Redes Neurais Artificiais

Thomas Walter Rauber
Departamento de Informática
Universidade Federal do Espírito Santo
Av. F. Ferrari, 29065-900 Vitória - ES, BRASIL
Tel.: (+55)(27) 3352654 — Fax: (+55)(27) 3352850
E-mail: thomas@inf.ufes.br — WWW-homepage: http://www.inf.ufes.br/~thomas

Resumo: *Este documento apresenta uma introdução para a área de redes neurais artificiais (RNA). Em primeiro lugar motiva-se o paradigma da neurocomputação pelas capacidades cognitivas de redes neurais biológicas, inspirado pelo conhecimento da neurociência. Os fundamentos das RNA são o modelo de um neurônio, a topologia da rede e os paradigmas de aprendizagem. O modelo de McCulloch e Pitts serve como modelo básico de um neurônio artificial. Propagação para frente e redes com realimentação constituem as principais topologias de redes. Paradigmas de aprendizagem apresentados são a aprendizagem supervisionada e aprendizagem não-supervisionada. Em relação às regras para a adaptação dos pesos distingue-se entre a regra de Hebb, a regra de delta e a aprendizagem competitiva.*

A classe de redes de propagação para frente é representada pelo perceptron, o ADALINE e o perceptron com uma camada escondida. Nas regras de aprendizagem do ADALINE apresentam-se duas técnicas principais para adaptar os pesos, a solução determinística linear pela pseudoinversa e a descida de gradiente. O algoritmo de retropropagação do erro é a ferramenta fundamental para treinar o perceptron multi-camada.

Redes com realimentação que apresentam um comportamento dinâmico são representadas pelo modelo de Hopfield. Motiva-se a topologia e funcionalidade da rede e analisa-se a estabilidade dos pesos. A introdução de uma função de energia por Hopfield junto com pesos simétricos garante a estabilidade da rede. Um exemplo de aplicação desse tipo de rede é o armazenamento e recuperação de imagens binárias.

Redes competitivas concluem a apresentação. Dentro de uma camada competitiva existe um vencedor que mostra a maior coincidência com o sinal de entrada. O mapa de preservação topológica de Kohonen adicionalmente ao uso da aprendizagem competitiva impõe uma ordem topológica sobre os neurônios individuais.

ÍNDICE

I. INTRODUÇÃO

Uma das áreas de pesquisa mais fascinante presentemente é a simulação de capacidades cognitivas de um ser humano. Projetam-se máquinas capazes de exibir um comportamento inteligente, como se fossem reações humanas. A inteligência do ser humano é a mais avançada dentro do universo das criaturas e o local dessa inteligência dentro do corpo humano é o cérebro. As entidades básicas são os neurônios, interconectados em redes o que permite a troca de informação entre eles, criando a inteligência biológica. Uma ambição óbvia que surge desses fatos é a tentativa de copiar a estrutura e o funcionamento do cérebro em um ambiente técnico. Isso significa que a pesquisa tenta entender o funcionamento da inteligência residente nos neurônios e mapeá-la para uma estrutura artificial, por exemplo uma combinação de hardware e software, assim transformando as redes neurais biológicas em redes neurais artificiais.

Foram definidas uma quantidade extensa de modelos de redes neurais artificiais e os métodos associados para adaptá-los às tarefas a serem resolvidas. Um aviso prévio é o fato que os modelos artificiais têm pouco em comum com as redes neurais reais. Por outro lado existem paralelos entre os dois mundos que prometem que as redes neurais artificiais sejam uma aproximação apropriada para resolver problemas cognitivos complexos.

Neste material tenta-se dar uma breve introdução ao campo de computação neural. As restrições de espaço permitem unicamente a apresentação de alguns dos modelos e algoritmos de redes neurais artificiais. Tenta-se dar uma idéia básica sobre as capacidades e limitações desse área de pesquisa e engenharia. Para o leitor interessado dão-se referências para o aprofundamento da matéria.

I.1 A Inspiração da Neurociência

Quais são as qualidades do cérebro humano que o capacitam de um comportamento inteligente? Os seguintes tópicos refletem as mais importantes características que são especialmente atrativas para serem simuladas em uma rede neural artificial:

Robustez e tolerância a falhas: A eliminação de alguns neurônios não afeta a funcionalidade global.

Capacidade de aprendizagem: O cérebro é capaz de aprender novas tarefas que nunca foram executadas antes.

Processamento de informação incerta: Mesmo que a informação fornecida esteja incompleta, afetada por ruído ou parcialmente contraditória, ainda um raciocínio correto é possível.

Paralelismo: Um imenso número de neurônios está ativo ao mesmo tempo. Não existe a restrição de um processador que obrigatoriamente trabalhe uma instrução após a outra.

O processamento local de informação no cérebro efetua-se em cerca de 10^{11} unidades (os *neurônios*) que têm uma estrutura relativamente simples. Na Figura 1 apresenta-se o modelo simplificado de um único neurônio real. O neurônio é uma célula com *núcleo* e corpo (*soma*) onde reações químicas e elétricas representam o processamento de informação. A saída da informação do soma é realizada por impulsos elétricos que se propagam através do *axônio*. No final do axônio existem inúmeras ramificações que distribuem a informação para outros neurônios vizinhos. A ligação com outros neurônios é realizada através de *sinapses* que estão conectadas a um *dendrite* do neurônio receptor. A sinapse dispara uma substância química

quando for excitada pelo impulso do axônio. A substância se transmite entre sinapse e dendrite realizando a conexão entre dois neurônios vizinhos. Conforme as excitações (ou inibições) que células vizinhas transmitem para a célula em consideração ela processa a informação novamente e a transmite via seu axônio.

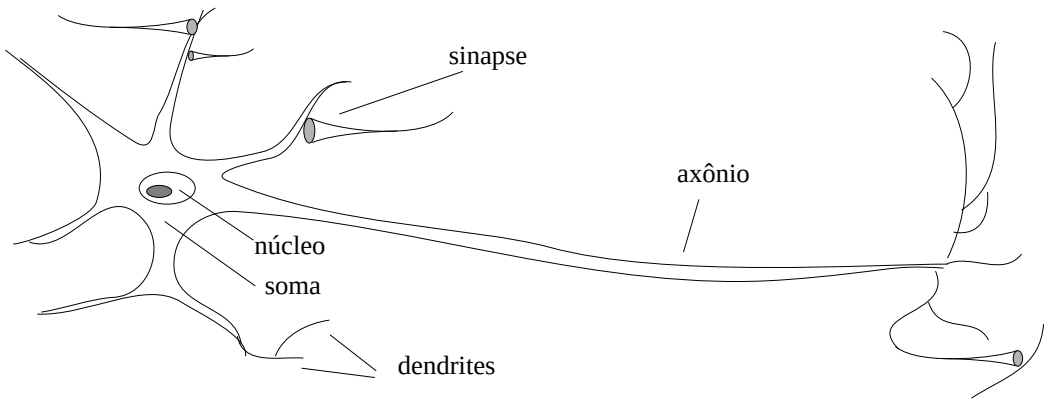

Figura 1 - Neurônio biológico

I.2 História da Neurocomputação

Um ponto marcante na história das redes neurais artificiais foi a apresentação de um modelo de um neurônio artificial por [McCulloch and Pitts, 1943]. As atividades nessa linha de pesquisa culminaram na concepção do *perceptron* por [Rosenblatt, 1958] e em um modelo parecido, o *adaline* por [Widrow and Hoff, 1960]. O perceptron é capaz de *classificar* entre classes que são linearmente separáveis. Foi usado para reconhecer por exemplo caracteres. Essa aplicação foi realizada em uma máquina chamada MARK I PERCEPTRON e causou uma grande euforia certamente exagerada em relação a imaginação das capacidades de futuros robôs inteligentes. A característica importante do perceptron foi a apresentação de um algoritmo de aprendizagem capaz de adaptar os pesos internos do neurônio de maneira que seja capaz de resolver o problema de classificação linear, em caso da separabilidade linear das classes.

O caso exemplar das limitações do perceptron é o problema “Ou exclusivo” (XOR) ($f(0,0) = f(1,1) = 0, f(0,1) = f(1,0) = 1$) que prova que uma função tão simples de classificação não pode ser calculada pelo perceptron. Essa crítica centrou-se no livro “Perceptrons” de [Minsky and Papert, 1969]. O impacto dessa crítica foi tão grande que a comunidade científica abandonou a área das redes neurais artificiais, com a exceção de alguns pesquisadores por exemplo Fukushima, Grossberg, Hopfield e Kohonen.

A solução para o problema XOR já era conhecida. Bastava acrescentar mais uma camada de neurônios na rede (uma camada escondida). O que faltava era um algoritmo que fosse capaz de treinar os pesos dessa rede multi-camada para que pudesse classificar corretamente problemas mais complexos. Várias soluções equivalentes foram descobertas durante os anos seguintes, más só a publicação do algoritmo de “retropropagação de erro” (error backpropagation) por [Rumelhart et al., 1986] popularizou uma solução de carácter universal para esse tipo de problema. A partir desse momento, surgiram os modelos que foram desenvolvidos durante os anos tranquilos da pesquisa e inúmeros outros modelos de redes neurais artificiais junto com algoritmos de aprendizagem foram apresentados. Espera-se para o futuro que o paradigma da neurocomputação prove ser uma ferramenta potente para resolver problemas complexos.

I.3 Referências para Aprofundamento na Matéria de Redes Neurais

Além de uma idéia superficial, este texto não pode transmitir muita informação sobre esse tema altamente interessante e desafiador. O leitor interessado deve ser dirigido para uma escolha boa de livros de texto e recursos eletrônicos. Leitores com acesso a Internet deveriam ler as "Questões mais perguntadas" ("FAQ — Frequently Asked Questions") do grupo de discussões USENET com o endereço news:comp.ai.neural-nets. Nessas questões dão-se recomendações de livros e artigos para uma maior especialização. Recomenda-se [Hinton, 1992] como artigo de introdução mais popular na área. Livros para iniciantes, parcialmente junto com código, incluem [Masters, 1994], [Fausett, 1994] e [Anderson, 1995], de nível intermediário e avançado [Bishop, 1995], [Hertz et al., 1991], [Haykin, 1994] e [Ripley, 1996].

Além das recomendações de livros encontram-se explicações básicas relacionadas aos tópicos de redes neurais artificiais (RNA), por exemplo respostas às perguntas: "O que é uma rede neural?", "O que se pode fazer com redes neurais artificiais e o que não?", "Quais são as aplicações possíveis?", entre outras.

A maioria das implementações de RNA é feita em software. Existem uma série de simuladores publicamente disponíveis, com código fonte e manual. Um simulador especialmente potente para o sistema operacional UNIX, que cobre a maioria das arquiteturas e algoritmos de aprendizagem é o "Stuttgart Neural Network Simulator" (SNNS). O programa está implementado em C e possui uma interface gráfica extensa. O endereço na internet desse software é ftp://ftp.informatik.uni-stuttgart.de/pub/SNNS.

II. FUNDAMENTOS

Uma rede neural artificial (RNA) tem duas facetas elementares: a arquitetura e o algoritmo de aprendizagem. Essa divisão surge naturalmente pelo paradigma como a rede é treinada. Ao contrário de um computador com arquitetura de von Neumann que é programado, a rede é treinada por exemplos de treino. O conhecimento sobre o problema em consideração está guardado dentro dos exemplos que têm que estar obrigatoriamente disponíveis. O algoritmo de aprendizagem generaliza esses dados e memoriza o conhecimento dentro dos parâmetros adaptáveis da rede, os pesos. Assim o construtor de um sistema baseado em RNA tem dois graus de liberdade, a definição sobre o tipo de rede para resolver o problema em consideração e o algoritmo para treinar a rede, i.e. para adaptar os pesos da rede.

A composição da rede é feita pelos neurônios. Normalmente o tipo de processamento de um único neurônio é a combinação linear das entradas com os pesos seguida pela passagem da combinação linear por uma função de ativação.

A natureza do problema a ser resolvido normalmente define restrições em relação aos tipos de redes e algoritmos de aprendizagem possíveis. Neste texto distinguem-se redes com propagação do fluxo de informação para frente, redes recorrentes (com realimentação das saídas para as entradas) e redes competitivas. Em relação aos algoritmos de adaptação, vamos distinguir entre aprendizagem supervisionada e aprendizagem não-supervisionada.

II.1 Modelo de Neurônio Artificial

Em primeiro lugar, vamos introduzir o modelo simplificado de um neurônio e as capacidades de processamento associadas. Na Figura 2 mostra-se o modelo de um neurônio artificial de [McCulloch and Pitts, 1943]. Este modelo tenta simular as realidades biológicas que ocorrem dentro de uma célula do sistema nervoso, compare Figura 1. A informação fornecida por

outros neurônios entra em D entradas x_j (=sinapses) no neurônio processador. O processamento consiste de uma combinação linear das entradas $net = w_1x_1 + w_2x_2 + \ldots + w_Dx_D = \sum_{j=1}^{D} w_jx_j = \underline{w}^T\underline{x}$. A cada entrada está associada um peso w_j que reflete a importância da entrada x_j. O resultado dessa combinação linear é o valor net. Se esse valor ultrapassar um limiar μ, o neurônio "dispara" o valor 1 na saída binária y, se não ultrapassar o limiar a saída fica passiva em $y = 0$. A comparação de net com o limiar μ é realizada pela função de Heaveside (função de escada) $\Theta(x) = 1$ se $x \geq 0$ e $\Theta(x) = 0$ caso contrário.

$$y = \Theta(\sum_{j=1}^{D} w_jx_j - \mu) \tag{1}$$

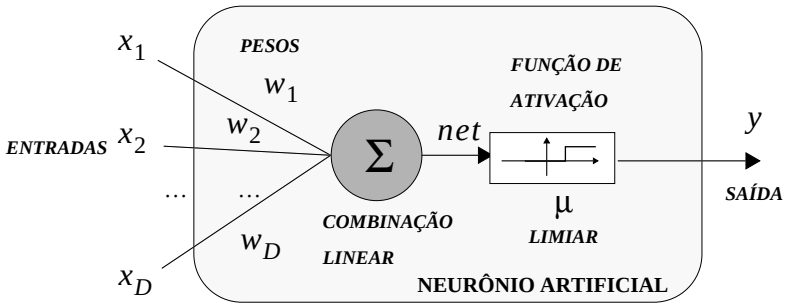

Figura 2 - Modelo de um neurônio de McCulloch e Pitts

A função de ativação no caso do modelo de [McCulloch and Pitts, 1943] não é a única maneira de produzir o valor de saída do neurônio. Figura 3 mostra diferentes tipos de funções de ativação. A função linear produz uma saída linear contínua, a função de escada, uma saída binária (não-linear discreta) e a função sigmoidal, uma saída não-linear contínua.

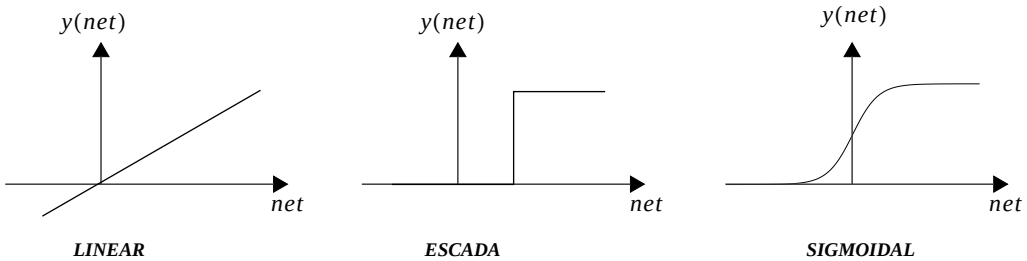

Figura 3 - Funções de Ativação

A definição da função sigmoidal é

$$\textit{Função sigmoidal: } g(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

e tem um conjunto de propriedades que se mostrarão muito úteis nos cálculos relacionados à aprendizagem dos pesos e ao mapeamento realizado pela rede:

— Não linear
— Contínua e diferenciável em todo o domínio de $\boldsymbol{R}$
— Derivada tem forma simples e é expressa pela própria função: $g'(z) = g(z)(1 - g(z))$
— Estritamente monótona: $z_1 \leq z_2 \Leftrightarrow g(z_1) \leq g(z_2)$

Em termos do domínio dos valores calculados y_i distingue-se basicamente entre saídas binárias com $y_i \in \{0, 1\}$ ou $y_i \in \{1, -1\}$ e saídas contínuas com $y_i \in \boldsymbol{R}$.

Eventualmente o neurônio possui uma memória local, i.e. o estado de ativação anterior é tomado em consideração no cálculo da ativação atual. Esse tipo de processamento dinâmico está fora do âmbito desse texto.

II.2 Topologia de Redes de Neurônios Artificiais

Acabamos de definir uma entidade de processamento relativamente simples que calcula uma função de saída y a partir das entradas x_j e dos pesos w_j, com uma função de ativação predefinida. O potencial e flexibilidade do cálculo baseado em redes neurais vêm da criação de conjuntos de neurônios que estão interligados entre si. Esse paralelismo de elementos com processamento local cria a "inteligência" global da rede. Um elemento da rede recebe um estimulo nas suas entradas, processa esse sinal e emite um novo sinal de saída para fora que por sua vez é recebido pelos outros elementos.

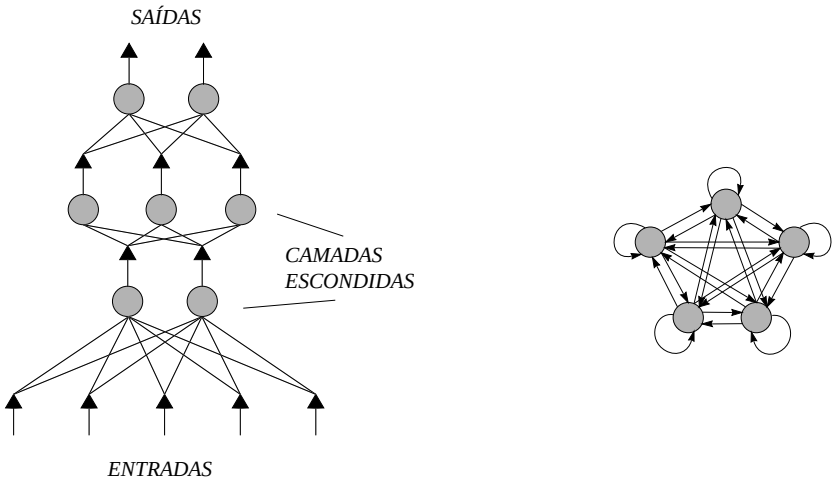

Figura 4 - Topologias principais de redes neurais artificiais

Uma categorização fundamental da topologia dos neurônios pode ser feita em relação ao método de propagação da informação recebida, veja Figura 4. Pode-se distinguir entre redes de propagação para frente (*feedforward*) e redes realimentadas (*recurrent*). No caso das redes de propagação para frente o fluxo de informação é unidirecional. Neurônios que recebem a informação simultaneamente agrupam-se em *camadas*. Camadas que não estão ligadas às entradas e nem às saídas da rede chamam-se *camadas escondidas*. Exemplos para esse tipo de rede são o perceptron [Rosenblatt, 1958], o perceptron multi-camada [Rumelhart et al., 1986] e

o ADALINE [Widrow and Hoff, 1960]. Uma rede que adicionalmente tem uma relação topológica de vizinhança entre os neurônios é o mapa auto-organizável de Kohonen [Kohonen, 1972], [Kohonen, 1990].

Redes realimentadas têm ligações entre os neurônios sem restrições. Ao contrário das redes sem realimentação, o comportamento dinâmico desempenha o papel fundamental nesse modelo. Em alguns casos os valores de ativação da rede passam por um processo de relaxação até chegarem a um estado estável. O modelo que se apresentará como representante é a rede auto-associativa de [Hopfield, 1982].

II.3 Paradigmas de Aprendizagem

Uma vez definida a rede neural, essa tem que ser treinada. Isso significa que os graus de liberdade que a rede dispõe, para solucionar a tarefa em consideração, têm que ser adaptados de uma maneira ótima. Normalmente, isso significa que temos que modificar os pesos w_{ij} entre o neurônio i e o neurônio j, segundo um algoritmo. Um conjunto finito T de n *exemplos de treino* está à nossa disposição para adaptar os pesos durante a fase de treinamento da rede.

Uma distinção principal em relação ao paradigma de aprendizagem que é válido para todo tipo de sistemas com capacidade de adaptação é *aprendizagem supervisionada* e *aprendizagem não-supervisionada*.

II.3.1 Aprendizagem supervisionada

Na aprendizagem supervisionada cada exemplo de treino está acompanhado por um valor que é o valor desejado. Isso significa que o conjunto de treino T está composto por n pares de exemplos $(\underline{x}_p, \underline{y}_p)$ onde $T = \{(\underline{x}_p, \underline{y}_p)\}_{p=1}^{n}$. A dimensão do vetor de entrada é D, i.e. as variáveis de entrada estão agrupadas em um valor multidimensional (vetor de coluna), normalmente do domínio dos números reais: $\underline{x} = (x_1, \ldots, x_j, \ldots, x_D)^T$, $x_j \in \boldsymbol{R}$. (A transposição $(.)^T$ é usada para economizar espaço escrevendo um vetor em uma linha). As variáveis de saída estão agrupadas em um vetor de saída $\underline{y} = (y_1, \ldots, y_i, \ldots, y_c)^T$.

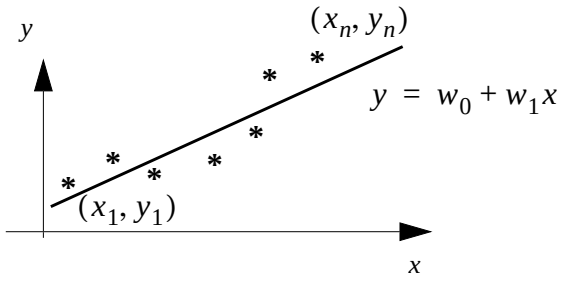

Figura 5 - Regressão linear

Um exemplo de uma tarefa de aprendizagem supervisionada é a regressão linear. Usamos o caso unidimensional para facilitar a ilustração, veja Figura 5. Nesse problema o conjunto de treino consiste em pares de números reais (x_p, y_p). O objetivo da aprendizagem é a determinação de coeficientes w_0 e w_1 da reta $y = w_0 + w_1 x$. O algoritmo de aprendizagem

tenta minimizar a discrepância entre o valor desejado y_p e o valor que é a resposta $y' = w_0 + w_1 x_p$ do sistema, e isso em média para cada exemplo (x_p, y_p).

II.3.2 Aprendizagem não-supervisionada

Quando a única informação disponível são os valores $(\underline{x}_p)$ a tarefa de aprendizagem é descobrir correlações entre os exemplos de treino $T = \{(\underline{x}_p)\}_{p=1}^{n}$. O número de categorias ou classes não está definido a priori. Isso significa que a rede tem que achar atributos estatísticos relevantes, ela tem que desenvolver uma representação própria dos estímulos que entram na rede. Um sinônimo para aprendizagem não-supervisionada é aglomeração ("clustering").

Um exemplo da área de medicina é a detecção de doenças a partir de imagens, por exemplo imagens de raio-X. Existem várias regiões dentro da imagem que se deixam atribuir ao mesmo material, por exemplo osso. O número dos materiais (das aglomerações) não é conhecido a priori. O objetivo do sistema é descobrir o número dos materiais diferentes e ao mesmo tempo categorizar cada ponto da imagem para o respectivo material. A entrada para a rede seriam os pontos da imagem, por exemplo uma pequena janela de 5 por 5 pontos. A resposta ideal da rede seria o material a qual pertence essa região da imagem.

II.4 Regras de Adaptação dos Pesos

Durante o processo de aprendizagem os pesos normalmente percorrem uma modificação iterativa. O peso entre neurônio i e neurônio j seja w_{ij}, veja Figura 6. Na iteração l o peso $w_{ij}^{(l)}$ influencia a função calculada pela rede. O algoritmo de aprendizagem julga a qualidade do peso e eventualmente determina se o peso deve sofrer uma modificação no seu valor de uma diferença $\Delta w_{ij}^{(l)}$ na próxima iteração $l + 1$. Assim, se define a regra básica de adaptação dos pesos:

$$\textit{Adaptação de peso}: w_{ij}^{(l+1)} = w_{ij}^{(l)} + \Delta w_{ij}^{(l)} \tag{3}$$

Costuma-se inicializar os pesos aleatoriamente. O algoritmo de aprendizagem percorre um número fixo de iteração e/ou até que uma condição de parada seja atingida, por exemplo numa aprendizagem supervisionada a discrepância entre o valor calculado e o valor desejado desaparece para todos os exemplos de treino.

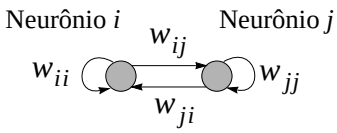

Figura 6 - Pesos entre neurônios, caso geral com realimentação

II.4.1 Aprendizagem pela Regra de Hebb

Um dos trabalhos pioneiros nos estudos de sistemas capazes de aprender foi feito por [Hebb, 1949]. Ele criou uma hipótese de que o peso de ligação entre dois neurônios que estão ativos

aos mesmo tempo deve ser reforçado. Para o nosso modelo essa lei traduz-se para a

$$\textit{Regra de aprendizagem de Hebb}: \Delta w_{ij} = \eta y_i y_j \tag{4}$$

onde a taxa de aprendizagem η é um fator de escala positivo que determina a velocidade da aprendizagem. A definição dessa regra baseia-se em estudos biológicos do cérebro, mas como já foi constatado, a correspondência do modelo matemático com a realidade biológica é somente uma idealização aproximada. A regra de Hebb define um algoritmo de adaptação dos pesos, porém sem a definição de um objetivo a atingir, por exemplo, minimizar um erro entre um valor desejado e calculado. No estudo da rede de Hopfield voltaremos à regra de Hebb e vamos provar que tem utilidade prática.

II.4.2 Aprendizagem pela Regra de Delta

Uma regra de adaptação dos pesos com um objetivo bem visível é a regra de delta ou regra de Widrow-Hoff [Widrow and Hoff, 1960]. A rede calcula na saída (no neurônio i) uma função y'_i. Na aprendizagem supervisionada conhece-se o valor desejado y_i que a rede deve calcular. Assim pode-se calcular o *erro* $e_i = y_i - y'_i$ entre o calculado e o desejado. O peso entre o neurônio i e o neurônio j que é responsável por esse erro então deve ser modificado proporcional à ativação e ao erro, outra vez escalado por uma taxa de aprendizagem η:

$$\textit{Regra de Delta}: \Delta w_{ij} = \eta e_i y_j = \eta (y_i - y'_i) y_j \tag{5}$$

Neste caso, o objetivo do algoritmo de aprendizagem está bem claro, nomeadamente minimizar o erro entre os valores calculados pela rede e desejados pelos exemplos fornecidos num problema de aprendizagem supervisionada.

II.4.3 Aprendizagem Competitiva

Consideram-se as redes de neurônios onde um único neurônio pode ser ativo ao mesmo tempo. Isso significa que todos os outros neurônios têm uma ativação igual a zero $y_i = 0$ para $i \neq i^*$ e somente o *vencedor* i^* emite um sinal de ativação $y_{i^*} = 1$.

$$\textit{Aprendizagem Competitiva}: \Delta w_{ij} = \eta y_i (x_j - w_{ij}) \tag{6}$$

O efeito dessa regra é que os pesos $\underline{w}_i$ se deslocam em direção do estímulo (entrada) da rede $\underline{x}$.

Vamos considerar a rede de Kohonen como exemplo de uma rede neural artificial que usa aprendizagem competitiva para adaptar os pesos.

II.5 Taxinomia de Redes

Resumindo o paradigma de aprendizagem e as regras de adaptação dos pesos pode-se criar uma divisão hierárquica para os modelos de redes neurais apresentados nesse texto, veja Figura 7. Os modelos que têm uma capacidade de aprendizagem dividem-se em modelos que usam aprendizagem supervisionada e aprendizagem não supervisionada, dependendo se para cada estímulo $\underline{x}$ um sinal com a resposta desejada da rede y está disponível ou não. Perceptron, perceptron multi-camada e adaline são modelos com aprendizagem supervisionada que se baseiam no erro entre a resposta desejada e calculada da rede para adaptar os pesos. A rede de Hopfield usa a regra de Hebb para memorizar um conjunto de padrões. Na aprendizagem não

supervisionada a regra de Hebb também pode ser usada, por exemplo para estimar a densidade de probabilidade com base nos exemplos dados. Uma arquitetura de rede com um algoritmo competitivo é a rede topológica de Kohonen.

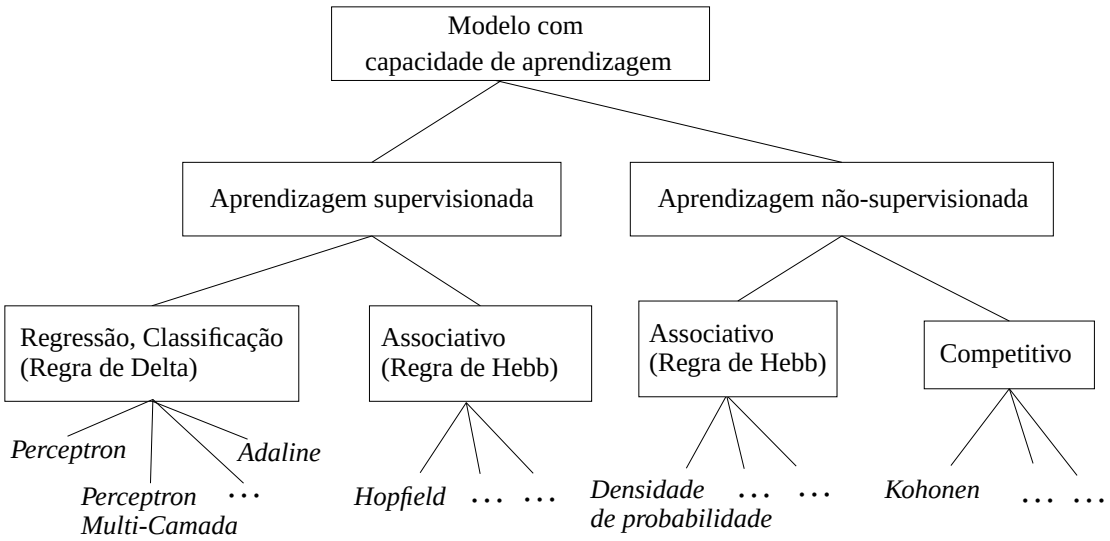

Figura 7 - Classificação estrutural e funcional de redes neurais artificiais

III. REDES DE PROPAGAÇÃO PARA FRENTE

Considera-se aprendizagem supervisionada em redes de propagação para frente que estão organizadas em camadas. No início das atividades de pesquisa essas redes chamaram-se perceptrons. O caso mais simples é o perceptron com uma única camada. O domínio dos valores de saída é binário. O ADALINE permite variáveis de saída com valores contínuos. Quando existe mais que uma camada, i.e. existe uma ou mais camadas escondidas, trata-se de perceptrons multi-camada. O fluxo de informação é sempre unidirecional, ao contrário de redes com realimentação. Existem pesos (assimétricos) unidirecionais entre dois neurônios que necessariamente têm que estar em camadas diferentes.

III.1 Perceptron

Classificação é uma das aplicações principais do cálculo que as redes neurais são capazes de realizar. O objetivo é associar uma categoria de um universo finito a um objeto. Exemplos para classificação são:

- Reconhecimento automático de caracteres
- Detecção de falhas em processos
- Identificação de pessoas por impressões digitais, voz, iris do olho
- Diagnóstico médico

O perceptron [Rosenblatt, 1958] é capaz de classificar entre duas classes que linearmente são separáveis, veja Figura 8 (O modelo é extensível facilmente para o caso de várias classes). Junto com a arquitetura foi proposto um método de como os pesos podem ser adaptados.

Também foi dada uma prova formal da convergência em um número finito de iterações desse algoritmo em caso da separabilidade linear.

A função que o perceptron implementa é a do neurônio de McCulloch e Pitts (1), onde a função de escada $\Theta(.)$ é substituída pela função do sinal $\mathrm{sgn}(z) = 1$ se $z \geq 0$ e $\mathrm{sgn}(z) = -1$ se $z < 0$. Pode-se absorver o limiar μ no cálculo como $\mu = -w_0$, introduzindo um novo valor de entrada fixo $x_0 = 1$:

$$\textit{Regra de Classificação do Perceptron}: d(\underline{x}) = \mathrm{sgn}(\sum_{j=0}^{D} w_j x_j) \qquad (7)$$

A função calculada $d(\underline{x})$ fornece uma resposta binária de "que lado" está o objeto $\underline{x} = (x_1, x_2)^T$, assim permitindo uma classificação linear entre duas classes.

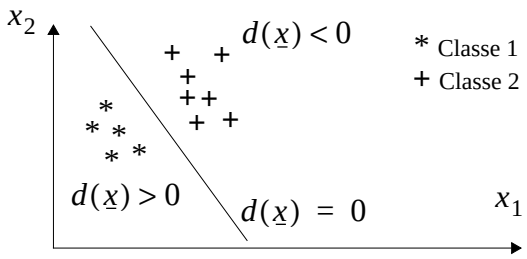

Figura 8 - Problema de classificação. Duas classes são linearmente separáveis. A reta $d(\underline{x}) = 0$ separa as duas classes. O objectivo do algoritmo de perceptron é achar pesos w_0, w_1 e w_2 para a definição do hiperplano separador (nesse caso a reta $d(\underline{x}) = w_0 + w_1 x_1 + w_2 x_2$).

A estrutura da rede então foi definida simplesmente pela equação linear de (7). Resta agora o algoritmo de adaptação dos pesos, o algoritmo de aprendizagem de perceptron. A ideia básica é uma adaptação iterativa nos moldes de (3) por um algoritmo iterativo. O algoritmo classifica todos os objetos de treino $T = \{(\underline{x}_p, t_p)\}_{p=1}^{n}$, com $\underline{x}_p = (x_{1p}, x_{2p})^T$ neste caso ilustrativo de duas dimensões pela regra (7) $d(\underline{x}_p)$. A classe verdadeira de $\underline{x}_p$ está disponível no *valor de alvo* t_p. Se nenhum erro de classificação ocorreu temos $d(\underline{x}_p) = t_p$. Nesse caso estamos satisfeitos com o conjunto dos pesos $W = \{w_j\}_{j=1}^{D}$. Se $d(\underline{x}_p) \neq t_p$ ocorreu um erro de classificação. Todos os objetos $\underline{x}_p$ que provocaram um erro de classificação são usados para modificar os pesos W para que o número de erros diminua e finalmente desapareça.

Define-se uma regra simples de modificação de pesos

Regra de aprendizagem do perceptron (versão simples):

$$\Delta w_j = \eta t_p x_j \text{ se } d(\underline{x}_p) \neq t_p \text{ e } \Delta w_j = 0 \text{ se } d(\underline{x}_p) = t_p \qquad (8)$$

Escolhem-se pesos aleatórios inicialmente. A regra (8) adapta os pesos em um número finito de iterações, se existe uma separação linear.

O perceptron é um sistema de uma rede neural simples (nesse caso $D + 1$ entradas, uma saída), capaz de resolver problemas lineares de classificação. Está equipado com um algoritmo

de adaptação de pesos que aprende baseado nos exemplos de treino. Obviamente a capacidade de cálculo do perceptron está limitada pela separabilidade linear das classes. Para classificar problemas em que as classes se distribuem de tal maneira que seja impossível criar um hiperplano para separá-las, técnicas mais sofisticadas têm que ser usadas, como o perceptron multi-camada.

III.2 ADALINE

Vimos no exemplo anterior do perceptron que as saídas estavam limitadas para terem valores binários. Um modelo muito parecido com o perceptron em termos de arquitetura é o ADALINE [Widrow and Hoff, 1960]. A diferença porém está nas saídas contínuas, i.e. permite-se calcular valores de saída do domínio dos números reais, veja Figura 9. A função calculada é simplesmente a combinação linear dos pesos e das entradas, ou equivalentemente o produto interno do vetor de pesos e o vetor das entradas:

$$\textit{Função do ADALINE: } d(\underline{x}) = \sum_{j=0}^{D} w_j x_j = \underline{w}^T \underline{x} \qquad (9)$$

Foi outra vez introduzido uma entrada com um valor constante de 1 que facilita a representação da função calculada.

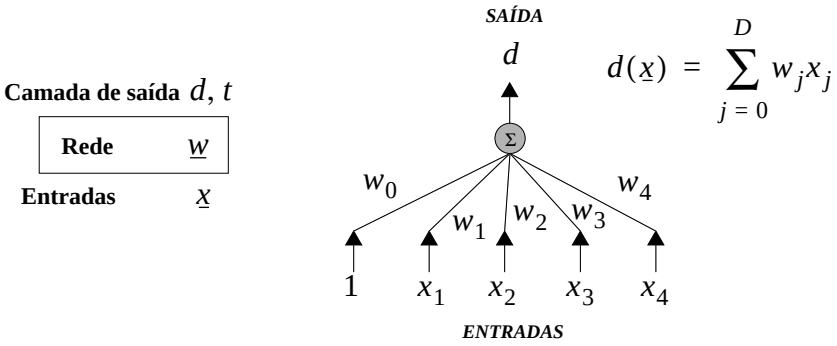

Figura 9- ADALINE com 4 variáveis de entrada

Outra vez o problema para resolver (a função para calcular) está especificado nos n exemplos de treino: $T = \{(\underline{x}_p, t_p)\}_{p=1}^{n}$, compare com o perceptron. Os graus de liberdade que existem para realizar essa *aproximação de função*[1] são os pesos $\underline{w} = (w_0, w_1, \ldots, w_D)^T$. O objetivo do algoritmo de aprendizagem está na busca dos pesos que exibem uma propriedade ótima em relação a função a ser calculada.

---

1. Se tivéssemos c saídas d_i e não só uma saída d, teríamos que introduzir mais um índice i para os pesos w_{ij}, assim indicando a ligação entre entrada j e saída i. Em vez de ter um vetor de pesos $\underline{w} = (w_0, w_1, \ldots, w_D)^T$ teríamos uma matriz de pesos $W = (\underline{w}_1^T, \ldots, \underline{w}_c^T)^T$ de dimensão $c \times (D+1)$, com $\underline{w}_i = (w_{i0}, w_{i1}, \ldots, w_{iD})^T$. O vetor de funções seria $W\underline{x} = \underline{d}(\underline{x})$.

III.2.1 O erro quadrático mínimo

Uma escolha natural para medir a qualidade da função calculada pela rede é a diferença entre o valor desejado para o exemplo $\underline{x}_p$ e o valor calculado pela rede:

$$\textit{Erro de cálculo para um exemplo } \underline{x}_p\text{: } e(\underline{x}_p) = desejado(\underline{x}_p) - calculado(\underline{x}_p) \qquad (10)$$

O valor desejado é o valor de alvo t_p. O valor calculado pela rede é $d(\underline{x}_p)$. Como o erro (10) pode ser negativo ou positivo, calcula-se o quadrado para sempre ter uma diferença positiva que não se elimina, considerando todos os exemplos individuais $\underline{x}_p$ do conjunto de treino T.

$$\textit{Erro quadrático para um exemplo } \underline{x}_p\text{: } e^2(\underline{x}_p) = (t_p - d(\underline{x}_p))^2 = (t_p - \underline{w}^T \underline{x}_p)^2 \qquad (11)$$

O objetivo do algoritmo vai ser a minimização do erro, considerando a média de todos os n exemplos (valor esperado $E\{e^2(\underline{x}_p)\}$). Isso permite definir o critério a ser minimizado:

$$\textit{Erro quadrático médio: } EQM = \frac{1}{n}\sum_{p=1}^{n} e^2(\underline{x}_p) = \frac{1}{n}\sum_{p=1}^{n} (t_p - \underline{w}^T \underline{x})^2 \qquad (12)$$

III.2.2 Solução determinística

Para esse tipo de problema linear existe uma solução explícita para obter aquele vetor de pesos $\underline{w}$ que minimize (12). Pode-se juntar todos os n exemplos de treino $\underline{x}_p$ em uma única matriz X de dimensão $n \times (D+1)$: $X = [\underline{x}_1^T, \ldots, \underline{x}_n^T]^T$, com $\underline{x}_p = (x_{p1}, \ldots, x_{pD})^T$. Em analogia pode-se juntar todos os valores de alvo t_p num vetor de alvo $\underline{t} = (t_1, \ldots, t_n)^T$ de dimensão $n \times 1$.

O exemplo $\underline{x}_p$ deve ser mapeado pela rede para $\underline{w}^T \underline{x}_p = t_p$. Assim pode-se formular o cálculo de todos os n exemplos de treino $\underline{x}_p$ numa única equação de dimensão $(n \times (D+1))((D+1) \times 1) = (n \times 1)$ de vetores e matrizes:

$$\textit{Mapeamento de ADALINE de todos os exemplos: } X\underline{w} = \underline{t} \qquad (13)$$

Uma pré-multiplicação de (13) pela matriz transposta X^T de X resulta em $X^T X \underline{w} = X^T \underline{t}$. Outra pré-multiplicação pela inversa $(X^T X)^{-1}$ de $X^T X$ (que sempre existe, sem prova) finalmente resulta na solução explícita do problema:

$$\text{Solução determinística do } \textit{ADALINE}\text{: } \underline{w} = (X^T X)^{-1} X^T \underline{t} \qquad (14)$$

onde a matriz $X^\dagger = (X^T X)^{-1} X^T$ é denominada como *Pseudoinversa* de X.

III.2.3 Solução iterativa: Descida de Gradiente

Acabamos de ver que existe uma solução direta para calcular o conjunto de pesos que

minimiza o critério de qualidade do mapeamento da rede. Esse solução deve-se principalmente à natureza linear do problema, possibilitando assim a solução por álgebra linear. Em caso de redes que realizam um mapeamento não-linear, como no caso do perceptron multi-camada com função de ativação sigmoidal, essa solução determinística já não pode ser encontrada. Temos que empregar técnicas de otimização de problemas sem restrições. Analisaremos como caso exemplar a técnica da *descida de gradiente*. Em primeiro lugar apresenta-se esse método para o caso linear do ADALINE. Como vamos ver no caso do perceptron multi-camada a descida de gradiente constitui o veículo para o conhecido algoritmo de retropropagação de erro.

Figura 10 mostra a idéia principal da técnica da descida de gradiente. O erro quadrático médio EQM é uma função dos pesos da rede $EQM(\underline{w}) = f(\underline{w})$ (para fins de ilustração usa-se só um único peso w). Essa função em geral não é conhecida para quem está procurando os pesos ótimos (se fosse conhecida existia uma solução determinística). O objetivo geral é encontrar o peso w_{min} que minimize o erro EQM (12), i.e. que faz a rede aproximar da melhor maneira possível o mapeamento entre todos os n exemplos $\underline{x}_p$ para os valores de alvo t_p.

Tenta-se chegar *iterativamente* ao mínimo global w_{min}. A única informação que é conhecida na iteração l é o valor do erro $EQM(w^{(l)}) = E(w^{(l)})$ para o peso $w^{(l)}$ atual. Supõe-se que a função do erro seja derivável em todo o domínio. Isso significa que o gradiente da função de erro $\nabla E = dE(\underline{w})/d\underline{w}$ existe (no caso de um peso: $E'(w) = dE/dw$).

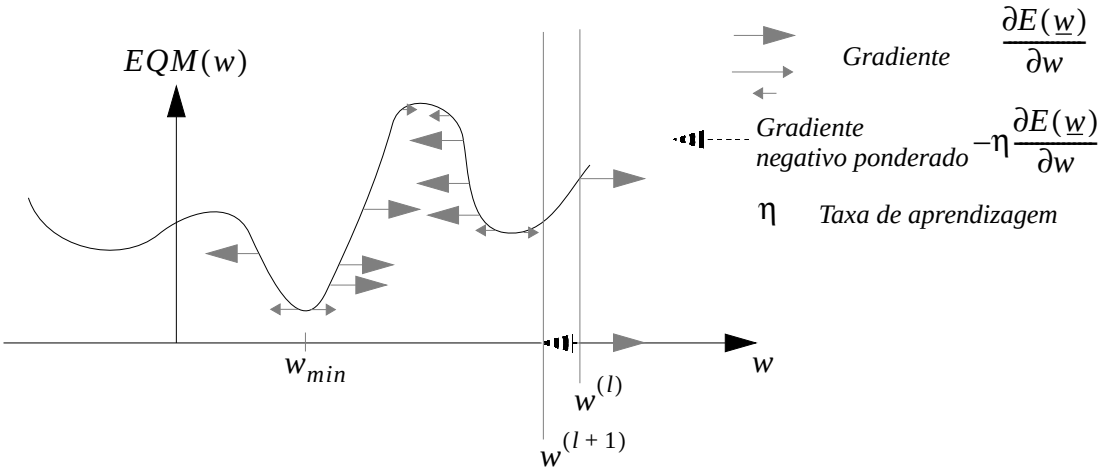

Figura 10- Descida de Gradiente. Considera-se um único peso $w = w_j$.

O gradiente é um vetor. Ele aponta na direção do *crescimento* da função E. Consequentemente o gradiente negativo aponta na direção de *decrescimento* da função E. A tentativa para chegar no mínimo da função então é a modificação do peso na iteração l para a iteração $l+1$ na direção do gradiente negativo $-\nabla E$ (da descida do gradiente). Para controlar a velocidade da modificação do peso de $w^{(l)}$ para $w^{(l+1)}$ usa-se um fator de escala, a *taxa de aprendizagem* η. Temos então no espírito da técnica da adaptação dos pesos (3) uma regra de como vamos procurar aquele peso que minimize o erro de mapeamento da rede:

Regra de adaptação de peso por descida de gradiente:

$$\underline{w}^{(l+1)} = \underline{w}^{(l)} + \Delta\underline{w}^{(l)} = \underline{w}^{(l)} - \eta\nabla E^{(l)} \tag{15}$$

Agora fica mais clara a necessidade da propriedade da função sigmoidal (2) de que ela seja contínua e diferenciável em todo o domínio de $\boldsymbol{R}$. Assim consegue-se calcular o gradiente para uma rede com função de mapeamento não-linear, como ainda vamos verificar mais tarde.

Começa-se com um valor aleatório do peso $w^{(0)}$ na iteração 0. A regra de adaptação de peso é aplicada um número fixo de vezes ou até que a diferença entre dois conjuntos de pesos consecutivos $\underline{w}^{(l+1)}$ e $\underline{w}^{(l)}$ seja menor do que uma tolerância eps. A diferença poderia por exemplo ser a distância Euclidiana entre os dois vetores de pesos. Todos os passos estão resumidos no Algoritmo 1.

Algoritmo 1: Descida de Gradiente

Objetivo: Aprende uma vetor de pesos ótimos $\underline{w}_{opt}$ que minimize a função de custos E

0.) Número máximo de iterações: l_{max}

Diferença mínima entre dois vetores de pesos consecutivos: eps

Taxa de aprendizagem: η

1.) $l = 0$, $\underline{w}^{(0)}$ arbitrário

2.) ***Repete*** iteração l

2.1) Calcule o gradiente do erro relativo aos pesos individuais w_j: $\nabla E_j^{(l)} = \partial E^{(l)} / \partial w_j^{(l)}$

2.2) Adapte os pesos na direção oposta ao gradiente

$$w_j^{(l+1)} = w_j^{(l)} + \Delta w_j^{(l)} = w_j^{(l)} - \eta\nabla E_j^{(l)}$$

até $(l > l_{max}$ ou $\left\|\underline{w}^{(l+1)} - \underline{w}^{(l)}\right\| = \sqrt{\sum_{j=0}^{D}(w_j^{(l+1)} - w_j^{(l)})^2} < eps)$

Problemas com o algoritmo são a escolha da taxa de aprendizagem η e a ocorrência de mínimos locais. Se η for escolhida muito pequena a aprendizagem poderia ficar lenta, se for escolhida grande demais poderiam surgir oscilações do erro com uma divergência dos pesos, i.e. os pesos não param de crescer. Mínimos locais são locais na função de erro onde o gradiente desaparece, portanto já não há uma modificação dos pesos. Assim o algoritmo pode parar num conjunto de pesos que não corresponde ao mínimo global da função de erro o que seria desejável.

Resta definir a própria função de erro. Temos duas possibilidades de como modificar os pesos:

1.) Depois da apresentação de cada exemplo $\underline{x}_p$, i.e. $E = E(\underline{x}_p) = e^2(\underline{x}_p)$ (aprendizagem estocástica, apresentação dos exemplos em ordem aleatória). Neste caso calcula-se o gradiente relativo ao peso individual w_j como:

$$\nabla E_j = \partial E / \partial w_j = \partial(e^2(\underline{x}_p))/\partial w_j = 2e(\underline{x}_p)(\partial e(\underline{x}_p)/\partial w_j)$$
$$= 2e(\underline{x}_p)(\partial(t_p - \sum_{j=0}^{D} w_j x_{pj})/\partial w_j) = -2e(\underline{x}_p)x_{pj}$$

2.) Depois da apresentação de todos os exemplos $\underline{x}_p$, i.e. $E = \sum_{p=1}^{n} e^2(\underline{x}_p)$ (aprendizagem "batch"). Neste caso calcula-se o gradiente relativo ao peso individual w_j como:

$$\nabla E_j = \partial E / \partial w_j = -2\sum_{p=1}^{n} e(\underline{x}_p) x_{pj}$$

o que resulta nas regras de atualização de pesos (15) (absorvendo o termo constante 2 na taxa de aprendizagem η):

1.) $w_j^{(l+1)} = w_j^{(l)} + \eta e(\underline{x}_p) x_{pj}$, para $j = 0, \ldots, D$

2.) $w_j^{(l+1)} = w_j^{(l)} + \eta \sum_{p=1}^{n} e(\underline{x}_p) x_{pj}$, para $j = 0, \ldots, D$

A regra de adaptação de pesos apresentada acima chama-se *regra de delta, regra de adaline, regra de Widrow-Hoff* [Widrow and Hoff, 1960] ou *regra da média dos quadrados mínimos (LMS, least mean square rule)*, veja também [Hertz et al., 1991].

III.3 Perceptron Multi-Camada e Retropropagação de Erro

O estudo dos modelos do perceptron e do ADALINE trouxe-nos conhecimento sobre a natureza das funções calculadas pela rede neural artificial e os métodos de adaptação de pesos. Uma limitação natural desse tipo de rede é a linearidade das funções calculadas. Como já foi referido anteriormente a incapacidade do perceptron de calcular a classificação dos dois resultados da função do "Ou exclusivo" (XOR) ($f(0,0) = f(1,1) = 0$, $f(0,1) = f(1,0) = 1$) na Figura 11 levou a um certo desinteresse por parte da comunidade científica, devido especialmente a publicação de "Perceptrons" [Minsky and Papert, 1969].

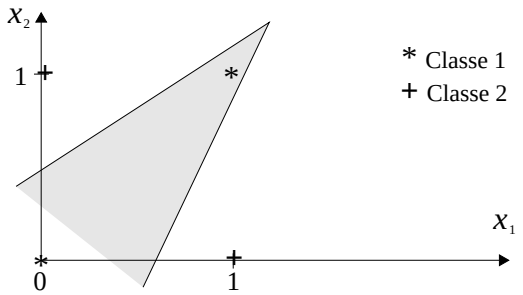

Figura 11- O problema XOR. Não existe uma separação linear entre as duas classes.

Era conhecido que a introdução de mais camadas no perceptron poderia resolver o problema de XOR. O que não se conhecia era um método para adaptar os pesos, especialmente em problemas de regressão e classificação de maior complexidade. [Rumelhart et al., 1986] popularizaram o algoritmo de retropropagação de erro. Apresenta-se em seguida o perceptron com uma camada escondida e várias saídas, treinado pela retropropagação de erro.

III.3.1 Arquitetura

Qualquer perceptron com pelo menos uma camada escondida (nem entrada, nem saída) é um perceptron multi-camada. Consideramos aqui o caso de uma única camada escondida. A generalização para mais que uma camada escondida é direta e a teoria aplica-se sem alteração,

veja a literatura. Um neurônio recebe várias entradas da camada anterior e calcula uma combinação linear (9) dessas variáveis. O resultado da combinação linear passa pela função de ativação, neste caso novamente a função sigmoidal (2). Usamos mais que uma saída. Isso significa que a saída é um vetor $\underline{d} = (d_1, \ldots, d_c)^T$ de c funções individuais d_i calculadas. Assim a rede realiza um mapeamento de um vetor multidimensional $\underline{x}$ para outro vetor multidimensional $\underline{d}$, i.e. a rede calcula $\underline{d}(\underline{x})$ com $\underline{d}(\underline{x}) = (d_1(x_1, \ldots, x_D), \ldots, d_c(x_1, \ldots, x_D))$.

A camada escondida tem um número H de neurônios. Usa-se eventualmente outra vez uma entrada constante de 1 também na camada escondida. Um peso entre a variável de entrada x_j e o neurônio com índice h na camada escondida chama-se w_{hj}. Todos os pesos w_{hj} podem ser juntados na matriz de pesos "entrada-escondida" W_h. Em analogia, existe um peso w_{ih} que liga o neurônio com índice h na camada escondida com a saída d_i. Todos os w_{ih} formam a matriz W_i.

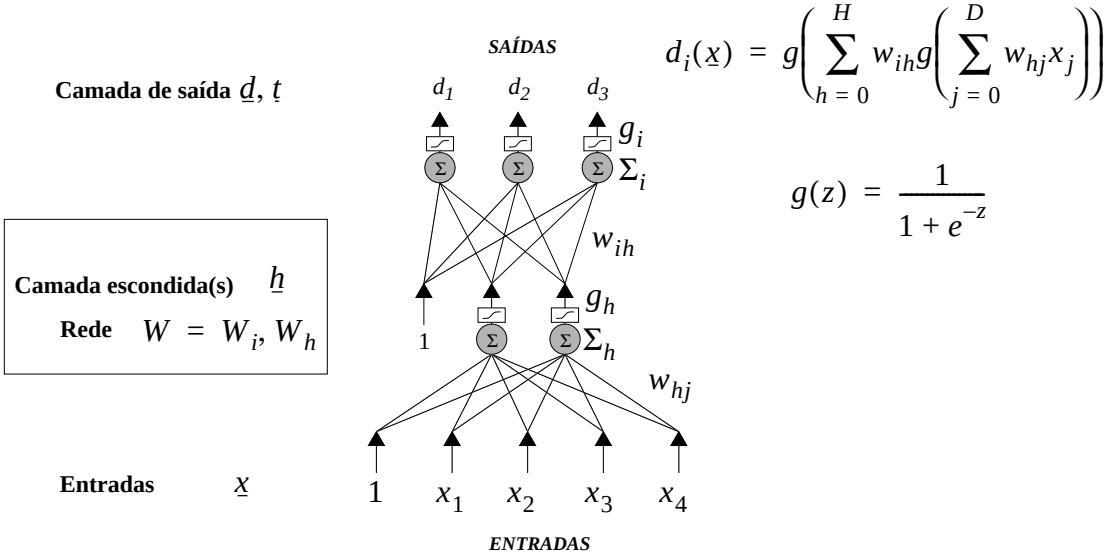

Figura 12- Perceptron multi-camada com uma camada escondida. Quatro variáveis de entrada, três variáveis de saída. Função de ativação sigmoidal.

O mapeamento realizado para uma função d_i de saída é:

Função do perceptron multi-camada: $d_i(\underline{x}) = g(\sum_{h=0}^{H} w_{ih} g(\sum_{j=0}^{D} w_{hj} x_j))$ (16)

A função (16) constitui um instrumento poderoso de realizar cálculos não lineares em muitas áreas de aplicação. Pode-se usar (16) para classificação de classes que não são linearmente separáveis, como no caso do XOR. Também para problemas de regressão o perceptron multi-camada é muito útil, por exemplo para realizar previsões. A área de controle automático e a identificação de plantas é outro exemplo.

Uma das grandes vantagens dessa técnica é que o perceptron multi-camada é um aproximador universal de funções. Isso significa que desde que os pesos sejam bem adaptados e a rede dispõe um número suficiente de neurônios escondidos, o cálculo desejado é atingido.

III.3.2 Adaptação dos pesos

Em analogia ao ADALINE tenta-se aprender os pesos da rede baseado numa função de erro entre o mapeamento realizado e desejado. Usa-se outra vez a técnica de descida de gradiente do Algoritmo 1. Temos que reformular o erro quadrático (10) porque aumentamos o número de saídas d_i para c. A diferença entre os valores desejados para o exemplo $\underline{x}_p$ e os valores calculados pela rede (10) agora virou diferença entre dois vetores. Os valores desejados é o vetor de alvo $\underline{t}_p = (t_{p1}, \ldots, t_{pc})^T$. O valor calculado pela rede é o vetor $\underline{d}(\underline{x}_p) = (d_1(\underline{x}_p), \ldots, d_c(\underline{x}_p))^T$. A escolha comum para a diferença entre o desejado e o calculado pela rede é a distância Euclidiana[1] quadrática entre os dois vetores:

$$\textit{Erro quadrático para um exemplo } \underline{x}_p\text{: } e^2(\underline{x}_p) = \left\|\underline{t}_p - \underline{d}(\underline{x}_p)\right\|^2 = \sum_{i=0}^{c} (t_{pi} - d_i(\underline{x}_p))^2 \qquad (17)$$

Finalmente o valor esperado do erro quadrático $E\{e^2(\underline{x}_p)\}$ pode ser estimado pela média dos erros quadráticos de todos os exemplos do conjunto de treino T.

$$\textit{Erro quadrático médio}\text{: } EQM = \frac{1}{n}\sum_{p=1}^{n} e^2(\underline{x}_p) = \frac{1}{n}\sum_{p=1}^{n}\sum_{i=0}^{c} (t_{pi} - d_i(\underline{x}_p))^2 \qquad (18)$$

Utilizamos outra vez a filosofia da descida de gradiente (15) e o Algoritmo 1 para adaptar os pesos. Desta vez porém, temos que adaptar pesos que dependem não-linearmente do gradiente de erro. Na adaptação dos pesos da camada escondida para a camada de saída $W_i = [w_{ih}]$ ainda podemos usar a regra de delta. Para a adaptação dos pesos da entrada para a camada escondida $W_h = [w_{hj}]$ temos que usar a regra de delta generalizada.

Vamos considerar unicamente a aprendizagem orientada a cada exemplo $\underline{x}_p$. Resta então calcular o gradiente em relação aos pesos da camada escondida para a camada de saída $\nabla E_{ih} = \partial E / \partial w_{ih}$ e o gradiente em relação aos pesos da entrada para a camada escondida $\nabla E_{hj} = \partial E / \partial w_{hj}$. Usamos as seguintes abreviações:

$$e_i = t_{pi} - d_i(\underline{x}_p)$$

$$d_i(\underline{x}_p) = g_i = g(\Sigma_i) = g\left(\sum_{h=0}^{H} w_{ih} g_h\right) = g\left(\sum_{h=0}^{H} w_{ih} g(\Sigma_h)\right) = g\left(\sum_{h=0}^{H} w_{ih} g\left(\sum_{j=0}^{D} w_{hj} x_j\right)\right)$$

---

1. Distância Euclidiana entre dois vetores $\underline{x} = (x_1, \ldots, x_{dim})^T$ e $\underline{y} = (y_1, \ldots, y_{dim})^T$ de dimensão dim é $\|\underline{x} - \underline{y}\| = \sqrt{\sum_{i=0}^{dim} (x_i - y_i)^2}$

Camada escondida para a camada de saída: Regra de delta:

$$\nabla E_{ih} = \partial E/\partial w_{ih} = \partial e^2/\partial w_{ih} = \partial(\sum_{i=1}^{c} e_i^2)/\partial w_{ih}$$

$$= \partial(\sum_{i=1}^{c}(t_i - g_i)^2)/\partial w_{ih} = \sum_{i=1}^{c}\partial(t_i - g_i)^2/\partial w_{ih} =$$

$$= -2\sum_{i=1}^{c}(t_i - g_i)(\partial g_i/\partial w_{ih}) = -2(t_i - g_i)(\partial g_i/\partial w_{ih}) = -2e_i\partial g_i/\partial w_{ih}$$

$$= -2e_i[g_i(1-g_i)(\partial\Sigma_i/\partial w_{ih})] = -2e_i[g_i(1-g_i)g_h] = -2\delta_i g_h$$

onde a quantidade $\delta_i = e_i g'_i = e_i(g_i(1-g_i))$ foi definida como o *delta* da camada de saída.

Entrada para camada escondida: Regra de delta generalizada:

$$\nabla E_{hj} = \partial E/\partial w_{hj} = \partial e^2/\partial w_{hj} = \partial(\sum_{i=1}^{c} e_i^2)/\partial w_{hj} = \partial(\sum_{i=1}^{c}(t_i - g_i)^2)/\partial w_{hj}$$

$$= \sum_{i=1}^{c}\partial(t_i - g_i)^2/\partial w_{hj} = -2\sum_{i=1}^{c}(t_i - g_i)\partial g_i/\partial w_{hj}$$

$$= -2\sum_{i=1}^{c} e_i(\partial g(\Sigma_i)/\partial w_{hj}) = -2\sum_{i=1}^{c} e_i g_i(1-g_i)(\partial\Sigma_i/\partial w_{hj})$$

$$= -2\sum_{i=1}^{c} e_i g_i(1-g_i)[\partial(\sum_{h=0}^{H} w_{ih}g_h)/\partial w_{hj}]$$

$$= -2\sum_{i=1}^{c} e_i g_i(1-g_i)[(\sum_{h=0}^{H} w_{ih}\partial g_h)/\partial w_{hj}]$$

$$= -2\sum_{i=1}^{c} e_i g_i(1-g_i)[g_h(1-g_h)(\partial\Sigma_h/\partial w_{hj})]$$

$$= -2\sum_{i=1}^{c} e_i g_i(1-g_i)[w_{ih}g_h(1-g_h)x_j]$$

$$= -2g_h(1-g_h)\sum_{i=1}^{c} e_i g_i(1-g_i)w_{ih}x_j = -2\delta_h x_j$$

onde a quantidade $\delta_h = g'_h\sum_{i=1}^{c}\delta_i x_j$ foi definida como o *delta* da camada escondida.

Fica clara que os deltas da camadas anteriores são propagadas “para trás”, assim justificando o nome “retropropagação de erro” (error backpropagation).

A rede neural artificial do perceptron multi-camada é um instrumento poderoso de realizar aproximações universais de funções a partir de um conjunto de dados de treino. Existem muitos campos de aplicação que permitem mapear o problema para uma rede e adaptar os pesos pelos dados de treino especificados.

Uma desvantagem do perceptron multi-camada é o tempo de treino extenso. Em um problema complexo pode-se levar várias semanas até se obter um conjunto de pesos adequados. A escolha da taxa de aprendizagem desempenha também um papel fundamental. Como já foi dito existe a possibilidade de que a função de erro fique presa em um mínimo local. Existem heurísticas para tentar resolver esse problema (fora do âmbito desse texto). *Paralisia* do treino é um fenômeno que acontece quando a magnitude dos pesos é grande. A derivada da função sigmoidal (2) fica muito pequena nesse caso, o que causa que a modificação dos pesos praticamente desaparece.

IV. REDES COM REALIMENTAÇÃO

Todos os modelos até agora apresentados (perceptron e adaline) são redes com propagação para frente. Isso significa que a informação é processada unidirecionalmente dentro da rede. Esse tipo de processamento supõe que não há comportamento dinâmico da rede, i.e. a ativação dos neurônios depende deterministicamente das entradas. Por outro lado, podemos conceber uma rede que realimenta os valores calculados como entradas novamente, o que provoca o cálculo de valores modificados nas saídas, que por sua vez novamente são realimentadas, compare Figura 4 e Figura 6. Certamente esse tipo de rede exibe um comportamento dinâmico dos seus valores, i.e. uma modificação dos valores ao longo do tempo, dado um único estímulo inicial. O conjunto de valores de saída atuais dos neurônios chama-se o *estado* da rede. Desejável é um comportamento estável. Isso significa que depois de um estímulo inicial da rede os valores de saída caminham para uma constante.

IV.1 Modelo de Hopfield

Como exemplo para apresentar os conceitos de redes com realimentação e comportamento dinâmico apresenta-se o modelo de Hopfield, denominado segundo um pesquisador com um grande impacto nesse tipo de arquitetura de rede [Hopfield, 1982]. Entradas x_j e saídas x_i do neurônio i são valores binários, sendo o valor 1 atribuído ao estado "ativo" e o valor -1 atribuído ao estado "inativo", i.e. $x_j \in \{-1, 1\}$ e $x_i \in \{-1, 1\}$. O neurônio é do modelo McCulloch-Pitts, com pesos w_{ij} entre neurônio i e neurônio j, veja Figura 2. Para facilitar o cálculo usa-se como função de ativação a função do sinal $\text{sgn}(z) = 1$ se $z \geq 0$ e $\text{sgn}(z) = -1$ se $z < 0$ e seja o limiar $\mu = 0$. Assim permite-se a formulação da

Regra de adaptação dinâmica da saída de neurônio i: $x_i = \text{sgn}(\sum_{j=0}^{D} w_{ij} x_j)$ (19)

Em princípio não se distinguiu entre variáveis de entrada x_j e variáveis de saída x_i, porque a saída de um neurônio pode servir como entrada realimentada do mesmo neurônio. Tendo uma rede de H neurônios o estado atual da rede caracteriza-se por um vetor de dimensão H com valores de 1 ou -1, e.g. $Estado = (-1, 1, 1, -1, 1, -1)^T = \underline{x}(t)$ para uma rede de 6 neurônios. O comportamento dinâmico é a mudança do estado de $\underline{x}(t)$ no momento t para o estado $\underline{x}(t + \Delta t)$ no momento $t + \Delta t$. A rede está estável se já não há mudança de estado, i.e. $\underline{x}(t + \Delta t) = \underline{x}(t)$ para todos $\Delta t \geq 0$.

Existem duas maneiras de atualizar o estado da rede. Na atualização *síncrona* existe uma entidade central que regula a transição de $\underline{x}(t)$ para $\underline{x}(t + \Delta t)$ ao mesmo tempo para todos os H neurônios. Na atualização *assíncrona* cada neurônio se atualiza independentemente dos outros neurônio (mas com a mesma frequência). A melhor maneira de realizar uma simulação é passar por todos os neurônios numa ordem aleatória e atualizar sequencialmente.

IV.2 Associatividade de Padrões na Rede de Hopfield

O objetivo da rede é memorizar n padrões $\underline{x}_p$ de um conjunto de padrões $T = \{\underline{x}_p\}_{p=1}^{n}$. A

rede tem que responder nas saídas com o estado $\underline{x}(t) = \underline{x}_p$ quando esse mesmo estado for apresentado inicialmente à rede, i.e. $\underline{x}(0) = \underline{x}_p$. A resposta deveria ser também o padrão memorizado, mesmo se o estímulo inicial somente for *parecido*[1] com um dos padrões memorizados, i.e. $\underline{x}(0) \approx \underline{x}_p$. Dessa capacidade de recuperar informação memorizada, mesmo com informação inicial incompleta, corrompida ou parcialmente errada deriva-se a ambição desse tipo de rede, nomeadamente de ser uma *memória associativa* que é capaz de imitar as capacidades do ser humano.

IV.3 Estabilidade e Aprendizagem de Pesos

Quais são as condições para que a rede seja estável? Se tivéssemos só um único padrão $\underline{x} = (x_1, \ldots, x_H)^T$ o estado calculado pela rede segundo (19) teria que ficar inalterado para cada neurônio $i = 1, \ldots, H$, i.e. $x_i(t + \Delta t) = x_i(t)$. Substituindo o peso w_{ij} pela expressão $x_i(t) \cdot x_j(t)$ na regra (19) $x_i(t + \Delta t) = \text{sgn}(\sum_{j=0}^{D} w_{ij} x_j(t))$, obtém-se

$$x_i(t + \Delta t) = \text{sgn}(\sum_{j=0}^{D} x_i(t) \cdot x_j(t) \cdot x_j(t)) = \text{sgn}(\sum_{j=0}^{D} x_i(t) \cdot 1)$$
$$= \text{sgn}(D \cdot x_i(t)) = x_i(t)$$

.

Normalizando pelo número dos neurônios presentes na rede (e ignorando o índice t do tempo) pode-se definir a:

$$\textit{Regra de aprendizagem para o peso } w_{ij} \textit{ para um padrão: } w_{ij} = \frac{1}{H} x_i x_j \qquad (20)$$

Uma rede treinada por essa regra vai recuperar o padrão ensinado $\underline{x}$ mesmo quando a metade (menos 1) das entradas esteja diferente do padrão. Um estímulo inicial parecido com $\underline{x}$ rapidamente *relaxa* para $\underline{x}$.

No caso de n padrões diferentes a extensão óbvia da regra (20) é a superposição do peso para todos os padrões. Observa-se que a regra de aprendizagem de Hebb (4) foi aplicada em (20), dando o nome à regra de aprendizagem para o peso w_{ij} para n padrões.

$$\textit{Regra de Hebb, Regra de Hebb Generalizada: } w_{ij} = \frac{1}{H} \sum_{p=0}^{n} x_{pi} x_{pj} \qquad (21)$$

onde x_{pi} é o estado de neurônio i para padrão p.

IV.4 Relaxação, Minimização de Energia

Na rede de Hopfield os pesos entre os neurônios são simétricos, i.e. $w_{ij} = w_{ji}$ e define-se um peso nulo da realimentação do próprio neurônio i.e. $w_{ii} = 0$. Com essas pré-condições a

---

1. A semelhança entre dois padrões binários $\underline{x}_p = (x_{p1}, \ldots, x_{pH})^T$ e $\underline{x}_q = (x_{q1}, \ldots, x_{qH})^T$ pode-se medir por exemplo em termos da distância de Hamming: $dist(\underline{x}_p, \underline{x}_q) = \sum_{i=0}^{H} [x_{pi}(1 - x_{qi}) + (1 - x_{pi}) x_{qi}]$

introdução de uma

$$\textit{Função de Energia: } H = -\frac{1}{2}\sum_{i=0}^{H}\sum_{j=0}^{H} w_{ij}x_i x_j \tag{22}$$

garante que a rede sempre relaxa para um estado estável. Isso acontece porque com os pesos simétricos a energia obrigatoriamente tem que diminuir em cada passo de relaxamento. Para uma prova desse fato veja [Hertz et al., 1991].

Vale a pena ainda ser mencionada que a rede relaxa para estados que não só são os padrões memorizados. Existem estados *reversos* e estados *espúrios*. Veja também [Hertz et al., 1991] para a definição desses efeitos colaterais não desejáveis na recuperação de informação na rede de Hopfield.

IV.5 Aplicação: Recuperação de Imagens

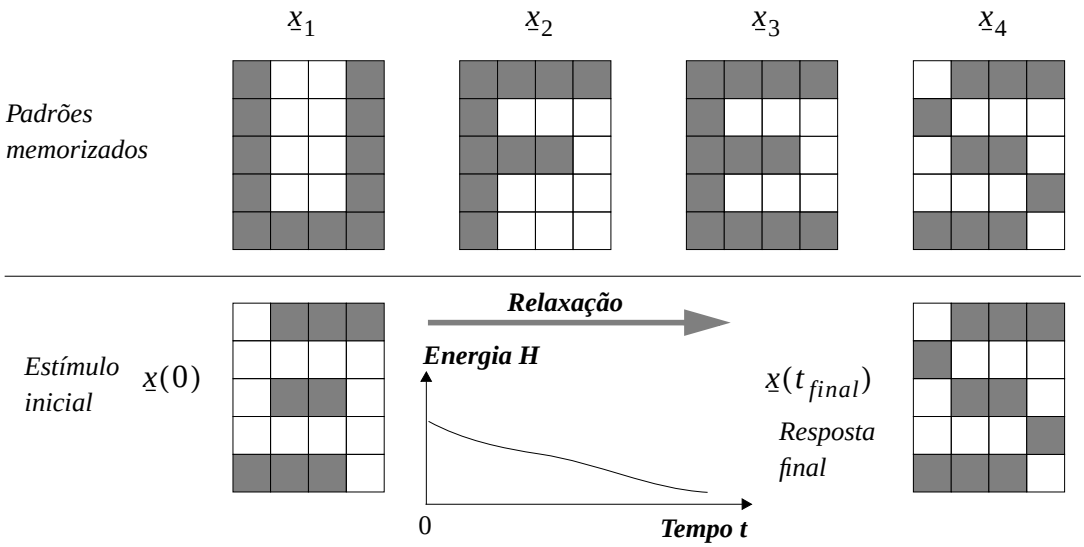

Figura 13 - Rede de Hopfield como memória de imagens. Quatro imagens binárias 4×5 são guardadas em vetores $\underline{x}_p = (x_{p1}, \ldots, x_{p20})^T$ de dimensão $20 = 4 \cdot 5$. Na fase de recuperação uma imagem corrompida é apresentada como estímulo inicial a rede que por relaxação chega no estado final da imagem memorizada mais parecida.

Uma aplicação que demostra a capacidade de associar informação incompleta é a memorização de imagens. Usam-se imagens binárias onde o valor -1 significa que o pixel na posição (a, b) da imagens seja branco e o valor 1 significa preto. Seja A o número de linhas da imagem e B o número de colunas, o que resulta em $A \cdot B$ pixels. Assim a imagem completa pode ser guardada em um vetor $\underline{x}$ de dimensão $H = A \cdot B$ simplesmente por linearização da imagem, introduzindo linha por linha no vetor $\underline{x}$.

Dessa maneira pode-se memorizar um conjunto de imagens diferentes $\{\underline{x}_p\}$, quatro no exemplo da Figura 13. A recuperação de uma imagem guardada se efetua pela apresentação de

uma imagem inicial à rede, i.e. inicializar o estado $\underline{x}(0)$ com a imagem inicial. Espera-se então que a rede relaxe para uma das imagens guardadas que seja a mais parecida com a imagem inicial. Assim, permite-se relembrar de uma imagem completa, mesmo que a imagem inicial seja apenas uma versão parcialmente errada ou incompleta da imagem memorizada.

V. REDES COMPETITIVAS

Consideremos o caso de aprendizagem não-supervisionada, em que a única informação disponível é o conjunto de exemplos $T = \{(\underline{x}_p)\}_{p=1}^{n}$. Todos os H neurônios i, $i = 1, \ldots, H$ recebem o mesmo estímulo $\underline{x}$. Entre a entrada x_j e o neurônio y_i existe um peso w_{ij}, i.e o vetor $\underline{w}_i$ liga a entrada $\underline{x}$ a saída y_i. Todos os neurônios formam a *camada competitiva*, veja Figura 14. Determina-se um *vencedor* dentro dessa camada competitiva. O neurônio vencedor i^* é o único que tem o direito de emitir um sinal de saída que seja 1, i.e. $y_{i^*} = 1$ (vencedor-pega-tudo ou "winner takes all"). Todos os outros neurônio ficam inativos, i.e. $y_i = 0$, para $i^* \neq i$.

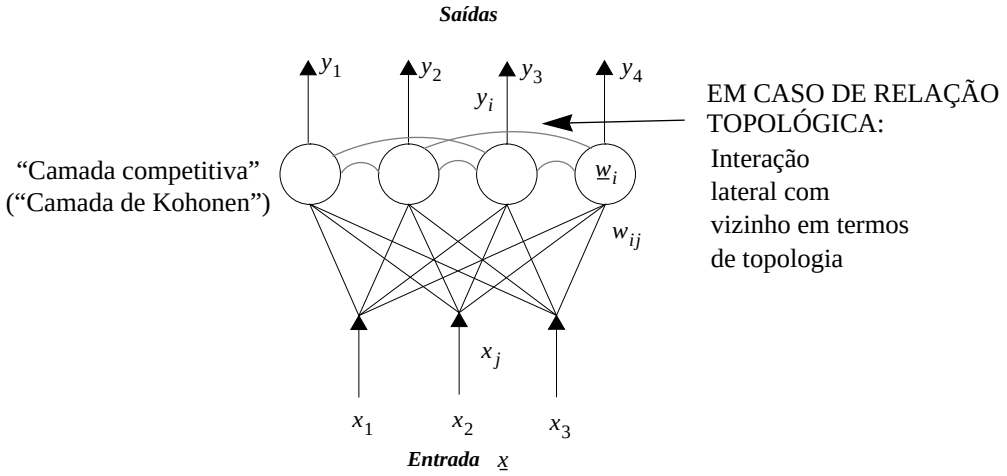

Figura 14 - Rede Competitiva. A única saída que emite um sinal $y_{i^*} = 1$ é aquela do neurônio i^*, o vencedor, que tem a maior semelhança com o estímulo $\underline{x}$.

V.1 Determinação do Vencedor

Basicamente usam-se duas possibilidades de determinar o vencedor dentro da camada competitiva. No primeiro caso normalizam-se ambos o sinal de entrada e os pesos para um comprimento de 1, i.e. $\underline{x} \leftarrow \underline{x} / \|\underline{x}\|$ e $\underline{w}_i \leftarrow \underline{w}_i / \|\underline{w}_i\|$ para que seja $\|\underline{x}\| = 1$ e $\|\underline{w}_i\| = 1$. Nesse caso o vencedor determina-se pelo maior produto interno:

$$\textit{Vencedor (caso vetores normalizados):}\ \underline{w}_{i^*}^T \underline{x} > \underline{w}_i^T \underline{x} \tag{23}$$

ou equivalentemente pelo menor ângulo θ_i entre peso e sinal de entrada, veja Figura 15 a.

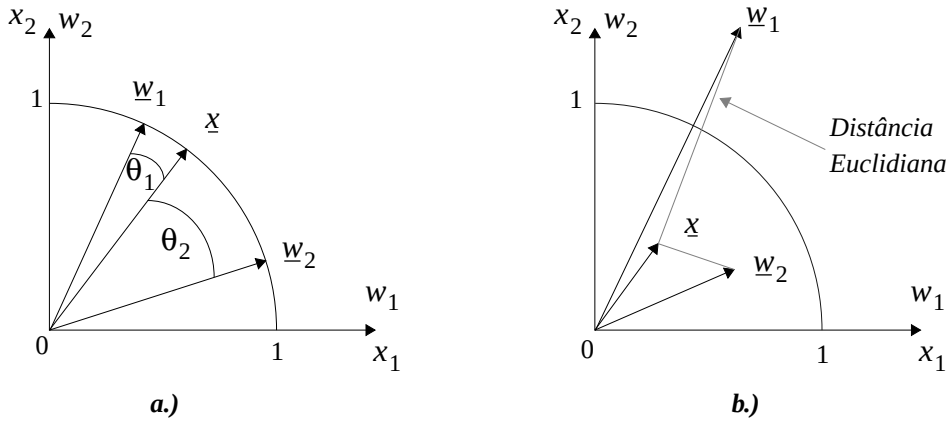

Figura 15 - Determinação do vencedor (da semelhança entre o estímulo $\underline{x}$ e o peso $\underline{w}_i$). Caso de duas dimensões ($D = 2$). A semelhança pode ser o produto interno $\underline{w}_i^T \underline{x}$ em caso de comprimento normalizado dos vetores, i.e. $\|\underline{w}_i\| = 1$ e $\|\underline{x}\| = 1$ ou pode ser a distância Euclidiana $\|\underline{w}_i - \underline{x}\|$ caso contrário.

A segunda opção para determinar a maior semelhança entre a entrada $\underline{x}$ e o peso $\underline{w}_i$ é medir a distância Euclidiana entre os dois vetores $\|\underline{x} - \underline{w}_i\| = \sqrt{\sum_{j=0}^{D} (x_j - w_{ij})^2}$. No caso em que os dois vetores não estejam normalizados como na Figura 15 b, essa convenção toma melhor em consideração o comprimento dos vetores e não o ângulo entre eles.

V.2 Adaptação dos Pesos

Usando o método iterativo de adaptação de pesos (3), os pesos da rede se modificam exclusivamente baseado na estrutura dos dados de treino T. A expectativa do comportamento de uma rede desse tipo é que estímulos $\underline{x}_p$ e $\underline{x}_q$ que são parecidos provocam a resposta do mesmo vencedor. Considerando esse fato é óbvio que os pesos deveriam movimentar-se em direção aos estímulos, i.e. na direção do vetor $\underline{w}_{i^*}$ pertencendo ao neurônio vencedor i^*, i.e. $\Delta \underline{w}_{i^*} = \underline{x} - \underline{w}_{i^*}$. Naturalmente temos que ponderar o passo de aproximação pela taxa de aprendizagem η que controla a velocidade do processo de aprendizagem. A adaptação pode ser feita outra vez independentemente para cada componente j do vetor $\underline{x}$, $j = 1, \ldots, D$. Como as saídas dos neurônios não-vencedoras são nulos para $i \neq i^*$, pode-se juntar a regra para todos os pesos, obtendo (6):

$$\textit{Regra de aprendizagem competitiva: } w_{ij}^{(l+1)} = w_{ij}^{(l)} + \eta y_{pi}(x_{pj} - w_{ij}^{(l)}) \qquad (24)$$

onde y_{pi} é a saída de neurônio i para a entrada $\underline{x}_p$ e x_{pj} é a j-ésima componente da entrada $\underline{x}_p$.

V.3 O Mapa de Preservação Topológica de Kohonen

Até agora não existia nenhuma relação topológica entre neurônios da mesma camada. Por exemplo, num perceptron multi-camada o valor da função de ativação de um neurônio na camada escondida não tem nenhuma influência sobre o valor dessa função do vizinho dele na mesma camada, ou sobre o peso na fase de aprendizagem. Vamos considerar agora redes que na fase da adaptação dos pesos exercem um efeito colateral sobre os vizinhos na camada competitiva. Na Figura 14 essa interação está implementada pelas ligações laterais entre vizinhos imediatos e mais afastados da camada competitiva. Muitas contribuições foram feitas por T. Kohonen nessa área [Kohonen, 1972], [Kohonen, 1990]. Por isso as vezes a camada competitiva topológica com interação é chamada camada de Kohonen. A motivação biológica para essa aproximação é o fato que também no cérebro pode-se observar que estímulos sensoriais externos parecidos muitas vezes excitam as mesmas áreas do cérebro, especialmente na formação das capacidades de motricidade e sensibilidade.

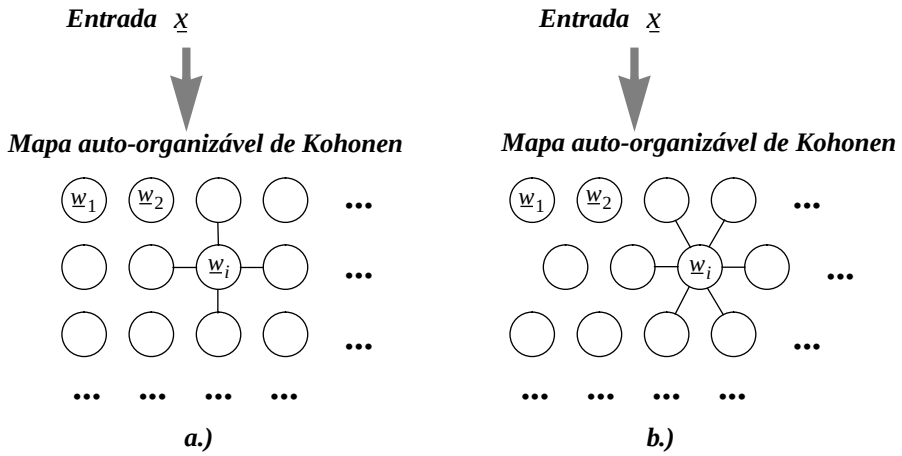

Figura 16- Mapa auto-organizável de Kohonen de duas dimensões: a.) Topologia retangular. Cada neurônio tem quatro vizinhos imediatos. b.) Topologia hexagonal. Cada neurônio tem seis vizinhos imediatos.

Na Figura 16 mostra-se uma organização topológica dos neurônios (que todos fazem parte da camada competitiva) em uma estrutura 2-D. Cada neurônio tem 4 ou 6 vizinhos imediatos, conforme a organização retangular ou hexagonal das ligações entre os membros da camada. Cada neurônio recebe o mesmo estímulo externo $\underline{x}$ e o vencedor é determinado, conforme a convenção ilustrada na Figura 15. A regra (24) é modificada de maneira que se modificam *todos* os pesos, dependendo da função de relação topológica (função de vizinhança) $h(i^*, i)$ entre o vencedor i^* e os seus vizinhos i, sendo $h(i^*, i^*) = 1$.

Regra de aprendizagem competitiva na camada de Kohonen:

$$w_{ij}^{(l+1)} = w_{ij}^{(l)} + \eta h(i^*, i)(x_{pj} - w_{ij}^{(l)}) \qquad (25)$$

A função $h(.,.)$ deve ter um valor máximo no centro, i.e. no neurônio vencedor e ter um comportamento decrescente com distância crescente do centro. Uma função que satisfaz essas condições é a função Gaussiana, veja Figura 17. A introdução de $h(.,.)$ provoca que, além do vencedor, os pesos dos vizinhos do vencedor também se atraem em direção do estímulo, mas

em quantidade menor. Se $h(.,.)$ for outra função com valores menor que zero na vizinhança teríamos ligações inibitórias que em vez de atrair os pesos para o estímulo os afastariam do estímulo.

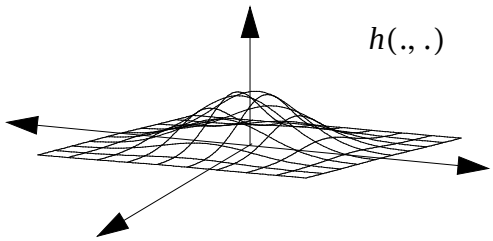

Figura 17 - Gaussiana em duas dimensões.

VI. CONCLUSÕES

Neste documento deu-se uma breve introdução à área das redes neurais artificiais. A ambição desse texto é a familiarização com os conceitos básicos da neurocomputação, neurônio artificial, topologia da rede, paradigmas de aprendizagem e regras para a adaptação dos pesos. Ao mesmo tempo tentou-se esclarecer as capacidades e limitações de uma rede neural artificial. Desde que seja possível mapear o problema em consideração para a arquitetura de uma rede, pode-se esperar a resolução de problemas cognitivas interessantes. Ao mesmo tempo tem que ficar claro que a classe de problemas resolúveis por uma rede neural artificial é limitada. Basicamente redes neurais artificiais são aproximadores universais de funções. Em outras áreas científicas existem propostas para a obtenção da função desejada, como por exemplo na estatística. A vantagem das redes é a aprendizagem por um conjunto de treino, junto com um caráter universal da arquitetura da rede. Isso significa que o projetista de uma rede não tem que se preocupar muito com parâmetros internos da rede. Por exemplo em um perceptron com uma camada escondida praticamente o único grau de liberdade é o número dos neurônios nessa camada escondida. Mesmo assim esse tipo de rede possui um potencial grande para ser usado em várias áreas de aplicação. Pode-se concluir que o caminho para um sistema artificial realmente inteligente provavelmente tenha que ter incorporado alguns dos princípios de redes neurais artificiais.

BIBLIOGRAFIA

[Anderson, 1995]
Anderson, J. A., *An Introduction to Neural Networks*, Cambridge, MA: The MIT Press, 1995

[Anderson and Rosenfeld, 1988]
Anderson, J. A. and Rosenfeld E., eds., *Neurocomputing: Foundations of Research*, Cambridge, MIT Press, 1988.

[Bishop, 1995]
Bishop, C. M., *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press, 1995

[Fausett, 1994]
Fausett, L., *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1994

[Haykin, 1994]
Haykin, S., *Neural Networks, a Comprehensive Foundation*, Macmillan, New York, NY, 1994

[Hebb, 1949]
Hebb, D. O., *The Organization of Behavior.* Wiley, New York, 1949, Partially reprinted in [Anderson and Rosenfeld, 1988]

[Hertz et al., 1991]
Hertz, J., Krogh, A., and Palmer, R., *Introduction to the Theory of Neural Computation*. Addison-Wesley: Redwood City, California, 1991

[Hinton, 1992]
Hinton, G.E., "How Neural Networks Learn from Experience", Scientific American, 267 (September), 144-151, 1992

[Hopfield, 1982]
Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," Proc. of the National Academy of Sciences, USA, vol. 79, pp. 2554-2558, 1982, Reprinted in [Anderson and Rosenfeld, 1988]

[Kohonen, 1972]
Kohonen, T., "Correlation Matrix Memories," IEEE C-21 (4), pp. 353-359, 1972

[Kohonen, 1990]
Kohonen, T., "The self-organizing map," *Proc. of the IEEE*, vol. 78, no. 9, 1990.

[Masters, 1994]
Masters, T., *Practical Neural Network Recipes in C++*, Academic Press, 1994

[McCulloch and Pitts, 1943]
McCulloch, W. S., and Pitts, W., "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943, Reprinted in [Anderson and Rosenfeld, 1988]

[Minsky and Papert, 1969]
Minsky, M., Papert S., *Perceptrons*, The MIT Press, Cambridge, MA, 1969.

[Ripley, 1996]
Ripley, B.D., *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press, 1996

[Rosenblatt, 1958]
Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386-408, 1958.

[Rumelhart et al., 1986]
Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning internal representations by error propagation," in Rumelhart, D. E., and McClelland, J. L. (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition I*, pp. 318-362, MIT Press, Cambridge MA, 1986.

[Widrow and Hoff, 1960]
B. Widrow, and M. E. Hoff, "Adaptive switching circuits," in *1960 WESCON Convention Record*, New York, 1960.