectados. Estes adalines têm uma diferença fundamental: eles utilizam uma função do tipo *sigmoid* como função de limiar.
- A função de limiar é do tipo *sigmoid* uma vez que é necessária uma função não-linear para ampliar o potencial de classificação de um modelo. Essa variação foi o que possibilitou a este e outros modelos realizarem representações complexas, como o aprendizado da função lógica XOR (ver seção 1.1). A função sigmoid tem a forma:

  $$sgm(S_i) = 1/(1 + e^{-(S_i - \theta)})$$

  onde S é a saída linear resultante da soma ponderada do nodo i e θ é o coeficiente de limiar.
- Neste modelo, o erro obtido na saída é transferido para as camadas intermediárias. Daí o nome *retropropagação* (backpropagation). Isso se dá pela necessidade de ajuste dos neurônios que não têm contato com a saída, necessitando, assim, de algum parâmetro para atualização dos pesos.
- O cálculo do erro começa na última camada, ele tem a forma:

  $$\varepsilon s_i(t) = S(t)(1 - S(t))(d_i(t) - S(t))$$

  onde S é a saída linear, d a saída desejada, e i o nodo atual. A partir deste erro são ajustados os pesos da última camada:

  $$P_i(t + 1) = P_i(t) + \alpha \varepsilon s_i(t) En(t)$$

  onde P é o vetor de pesos, α é o coeficiente de aprendizado e En o vetor resultante da saída da camada anterior.
- O erro da(s) camada(s) intermediária(s) é feito a partir do erro da camada de saída:

  $$\varepsilon_i(t) = En(t)(1 - En(t)) \sum_k \varepsilon_k p_{ik}(t)$$

  onde En é o vetor resultante da saída da camada anterior até esta camada intermediária; k é o número de nodos conectados a seguir do atual; ε é o erro do nodo k; p é o peso correspondente à conexão do nodo atual com o nodo k. A partir deste erro, são calculados os pesos:

  $$P_i(t + 1) = Pi(t) + \alpha \varepsilon_i(t) En(t) + \mu(P_i(t) - P_i(t - 1))$$

  onde μ é um coeficiente de aceleração de convergência denominado *momentum*.

2.3.2 Topologia

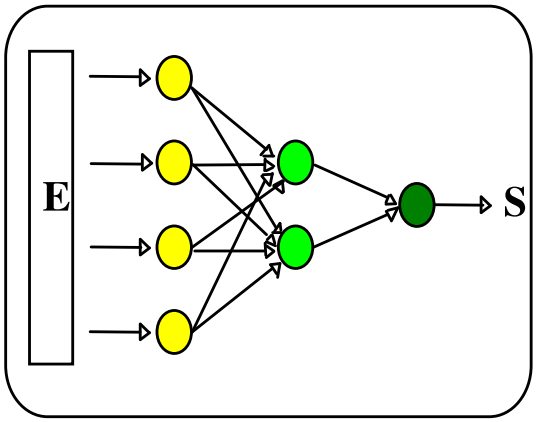

Figura 2.4: **Multicamada:** a topologia backpropagation é uma ampliação do modelo perceptron/adaline.

Como no modelo perceptron/adaline, ocorre uma camada de entrada para recepção dos valores. A novidade é a possibilidade de uma ou mais camadas intemediárias (figura 2.4).

2.3.3 Algoritmo

1. Inicializar os pesos e coeficientes de limiar com valores pequenos e randômicos.

2. Apresentar o vetor de entrada (padrão) e a saída desejada.

3. Calcule a saída:
$$S(t) = \sum_{i=0}^{n-1} P_i(t)E_i(t) - \theta$$

4. Aplique a função sigmoid vista anteriormente:
$$Sq(t) = sgm(S(t))$$

5. Atualize os pesos da última camada:
$$P_i(t+1) = P_i(t) + \alpha\varepsilon_i(t)En(t)$$
e o cálculo do erro:
$$\varepsilon s_i(t) = S(t)(1 - S(t))(d_i(t) - S(t))$$

6. Atualize os pesos da(s) camada(s) intermediária(s):

$$P_i(t+1) = P_i(t) + \alpha\varepsilon_i(t)En(t) + \mu(P_i(t) - P_i(t-1))$$

e o erro:

$$\varepsilon_i(t) = En(t)(1 - En(t)) \sum_k \varepsilon_k p_{ik}(t)$$

7. Voltar ao passo 2 até que atinja um valor próximo ao da saída desejada.

2.3.4 Uso

Aprendizado

A preparação do padrão de entrada segue os mesmos princípios do perceptron/adaline: não há muito pré-processamento, dependendo da aplicação desejada. A aplicação também define o número de neurônios na camada intermediária. Normalmente utiliza-se um neurônio para cada classe de padrão. Por exemplo, caso queira-se treinar o alfabeto, basta um neurônio por letra, mesmo que haja um número expressivo de padrões distintos (amostras) por letra. Também é comum a utilização de apenas uma camada intermediária para a grande maioria das aplicações. A seqüência de apresentação dos padrões continua a mesma do perceptron/adaline: aleatória ou alternada.

Após definidos os padrões e a topologia, são necessárias algumas iterações de treinamento para que se possa definir os parâmetros. Como no perceptron/adaline, α e θ podem ser definidos pelo projetista, assim como serem funções variáveis. O incremento deste modelo está no momentum μ. Este coeficiente permite a aceleração do processo de convergência. Seu valor fica no intervalo [0;1].

O treinamento se concretizará quando as saídas desejadas estiverem numa margem segura de proximidade à saída da função sigmoid. Esta aproximação existe devido à dificuldade de se chegar ao valor exato da saída deseja e, então, estima-se uma margem de erro para que se alcance a convergência.

Para a maioria das aplicações são necessárias muitas iterações até a convergência. Isso também é dependente de um bom ajuste dos coeficientes envolvidos.

Reconhecimento

Como no perceptron/adaline, o reconhecimento é apenas uma iteração do algoritmo para verificação da saída. Dado o padrão que se quer reconhecer, realiza-se uma iteração do algoritmo, excetuando-se obviamente o ajuste de pesos, e comparando a saída do sigmoid com as saídas desejadas aprendidas. Do resultado desta comparação sabe-se se a qual classe o padrão pertence.

Vantagens e Desvantagens

A grande vantagem deste modelo é sua capacidade de abranger as mais variadas classificações de padrões, podendo ser utilizado numa enorme gama de aplicações (veja seção seguinte).

Por outro lado, as disvantagens também são muitas. Em geral, o tempo de treinamento é extremamente longo para a maioria das aplicações de uso prático, como o reconhecimento de caracteres. Há também certos tipos de padrões, como os envolvidos em séries temporais, que demandam uma grande necessidade de pré-processamento, dada a complexidade da análise e classificação dos padrões em questão.

Aplicações

O backpropagation é utilizado para reconhecimento de caracteres, composição musical, séries temporais, reconhecimento de sinais em geral, treinamento de robôs, etc. Alguns destes exemplos estão citados no capítulo introdutório.

2.4 Hopfield

2.4.1 Características

- É caracterizada por ser do tipo binária, supervisionada e com retro-alimentação.
- É baseada no conceito de energia da rede, onde esta é proporcional à troca de estados da rede. Isto é, quanto maior for a oscilação entre um estado e outro na saída, maior será a energia. Portanto, a fase de reconhecimento tem por objetivo minimizar a quantidade de energia de maneira que a rede convirja para um único estado na saída [WAS 89].

2.4.2 Topologia

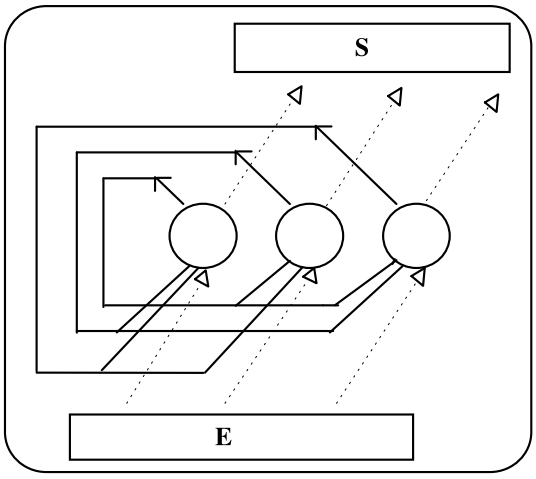

Figura 2.5: **Retro-alimentação:** topologia de Hopfield, com três nodos.

Como podemos ver pela figura 2.5, há uma retro-alimentação que vai da saída de cada neurônio para a entrada de todos os demais, **exceto a sua própria entrada**.

2.4.3 Algoritmo

1. Inicializar os pesos das conexões:

$$w_{ij} = \begin{cases} \sum_{s=0}^{M-1} x_i^s x_j^s & \text{se } i \neq j \\ 0 & \text{se } i = j \end{cases}$$

   onde w_{ij} é o peso da conexão entre o nodo i e o nodo j, e x_i^s é o elemento de entrada i do padrão s. Existem M padrões, de 0 a M-1. Os limiares das unidades é zero.

2. Inicializar a rede com um padrão desconhecido:

$$\mu_i(0) = x_i, \quad 0 <= i <= N-1$$

   onde $\mu_i(t)$ é a saída do nodo i no tempo t.

3. Iteragir até convergir:

$$\mu_i(t+1) = f_h[\sum_{i=0}^{N-1} w_{ij}\mu_j(t)]$$

   A função f_h é a função de limiar hard-limiter.

4. Volte ao passo 2 e repita a iteração até que as saídas dos nodos permaneça inalteradas.

2.4.4 Uso

Aprendizado

Pelo fato da rede de Hopfield ser do tipo binária, a primeira atitude a ser tomada antes do aprendizado e também da fase de reconhecimento da rede, é converter os valores binários (0,1) em bipolares (-1,1), para que o valor 0 não cause problemas quanto o cálculo das saídas.

Após, os padrões exemplares, isto é, aqueles com que a rede será treinada e para os quais a rede deverá convergir, são inseridos na entrada aleatoriamente. A fase de treinamento irá se encerrar a partir do instante que, não importando o padrão exemplar posto na entrada, a rede devolver o mesmo padrão na saída ao longo de sucessivas iterações. Portanto, ao contrário de outras redes como Backpropagation e Kohonen, onde pode-se determinar uma condição de parada na fase de aprendizado, no modelo de Hopfield isto não acontece, pois neste o treinamento só é encerrado após a rede convergir para um único estado na saída.

Reconhecimento

Na etapa de reconhecimento, padrões diversos serão inseridos na rede e para cada um deles a rede tentará convergir para o padrão exemplar que melhor se aproxima deste. O reconhecimento poderá não convergir em virtude das limitações do modelo explicadas na seção seguinte.

Vantagens e Desvantagens

A rede de Hopfield têm como vantagem o fato de ser adequada quando se deseja utilizar uma memória associativa, isto é, uma memória que armazena diversos padrões ao mesmo tempo e que cada um deles pode ser referenciado dependendo com qual deles melhor se assemelha o padrão de entrada .

Este modelo tem duas desvantagens, ou em outras palavras, duas restrições a fim de que possibilite uma boa capacidade de reconhecimento sem redundâncias. Uma delas é que o número de padrões a ser ensinado, ou número de padrões exemplares, deverá ser no máximo 15% do total de nodos da rede. E um outro ponto com o qual deve-se ter cuidado é não permitir que um padrão exemplar compartilhe muitos bits com um outro padrão, pois isto pode *confundir* a rede.

Aplicações

Uma das muitas aplicações do modelo de Hopfield que pode ser citado é um conversor Analógico-Digital. Neste, os amplificadores funcionam como se fossem os neurônios da rede e os resistores como os pesos. Todas as saídas da rede são trocadas no início de intervalos discretos chamados de época. No início de cada época, a soma das entradas de cada neurônio é somado, e se ultrapassar o limiar previamente definido, a saída será 1, caso contrário, será 0. O objetivo é selecionar os resistores a fim de que um aumento contínuo de uma tensão X aplicada a uma entrada, produza um conjunto de quatro saídas, cujo valor em binário corresponderia ao nível da tensão de entrada.

2.5 Kohonen

2.5.1 Características

- Possui auto-aprendizado, entrada intervalar e conexão competitiva.
- Não há propriamente um reconhecimento de padrão como em outros modelos, mas há a classificação de um padrão junto com outros que têm características semelhantes, formando classes. Estas classes são organizadas num mapa, onde pode-se observar a distribuição dos padrões. Desta maneira, no instante em que um padrão é inserido na rede, esta o coloca na classe onde melhor o padrão se adequa, em função das suas características.
- Um outro aspecto importante é que o modelo de Kohonen é chamado de biologicamente plausível. No córtex auditivo, por exemplo, existem con-

juntos de células que só reagem a determinados impulsos ou freqüências, enquanto a outros não [EBE 90]. No modelo ocorre o mesmo, onde um padrão ao ser reconhecido faz com que um ou somente alguns neurônios de saída sejam ativados (aqueles que mais se assemelham ao padrão inserido) enquanto outros não.

- Este tipo de rede é usado quando se deseja, por exemplo, reconhecer diversos padrões que possuam alguma relação entre si, como reconhecimento de voz, que será explicado posteriormente na seção de aplicações.

2.5.2 Topologia

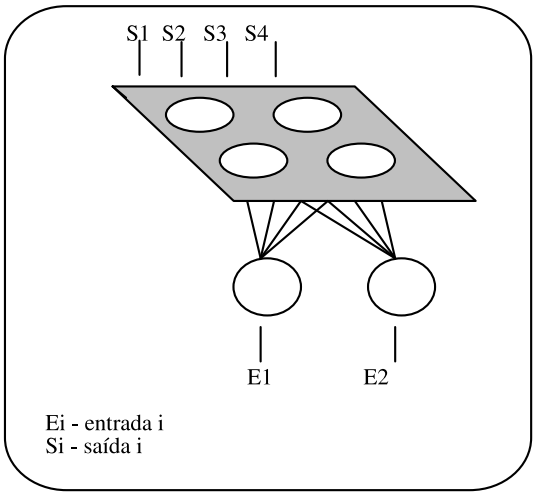

Figura 2.6: **Topologia:** rede Kohonen de duas entradas e quatro saídas.

A rede Kohonen (veja figura 2.6) é composta por um número de entradas correspondente ao tamanho dos padrões e um conjunto de neurônios de saída, sendo que cada padrão a ser reconhecido deverá ter, no mínimo, um neurônio de saída correspondente. Isto é, se for necessário a distinção entre 10 padrões, a quantidade mínima na camada de saída são 10 neurônios.

2.5.3 Algoritmo

1. Inicialização dos pesos da rede com valores baixos (0.01 a 0.1) escolhidos aleatoriamente. Ajuste inicial do raio de vizinhança, que poderá começar com a metade do diâmetro da rede e ir diminuindo linearmente.
2. Inserção do padrão de entrada.
3. Cálculo das distâncias de cada saída:

$$d_j = \sqrt{\Sigma_{i=0}^{N-1}(x_i(t) - w_{ij}(t))^2}$$

onde:

- d_j - distância entre a saída do nodo j com a entrada
- N - número de entradas
- $x_i(t)$ - entrada x_i no tempo t
- $w_{ij}(t)$ - peso da conexão do neurônio de entrada i para o neurônio j no tempo t

4. Seleção da menor distância
5. Atualização dos pesos do neurônio com a menor distância (neurônio vencedor) e seus vizinhos, definidos pelo raio de vizinhança. Isto é feito segundo a fórmula:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t))$$

   - $w_{ij}(t+1)$ - peso da conexão do neurônio de entrada i para o neurônio j no tempo t+1
   - $\eta(t)$ - coeficiente de aprendizado no tempo t

   O coeficiente $\eta(t)$ poderá variar desta maneira:

$$\eta(t) = 0.9(1 - t/1000)$$

   Os neurônios que não pertencem à vizinhança do vencedor *não devem ter seus pesos atualizados.*

6. Repetir a partir do passo 2.

2.5.4 Uso

Aprendizado

Este modelo é extremamente dependente da forma com a qual os padrões de entrada estão definidos. Isso porque o modelo apenas fará uma distribuição dos padrões na rede. Se estes padrões não possuem uma codificação numérica uniformemente distribuída, eles poderão entrar em conflito, não representando o que se deseja. Para tanto, são necessários muitos ajustes iniciais até encontrar-se um pré- processamento adequado à aplicação pretendida.

Após um padrão ser inserido na rede, é calculada a distância euclidiana, respectiva a cada neurônio de saída, e, após, verifica-se aquele que alcançou o maior valor. Este neurônio levará o nome de *neurônio vencedor*, e terá o peso da sua conexão alterada. Além deste neurônio, outros neurônios de saída poderão também ter seus pesos alterados. Isto porque antes de ser realizado o treinamento, ajusta-se uma variável chamada de raio de vizinhança σ (que poderá ser alterado durante o treinamento). O raio σ indica a qual distância mínima do neurônio vencedor um determinado neurônio deverá estar para ter seu peso alterado. Este cálculo é feito através da comparação da diferença entre a distância do neurônio vencedor e cada um dos demais, podendo esta distância ser vista como um mapa, como uma separação *física* entre os neurônios.

Reconhecimento

Depois de um determinado número de iterações, os pesos da rede estarão ajustados de maneira a poder classificar adequadamente os padrões a serem reconhecidos. Assim, ao ser inserido um padrão, um determinado neurônio de saída se ativará, indicando aquele que tiver a menor distância em relação a sua entrada. Desta forma, a rede não irá exibir um resultado exato de reconhecimento, mas sim irá classificar o padrão conforme as suas características.

Vantagens e Desvantagens

O modelo de Kohonen pode ser usado nas diversas aplicações onde comumente se utilizam outros modelos de redes neurais, como reconhecimento de padrões, robótica, processamento de sinais. Porém, ele se torna eficiente [KOH 90] naquelas aplicações em que os padrões possuam alguma relação entre si, podendo desta forma, serem classificados. Um exemplo que pode ser citado é o de reconhecimento da fala, citado no item referente a aplicações.

A rede possui como desvantagens o fato de ser um pouco complexa em relação aos outros modelos, pois as variáveis como raio de vizinhança e coeficiente de aprendizado devem ser ajustados adequadamente para que o aprendizado seja realizado com sucesso. Além disso, o número médio de iterações necessários para o treinamento é de 500 vezes o número de nodos de saída [KOH 90].

Aplicações

Como exemplo de aplicação pode ser citado a máquina de escrever fonética, ou *Phonetic Typewriter* . O objetivo desta rede é, ao ser pronunciada uma palavra, no caso no idioma finlandês, a rede irá informar os prováveis fonemas constituintes daquela palavra. Para tanto, a rede é treinada com os fonemas do idioma, onde estes, dependendo de suas características, são agrupados com outros que possuem características similares. Assim, a cada fonema pronunciado será ativado um determinado neurônio de saída correspondente.

2.6 ART

2.6.1 Características

- A Teoria da Ressonância Adaptativa (Adaptive Resonance Theory) resultou num classificador projetado por Carpenter e Grossberg que possui entrada binária, conexão retro-alimentada e aprendizado não-supervisionado [LIP 87].

- A idéia de armazenamento dos padrões tem semelhanças com o modelo de Kohonen, uma vez que há a distinção de zonas, a comparação entre as distâncias vetoriais das somas ponderadas, e a escolha de um neurônio vencedor.

- Como ele é retro-alimentado, o algoritmo tem variantes dos demais apresentados, possuindo um neurônio vencedor variável, dependente do padrão de entrada.

2.6.2 Topologia

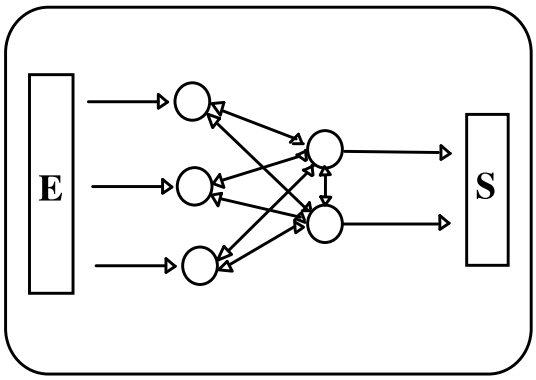

Figura 2.7: **ART:** uma topologia com conexões retro-alimentadas.

Na figura 2.7, vê-se que há a retro-propagação dos sinais, o que permite ao ART a seleção dos neurônios que melhor representariam um dado padrão de entrada. A interconexão entre os dois nodos de saída permite a inibição lateral entre eles para a definição do valor máximo de saída.

2.6.3 Algoritmo

1. Inicializar contadores:

   - pesos de conexões da entrada para a saída são inicializadas na forma:

     $$f_{ij}(0) = 1$$

   - pesos de conexões da saída para a entrada recebem:

     $$r_{ij}(0) = 1/(1+N)$$

     onde i é o nodo de entrada e $0 \leq i \leq N-1$, j é o nodo de saída e $0 \leq j \leq M-1$, N é o número de nodos de entrada e M o número de nodos de saída. Ainda deve-se dar um valor ao coeficiente de limiar θ, que serve para verificar se a soma ponderada da entrada está próxima aos pesos estabelecidos até o momento. Este coeficiente tem um valor no intervalo [0;1].

2. Apresentar padrão de entrada.

3. Calcular somas ponderadas dos nodos de entrada:

$$\mu_j = \sum_{i=0}^{N-1} r_{ij}(t)e_i$$

onde t é o laço da iteração do algoritmo e e é o componente do vetor de entrada.

4. Selecionar o neurônio vencedor através da comparação das somas ponderadas, ou seja, o que tem maior valor será o vencedor.

5. Realizar o teste de limiar:

se a divisão entre a distância euclidiana entre os pesos **F** da entrada para a saída do neurônio vencedor e a entrada **E** e o módulo vetorial da entrada for maior que o limiar:

$$\|F \cdot E\|/\|E\| > \theta$$

então continue, caso contrário, volte ao passo 3.

A distância entre F e E é calculada na forma:

$$\|F \cdot E\| = \sum_{i=0}^{N-1} f_{iv}(t)e_i$$

onde v é o neurônio vencedor, e o módulo de E:

$$\|E\| = \sum_{i=0}^{N-1} e_i$$

6. Ajustar os pesos do neurônio vencedor v:

primeiro os pesos da entrada para a saída:

$$f_{iv}(t+1) = f_{iv}(t)e_i$$

e depois da saída para a entrada:

$$r_{iv}(t+1) = f_{iv}(t+1)/(0.5 + \|F \cdot E\|)$$

7. Volte ao passo 2.

2.6.4 Uso

Aprendizado

O aprendizado do modelo ART é bem distinto dos modelos vistos anteriormente, apenas com algumas semelhanças com o de Hopfield e o de Kohonen. Assemelha-se ao de Kohonen em sua forma de distribuição dos pesos na rede através da seleção de um neurônio vencedor, com a diferença de que no ART

apenas o vencedor é atualizado, não havendo a atualização da vizinhança, que caracteriza o aprendizado competitivo. Quanto à semelhança ao Hopfield, nota-se que o padrão a aprender confunde-se com aquele a reconhecer, uma vez que há iterações para ambos. Apesar disso, há uma grande diferença entre ambos: Hopfield possui um aprendizado supervisionado, ou seja, os padrões devem ser apresentados anteriormente, e ART tem um aprendizado não-supervisionado, podendo assim ter um reconhecimento intercalado com o aprendizado, havendo a combinação de padrões novos com já aprendidos.

O aprendizado ART não necessita a tranformação da entrada binária (0 e 1) em bivalente (-1 e 1), como em outros modelos. Há somente um parâmetro a definir, o coeficiente de limiar θ, também chamado de *limiar de vigilância*. Como nos outros modelos, este coeficiente determina se um padrão está ou não treinado ou, neste caso, adaptado aos padrões atuais.

Reconhecimento

O reconhecimento confunde-se com o aprendizado, uma vez que este é um modelo não-supervisionado que incorpora padrões desconhecidos e adapta padrões semelhantes.

Vantagens e Desvantagens

O modelo ART possui a facilidade de não ter necessidade de um aprendizado prévio, podendo adaptar-se de acordo com a necessidade. Por outro lado, este modelo é muito sensível a padrões distorcidos, não podendo fazer uma boa diferenciação entre padrões semelhantes, limitando, assim, em muito suas aplicações.

Aplicações

Utiliza-se o ART para reconhecimento de imagens, mais comumente para reconhecimento de caracteres.

Referências Bibliográficas

[BEA 90] BEALE, R. & JACKSON, T. Neural Computing: An Introduction. Adam Hilger, Bristol, 1990.

[EBE 90] EBERHART, R. & DOBBINS, R. Neural Networks PC Tools - A Practical Guide. Academic Press, San Diego, 1990.

[EBEa 90] EBERHART, R. & DOBBINS, R. Case Study V: Making Music. In: Neural Networks PC Tools - A Practical Guide. Eberhart, R. & Dobbins, R. Academic Press, San Diego, 1990.

[KOH 90] KOHONEN, T. The Self-Organizing Map. Proceedings of the IEEE, V.78, n. 9, Sep. 1990.

[LIP 87] LIPPMAN, R. P. An Introduction to Computing With Neural Nets. IEEE ASSP Magazine, v. 3, n. 4, apr. 1987.

[RUM 86] RUMELHART, D. E. & MCCLELLAND, J. L. Learning Internal Representations By Error Propagation. In: Parallel Distributed Processing. Rumelhart, D. E. and McClelland, J. L. MIT Press, Cambridge, 1986.

[SIG 90] SIGILLITO, V. & HUTTON, L. Case Study II: Radar Signal Processing. In: Neural Networks PC Tools - A Practical Guide. Eberhart, R. & Dobbins, R. Academic Press, San Diego, 1990.

[SIM 90] SIMPSON, P. K. Artificial Neural Systems: Foundations, Paradigms, applications, and implementations.Pergamon Press, 1990.

[WAS 89] WASSERMAN, P. Neural Computing Theory and Practice. Van Nostrand Rheinhold, New York, 1989.

[WID 90] WIDROW, B. & LEHR, M. A. 30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation. Proceedings of the IEEE, v. 78, n. 9, sep. 1990.

[ZAR 90] ZAREMBA, T. Case Study III: Technology in Search of a Buck. In: Neural Networks PC Tools - A Practical Guide. Eberhart, R. & Dobbins, R. Academic Press, San Diego, 1990.