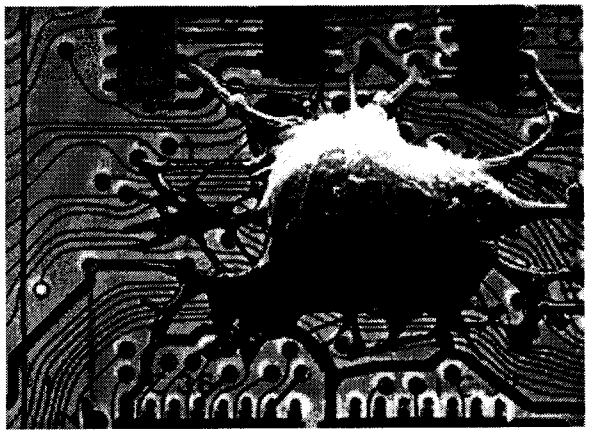
Notas Técnicas

CBPF-NT-006/97

Outubro 1997

Introdução à Redes Neurais

Deolinda M.P. Aguieiras de Lima e Nilton Alves Jr.



[1]

Deolinda M. P. Aguieiras de Lima
deolinda@cat.cbpf.br

Nilton Alves Jr.
naj@cat.cbpf.br

Resumo

Este trabalho tem como finalidade servir de primeira leitura para aqueles alunos que desejam iniciar-se no estudo de Redes Neurais. É uma leitura rápida e simples, onde são abordadas as bases deste ramo da Inteligência Artificial, que permite várias aplicações científicas.

Este texto está dividido em partes que se sucedem de maneira dependente e contínua. Sendo assim, não é recomendado para iniciantes o salto para sessões posteriores, sob pena de perda de entendimento e de clareza.

Na primeira sessão, é feito um paralelo entre o cérebro humano e o computador, aqui considerado cérebro artificial. São abordadas as diferenças e similaridades entre O Computador e o Homem.

Na segunda sessão, são apresentados os Conceitos Básicos do neurônio artificial, do aprendizado, do perceptron e suas características e limitações.

Na terceira sessão, analisamos o Perceptron de Múltiplas Camadas inclusive um problema típico como o XOR.

Na quarta sessão, são apresentadas as principais Técnicas de Aprendizado e por fim, na quinta e última sessão, algumas Aplicações Científicas.

Índice

Resumo 1
Índice 2
1. O Computador e o Homem 3
1.1. A Estrutura do Cérebro 3
1.2. A Organização do Cérebro 1
1.3. Aprendizado em Sistemas Biológicos e em Máquinas 4
1.4. Diferenças Entre o Computador e o Homem 5
2. Conceitos Básicos 5
2.1. Neurônio 6
2.2. Aprendizado Supervisionado 8
2.3.Perceptron 10
2.4.Visão Vetorial do Perceptron 10
2.5.As Limitações dos Perceptrons 11
3. Múltiplas Camadas 12
3.1. O Perceptron com Múltiplas Camadas 12
3.2. O Novo Modelo de Neurônio 14
3.3. Resolvendo o Problema do XOR 14
4. Técnicas de Aprendizado 16
4.1. Back Propagation 17
4.2. Error - Correction Learning[5] 18
5. Redes Hopfield 18
5.1. Definição 19
5.2. Aplicações da Rede Hopfield 20
Referências 21

Introdução à Redes Neurais

1. O Computador e o Homem

A principal diferença entre o cérebro humano e o computador é que computadores só fazem coisas lógicas, numéricas, tais como lidar com bancos de dados, fazer contas, gráficos e etc, enquanto que o cérebro humano, pode, além disso, fazer coisas mais abstratas, tais como reconhecer qualquer objeto, controlar nossos movimentos, aprender outras línguas e elaborar estratégias. Estas tarefas, em princípio não lógicas, também podem ser feitas por computadores, porém para isso, são necessárias técnicas e algorítmos complexos.

Uma maneira simples de caracterizar bem a diferença entre o computador e o Homem seria comparar o computador, que é uma máquina serial, com o nosso cérebro, que é altamente paralelo e possui como característica principal a capacidade de aprender coisas.

1.1. A Estrutura do Cérebro

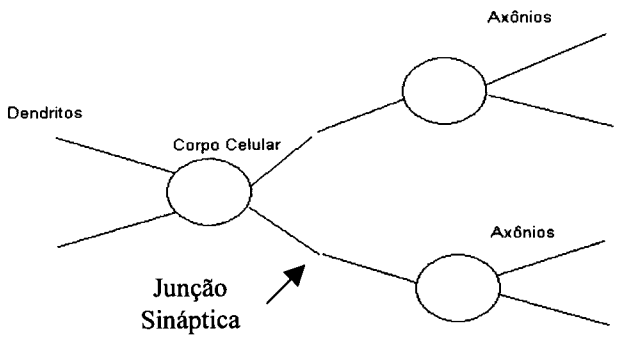
No estudo do cérebro humano, o baixo conhecimento é o maior problema a ser enfrentado. Existem perguntas simples que ainda não têm explicação como por exemplo "o que é minha mente?" ou "como eu penso?". Podemos então assumir que temos uma máquina muito complicada de origem desconhecida e que nossa meta é entender como ela funciona [2].

Sabemos que o cérebro humano é composto de aproximadamente 10^{10} (dez bilhões) unidades básicas, chamadas de neurônios, e cada neurônio é conectado a 10^4 (dez mil) outros neurônios.

Os neurônios podem ser separados em dois tipos: neurônios locais e neurônios que comunicam o cérebro com outras partes do corpo, como os músculos ou órgãos sensoriais.

Os neurônios são compostos de três partes: dendritos, axônio e corpo celular ou soma. Os dendritos fazem o transporte de informações vindas de outros neurônios para dentro da célula. As informações são somadas no corpo celular, processando assim uma outra informação, que sai da célula através do axônio. Então

esse axônio e outros axônios de outros neurônios vão se comunicar com os dendritos de uma determinada célula, montando uma rede. Essa comunicação recebe o nome de sinapse nervosa ou junção sináptica, que é a unidade funcional básica para a construção de circuitos neurais biológicos e envolve a junção das membranas plasmáticas de dois neurônios de modo a formar uma junção pontual orientada do neurônio pré-sináptico para o pós-sináptico. O tamanho de uma junção sináptica é menor do que 1mm.



1.2. A Organização do Cérebro

O cérebro é organizado em várias regiões diferentes, sendo cada região responsável por diferentes funções.

As maiores partes do cérebro são os hemisférios cerebrais, que ocupam a parte mais interior do crânio.

No córtex cerebral, a parte mais complexa do cérebro, as células nervosas são extremamente densas e portanto são capazes de permitir uma boa interconectividade. Por isso, nesta região ocorrem as mais altas funções e as mais altas ordens, ou seja, é o núcleo de nossa inteligência [7].

1.3. Aprendizado em Sistemas Biológicos e em Máquinas

Assumindo que a habilidade de aprender não é unicamente para o mundo biológico, podemos afirmar que modelos de redes neurais também são capazes de aprender.

O aprendizado em sistemas biológicos ocorre quando são feitas modificações na junção sináptica, ou seja, entre uma célula e outra, enquanto o

aprendizado em modelos de redes neurais ocorre quando é obtida uma resposta incorreta e forçamos uma resposta certa.

Um bom exemplo para entendermos como um computador pode aprender é o computador MENACE - *Matchbox Educable Noughts And Crosses Engine*. A primeira vez que um determinado jogo é executado, a máquina faz jogadas totalmente aleatórias e quando o jogo termina estas jogadas são armazenadas na máquina, para que na próxima vez em que este jogo seja executado o computador repita somente as jogadas que obtiveram sucesso, jogando assim cada vez melhor.

Este tipo de aprendizado é lento e continua até que a probabilidade de ganhar um jogo ou de dar uma resposta certa seja infinitamente maior que a probabilidade de perder um jogo ou de dar uma resposta errada.

1.4. Diferenças Entre o Computador e o Homem

A principal diferença entre o computador e o homem é a performance. O cérebro humano desenvolve tarefas altamente complexas com alta velocidade, devido ao paralelismo enquanto isso, computadores são lentos e seqüenciais.

Por exemplo, a visão, que é uma função altamente complexa e vai além de adicionar uma determinada quantidade de números, é manipulada sem esforços. Enquanto em um computador uma simples conta de divisão pode ocasionar erros.

Outra diferença notável é que no cérebro humano vários neurônios estão trabalhando em um determinado instante para uma determinada função. Se um neurônio tiver algum problema não será tão drástico quanto trabalhando serialmente, pois se em algum momento houver qualquer problema, a partir deste instante tudo terá problemas.

Além disso, todos os dias células nervosas morrem e mesmo assim nosso cérebro continua funcionando normalmente como se nada tivesse acontecido. Em redes neurais artificiais, neurônios não morrem a cada dia; caso tal fato ocorresse, com o passar do tempo, teríamos uma rede completamente obsoleta.

2. Conceitos Básicos

Nesta sessão, as noções básicas dos principais conceitos necessários para o desenvolvimento do trabalho são apresentadas com alguma profundidade.

Abordaremos os conceitos de: Neurônio, Aprendizado Supervisionado, Perceptron, Visão Vetorial e suas Limitações.

2.1. Neurônio

Neurônios biológicos são células nervosas que recebem e transformam estímulos em outros estímulos, enquanto neurônios artificiais transformam várias entradas em uma única saída.

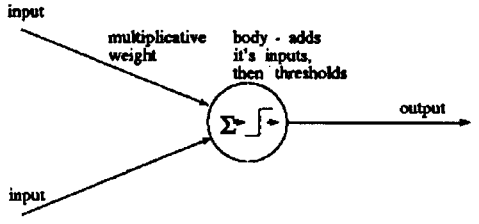
A função básica de um neurônio é adicionar entradas e produzir uma saída. Se o somatório das entradas for maior que um certo valor - valor limite ou *treshold* - haverá uma saída, caso o valor da soma seja inferior à este valor limite, não será produzida uma saída ou terá saída zero.

A estrutura de um neurônio artificial também será como nos neurônios biológicos. Informações chegam através dos dendritos, são somadas no corpo celular, processando assim uma nova informação, cuja saída será efetuada pelos axônios.

Podemos dizer que um neurônio só terá duas saídas: ligado ou desligado, dependentes única e exclusivamente das entradas. Algumas combinações de entradas serão capazes de disparar saídas.

Para melhorar a eficiência das junções sinápticas vamos colocar fatores multiplicativos em cada entrada que chamaremos pesos. Uma junção sináptica mais eficiente terá pesos maiores, pois assim estamos praticamente garantindo que haverá uma saída, ou seja, é mais provável que sinapses com pesos menores não atinjam o valor limite.

Introduzimos, então, uma diferença entre o neurônio biológico e o neurônio artificial. O neurônio artificial passa a funcionar da seguinte forma: as entradas chegam pelos dendritos, são multiplicadas pelos seus pesos e somadas dentro do corpo celular. Se o resultado do somatório for maior que um determinado valor limite, haverá uma saída, caso contrário não haverá uma saída, ou melhor, podemos dizer que a saída estará desligada. [4]



Podemos então formalizar este processo matematicamente:

entrada total = entrada 1 * peso da entrada 1 +

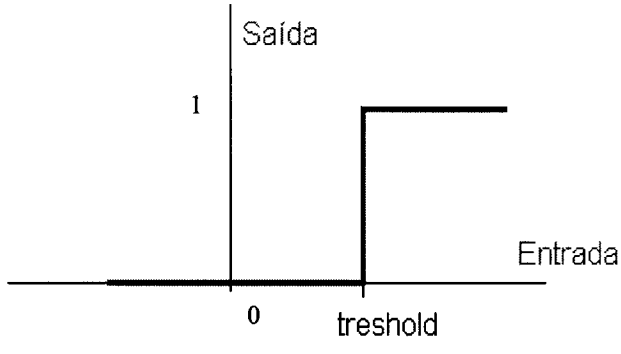
entrada 2 * peso da entrada 2 + ... +

entrada n * peso da entrada n

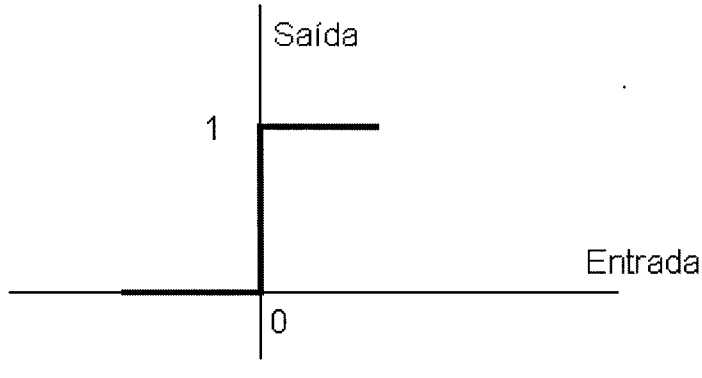
$= e1p1 + e2p2 + e3p3 + \ldots + e_np_n$

$= \sum e_np_n$

Então comparamos a entrada total com o valor limite: se for maior a saída terá valor 1; se for menor, a saída será 0. As saídas podem ser graficamente representadas através da função degrau esquematizada abaixo.



Poderíamos também, comparar a entrada total com o valor limite, fazendo mais uma conta: subtrairíamos o valor limite da entrada total. Caso o resultado fosse positivo a saída será 1; caso contrário, a saída será igual a 0.



Acima, já foram apresentadas duas maneiras para determinar a saída do neurônio. Na primeira temos a comparação com o valor limite, enquanto que na segunda maneira temos a comparação com zero (graficamente continuamos com a mesma

função, só que agora a descontinuidade passa a ser na origem). Notamos que em ambas as maneiras existe uma comparação.

Podemos mostrar a segunda maneira de uma outra forma: introduziríamos mais uma linha de entrada onde o valor de entrada fosse sempre 1. Assim, esta linha permaneceria sempre ligada e o valor do seu peso seria o valor limite com o sinal trocado. Então faríamos apenas um somatório, pois uma subtração pode ser considerada uma soma com sinal trocado. Isto é conhecido como neurônio de *biasing*, e o valor limite é conhecido como *bias* do neurônio ou *offset*.

Na segunda maneira temos:

entrada total = $\sum e_n p_n$

Chamaremos f(x) de função degrau e x de entrada total;

$y = f(x)$ $\quad f(x) = 1, \text{ se } x > 0$

$\quad f(x) = 0, \text{ se } x \leq 0$

Para finalizarmos esta seção, lembraremos que a nossa saída não precisa ser necessariamente zero ou um, mas tem que ser binária, isto é, ela tem que ter como saída somente dois valores diferentes.

2.2. Aprendizado Supervisionado

Precisamos de uma técnica para o aprendizado do neurônio, pois quando conectarmos os neurônios entre si, eles podem produzir qualquer saída, inclusive aquela errada. Assim sendo, teremos que treiná-los para que possam desempenhar suas funções corretamente. Começaremos por uma regra bem simples, pois terá a simplicidade do modelo de neurônio. Buscaremos então inspiração no sistema neural biológico.

Quando uma criança começa a falar, geralmente troca alguns sons pronunciando assim as palavras de forma incorreta. A criança passa, então, por um período de aprendizado, até falar de maneira correta. Este aprendizado é da seguinte forma: todas as vezes que a criança fala uma palavra errada alguém repete a mesma palavra de forma correta para que ela aprenda os sons.

O mesmo processo ocorre para atores. Eles repassam seus textos várias vezes, diminuindo assim a probabilidades da ocorrência de erros.

Podemos usar os exemplos acima para explicar o que é o aprendizado supervisionado. Os dois exemplos possuem características comuns, pois a resposta correta é enfatizada. Devemos passar esta idéia para a nossa rede neural artificial, ou seja, temos que estimular as situações que queremos que se repitam e reprimir as situações que não são as esperadas.

Vamos considerar um pequeno exemplo de reconhecimento de padrões. Temos dois grupos distintos e queremos reconhecê-los. Suponhamos várias letras A's e várias letras B's, escritas de maneiras diferentes. Assim sendo para um deles teremos como saída 1 e para o outro a saída será 0. Determinaremos a saída 1 quando a letra A é mostrada e 0 quando a letra B é mostrada. Na realidade queremos distinguir a letra A da letra B. Se quando um A for mostrado, tivermos como saída 0, precisaremos reduzir as chances para que tal situação não ocorra novamente e para isso é que são utilizadas as técnicas de aprendizado.

Aplicando a idéia de estudo supervisionado à nossa rede, teríamos como primeira etapa a colocação dos pesos aleatoriamente nas entradas e a apresentação de uma letra. Como segunda etapa, faríamos o somatório de todas as nossas entradas multiplicadas pelos seus respectivos pesos, e caso a soma ultrapassasse o nosso valor limite a saída seria 1, caso contrário, seria 0. Se a resposta estiver correta não precisaríamos fazer qualquer alteração, mas se o neurônio produzisse saída 0 quando um A for mostrado, logo precisaríamos alterar os pesos. Aumentando os pesos estaremos aumentando a nossa soma ponderada, e assim na próxima vez nossa soma ultrapassará o valor limite e produzirá saída 1, que é o valor esperado. Se for mostrado um B, a resposta esperada será 0. Caso a resposta seja 0, não será necessário fazer qualquer alteração. Se a resposta dada for 1, teremos novamente que fazer alterações na nossa soma ponderada. Mas nesta situação as alterações serão feitas com o objetivo de diminuir o valor da soma ponderada. Como a resposta dada foi 1, a soma ponderada ultrapassou o valor limite. Assim, para que a resposta seja 0, vamos diminuir os pesos para que o valor limite não seja mais ultrapassado. Logo, a resposta será a esperada.

Concluímos que a nossa rede de neurônios aprenderá alguma coisa através da alteração de seus pesos, ou seja, aumentaremos os pesos quando esperarmos a saída ativada (valor 1) e diminuiremos quando esperarmos a saída desativada (valor 0).

Quando alterarmos os pesos, aumentando ou diminuindo, só será necessário a alteração dos pesos cujas entradas estejam ligadas. As entradas desativadas não irão contribuir para a nossa soma ponderada, ou seja, caso as entradas que estão desativadas tenham seus pesos alterados isto não implicará em qualquer mudança.

Então podemos perceber que esta técnica de aprendizado tem como idéia básica uma saída esperada e caso essa saída não ocorra teremos que fazer modificações nos pesos. Esta técnica funciona como se a todo momento houvesse um professor dizendo qual é a resposta esperada. A essa técnica damos o nome de Aprendizado Supervisionado.

2.3.Perceptron

Um perceptron é basicamente um neurônio artificial com algumas características similares ao sistema nervoso central, mas também podemos descrever um perceptron como uma rede bem simples.

Perceptron é apenas um nome, pois de forma prática ele é um neurônio, já que existe uma entrada, uma soma ponderada e também uma saída.

O nome perceptron vem de percepção, tendo assim sua origem no sistema nervoso central pois é o nosso cérebro que comanda a nossa percepção [3].

2.4.Visão Vetorial do Perceptron

Vamos nesta seção tratar o perceptron de uma forma analítica, considerando-o como uma entidade matemática. Apresentaremos então, uma visão vetorial que será útil mais adiante.

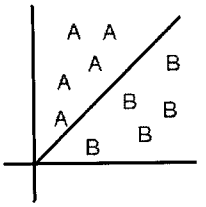
Vamos chamar de vetor E o vetor que conterá todas as nossas entradas. Por isto, terá dimensão n e podemos escrevê-lo como E = (e0,e1,e2,e3,...,en).

Da mesma forma como fizemos para o vetor E, faremos para o vetor P, que será o conjunto de todos os pesos. Assim, ele também terá dimensão n.

Teremos então a nossa soma ponderada como:

$$\Sigma = E * P$$

No exemplo anterior, o perceptron reconhecia a letra A e a letra B (também pode ser chamado de reconhecedor dos padrões A e B). Assim, podemos dizer que pela forma vetorial a solução para este problema seria como se estivéssemos traçando um vetor, isto é, dividindo o nosso plano vetorial em dois. Em um semiplano ficariam todos os A's e no outro ficariam todos os B's. Logo, a solução que estamos esperando para o nosso problema é exatamente este vetor que dividirá o plano em dois semiplanos.



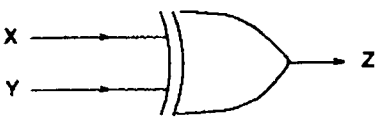
2.5.As Limitações dos Perceptrons

Notaremos nesta seção que existem limitações em nosso modelo de perceptron.

Sabemos que o nosso modelo de perceptron vai ter que traçar uma linha reta separando as devidas classes (semiplanos) que no nosso exemplo, onde buscávamos reconhecer os padrões A e B, serão a classe A e a classe B.

Entretanto, existem vários casos no qual nosso modelo de perceptron não consegue separar as classes. Um exemplo clássico é o problema do XOR, ou ou-exclusivo [4].

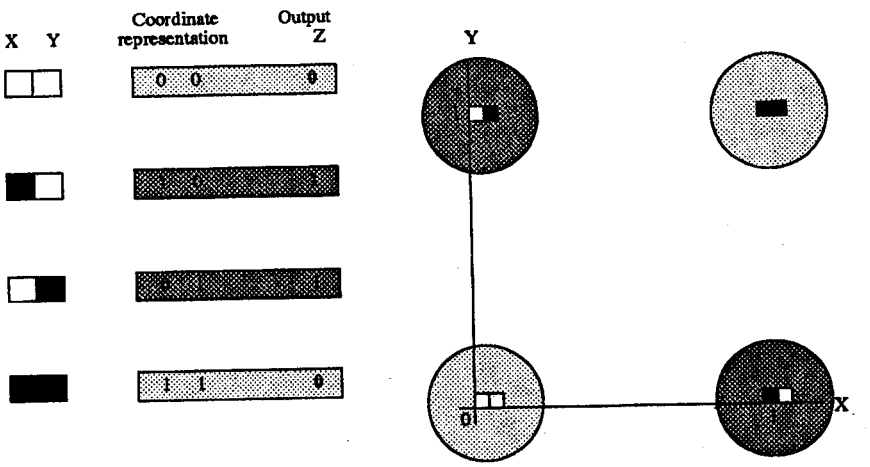
O problema do XOR será representado da seguinte forma: teremos duas entradas e uma saída. Caso tenhamos como entradas 0 e 0, teremos como saída 0; caso tenhamos 1 e 1, teremos como saída 0 e se nossas entradas forem 0 e 1, teremos como saída 1.



Na verdade, queremos que o perceptron obtenha como saída 1, se a nossa entrada x for 1 e a nossa entrada y for 0 ou se a nossa entrada x for 0 e a nossa entrada y for 1, e caso ocorra qualquer outra situação teremos como saída 0.

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Iremos então representar este problema graficamente no plano cartesiano. Quando as nossas entradas forem 0 e 0, marcaremos a origem do plano. Quando as entradas forem x = 0 e y = 1, marcaremos o ponto (0,1), quando forem x = 1 e y = 0, marcaremos o ponto (1,0) e finalmente, quando as entradas forem 1 e 1, marcaremos o ponto (1,1). Podemos então visualizar tal situação através do gráfico abaixo:



Podemos então perceber pelo gráfico acima que não poderemos de maneira nenhuma traçar uma reta que separe as nossas classes. Essas classes são chamadas de linearmente inseparáveis.

Logo, o perceptron não será capaz de identificar e separar estas duas classes, ou seja, não conseguiremos a resposta desejada com um perceptron de uma única camada.

3. Múltiplas Camadas

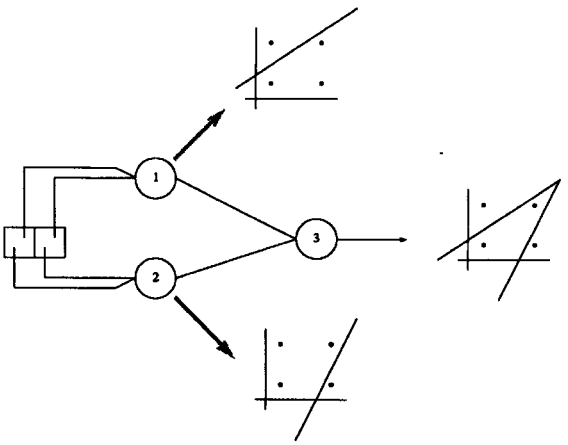
3.1. O Perceptron com Múltiplas Camadas

O principal objetivo do perceptron de múltiplas camadas é resolver problemas como o que foi citado no final do capítulo anterior, ou seja, problemas linearmente independentes.

Poderíamos pensar em usar mais de um perceptron. Como temos duas entradas, colocaríamos cada entrada em um perceptron diferente. Então ligaríamos

estes perceptrons a um terceiro perceptron, ou seja, as saídas seriam as entradas de um outro perceptron, e este por sua vez produziria a saída final.

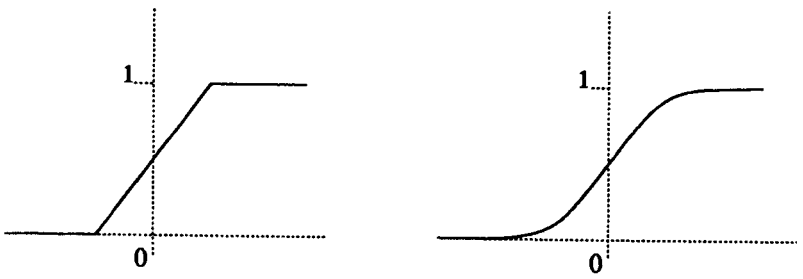
Então teremos uma primeira camada de perceptrons, que serão as entradas, uma ou mais camadas intermediárias ou camadas escondidas, que receberão como entradas as saídas da primeira camada e finalmente a última camada, que será a camada do resultado ou saída propriamente dita. Notaremos que nas camadas escondidas os perceptrons não saberão quais serão as verdadeiras entradas.



Sendo assim, teremos um problema, pois se o resultado final não for o resultado que estávamos esperando, não saberemos qual soma deve ser alterada (aumentada ou diminuída).

Para resolvermos este problema vamos usar uma função não linear. Esta função não terá apenas dois valores como a primeira que foi apresentada. Terá regiões entre esses dois valores.

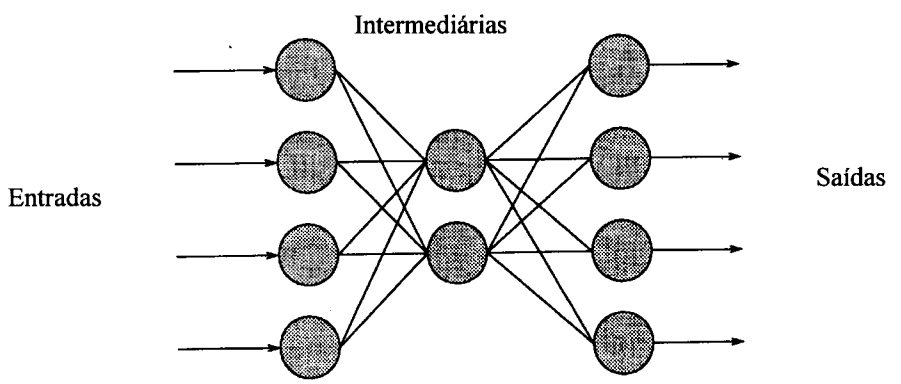
Estas regiões nos darão algumas informações a mais e assim seremos capazes de determinar quando vamos aumentar ou diminuir o valor da nossa soma ponderada ou nossos pesos.



Podemos perceber que em ambos os casos, o valor da saída será praticamente 1, se a soma ponderada ultrapassar o valor limite, e 0, caso contrário. Entretanto, quando o valor da soma ponderada for quase igual ao valor do limite, a saída do neurônio vai estar em algum ponto entre os dois extremos. Isso significa que a saída será passada de forma diferente dando maiores detalhes das entradas para os devidos neurônios. Alteramos, assim, o nosso modelo .

3.2. O Novo Modelo de Neurônio

Vamos ver os modelos organizados como uma rede, ou seja, teremos várias unidades de perceptrons organizados em camadas. Não teremos mais apenas um único neurônio mas sim um perceptron multicamadas ou *multilayer perceptron* [4]. unidades de perceptrons organizados em camadas. O modelo tem três camadas básicas: a camada de entradas, a camada de saídas e as camadas intermediárias entre as duas primeiras camadas, que conecta as duas camadas e que é chamada de camada escondida. Cada unidade na camada escondida e na camada de saídas funciona como o nosso perceptron propriamente dito com apenas uma diferença: agora, não haverá mais a função degrau para determinar qual será a saída do perceptron.



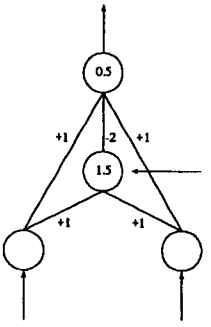
As unidades na camada de entradas servem para distribuir os valores que os seus perceptrons receberam para a próxima camada, e assim não apresentam uma soma ponderada e nem um valor limite.

3.3. Resolvendo o Problema do XOR

Não tínhamos conseguido resolver o problema do XOR (ou-exclusivo) com um perceptron de apenas uma única camada. Vamos tentar resolver este problema com um perceptron de múltiplas camadas.

Primeiramente verificaremos a possibilidade de construir uma rede que possa resolver este problema. Podemos representar o problema do XOR com uma rede de duas camadas, ou seja, na verdade teremos uma estrutura de três camadas, com duas entradas, uma unidade de perceptron como a camada escondida e uma unidade na camada de saída.

Vamos mostrar como ficaria nossa rede através da figura abaixo. Onde houver os pesos de cada linha intermediária e os valores limites de cada perceptron que estão dentro de cada perceptron, devemos também lembrar que como saída só teremos 1 ou 0 e que as unidades das camadas intermediárias não produzem uma saída propriamente dita, mas sim entradas para as unidades da camada de saída.

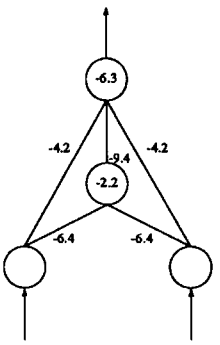


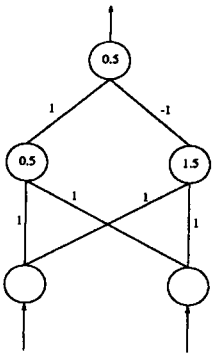
Camada Escondida

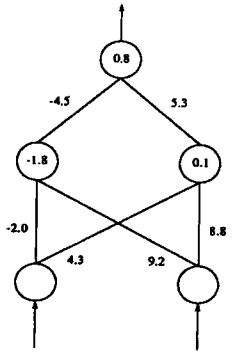
Devemos acompanhar pela figura acima como essa rede se comporta para o problema do XOR. Assim, perceberemos que quando as entradas forem 0 e 0, as camadas escondidas também estão desligadas. Então, a saída é 0 ou desligada. Se a entrada for 0 e 1 ou 1 e 0, a saída será 1. Caso as entradas sejam 1 e 1 a saída também será 0 (vide figura da pág.10).

A função XOR, de maneira geral, se comporta da seguinte forma: quando as duas entradas tem o mesmo valor, teremos como saída o valor 0, e quando as duas entradas forem diferentes, a saída será 1.

Podemos representar a rede do XOR através de outras configurações de rede.







Podemos, também, distinguir as duas últimas configurações de rede da primeira. Basta uma olhada para descobrirmos que, na primeira configuração, os perceptrons da camada das entradas se comunicam com a camada intermediária, além da comunicação com os perceptrons da camada de saída, enquanto nas duas últimas configurações, os perceptrons da camada de entradas comunicam-se somente com os perceptrons da camada intermediária.

A primeira configuração recebe o nome de rede direta, já que se comunica diretamente com a camada de saída, enquanto a segunda recebe o nome de rede indireta, já que a comunicação é feita somente com camadas intermediárias.

4. Técnicas de Aprendizado

Entre as características de uma rede neural, o capacidade de aprender é a mais interessante e é através desta característica que o desempenho de uma rede neural é melhorada. Uma rede neural aprende quando ocorre o ajuste de seus pesos em suas respectivas junções sinápticas (sinapses) e consequentemente um ajuste no seu valor limite (*treshold*).

A cada iteração, há um ajuste nos pesos e consequentemente um aumento no conhecimento da rede. Uma boa definição para o aprendizado em redes neurais é [5]:

"O aprendizado é um processo pelo qual os parâmetros aleatórios de uma rede neural são adaptados através da continuidade do processo de simulação do ambiente na qual a rede está. O tipo de aprendizado é determinado pela maneira como os parâmetros são alterados."

Para a realização do processo de aprendizado é utilizado um algoritmo específico, que por sua vez não é único, existindo uma variedade de algoritmos de aprendizado com suas vantagens e desvantagens.

A diferença básica entre esses algoritmos é a maneira como os ajustes nas junções sinápticas são executados. Vamos abordar brevemente alguns deles.

4.1. Back Propagation

Vamos supor que a rede foi programada para separar sílabas de uma palavra. Como exemplo serão apresentadas duas palavras: "demonstração" com quatro sílabas e "demonstrativo" com cinco sílabas [3].

Vamos supor que a primeira palavra foi apresentada à rede e quando perguntamos qual o número de sílabas da palavra, obtemos como resposta quatro. Através do aprendizado supervisionado, percebemos que a resposta esta correta e portanto não será necessário qualquer alteração nos pesos.

Agora, apresentamos a nossa segunda palavra. Devemos notar que as duas palavras são iguais até a terceira sílaba. Quando a palavra foi apresentada à rede, obtivemos como resposta quatro. Agora, pelo aprendizado supervisionado, percebemos que a resposta esta incorreta, já que a resposta esperada é cinco. Então, como as palavras são iguais até a terceira sílaba, a rede volta até esta sílaba e a partir daí começam as alterações nos pesos da nossa rede. Vamos então alterar os pesos até que a resposta esperada seja obtida.

Nesta técnica, podemos perceber que a nossa rede volta atrás buscando suas "experiências" anteriores, daí o nome *back propagation*.

4.2. Error - Correction Learning[5]

Primeiramente, teremos uma resposta $d_k(n)$, que é a resposta esperada para um neurônio k em um tempo n. A resposta dada pelo neurônio será $y_k(n)$ produzida por um estímulo $x(n)$.

Geralmente a resposta atual $y_k(n)$ do neurônio k é diferente da resposta desejada $d_k(n)$. Podemos então definir um sinal de erro, que será a diferença entre a resposta desejada $d_k(n)$ e a resposta atual $y_k(n)$:

$$e_k(n) = d_k(n) - y_k(n).$$

O aprendizado por correção de erros pode ser minimizado se utilizarmos uma função de custo baseada no sinal de erro. Um critério comumente usado para a função de custo é o critério de erro dos mínimos quadrados. A minimização do custo da função é feita através do método do gradiente decrescente.

A rede é otimizada pela minimização dos pesos das sinapses das redes.

O ajuste feito nas sinapses é dado por:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n),$$

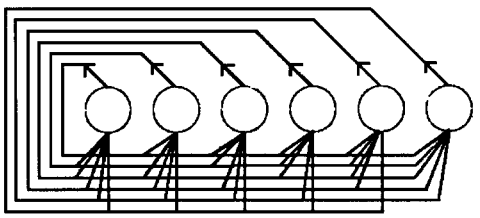
onde η é uma constante positiva que determina a taxa de aprendizado, ou seja, o ajuste é proporcional ao produto do sinal de erro e da entrada da sinapse. Se η é pequeno a taxa converge lentamente, levando mais tempo para que a resposta desejada seja alcançada. Se η for grande converge rapidamente, mas neste caso, poderá ocorrer uma divergência.

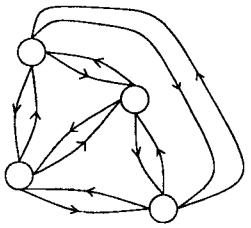
5. Redes Hopfield

John Hopfield foi um dos maiores contribuintes na área de redes neurais durante os anos oitenta. Hopfield estudou uma rede autoassociativa, que tinha características muito parecidas com perceptrons, além de relacionar a utilização de redes com funções de energia e também com outros sistemas físicos.

5.1. Definição

A rede de Hopfield corresponde a um número de nós, onde cada nó esta ligado com todos os outros nós pertencentes a rede, por isso essa rede é chamada de *fully-connected network.*





Esta rede também é chamada de rede *symmetrically-weighted*, pois se cada nó esta ligado a todos os outros, entre dois nós teremos dois caminhos com sentidos contrários, porém terão os mesmos pesos.

Como os perceptrons, cada nó tem um *treshold*, uma soma ponderada de suas entradas e uma função degrau. Os nós também funcionam como os perceptrons, calculando a soma ponderada para as suas entradas e comparando-a com o treshold. Depois, a função degrau é usada para determinar qual será a saída do nó. Esta rede tem como entrada dois valores: 0 ou 1, quando a rede é binária ou -1 ou +1, quando a rede é bipolar.

A principal característica dessa rede é o fato de todos os nós se comunicarem. Essa diferença de arquitetura faz a rede operar de um modo diferente, pois todas as entradas da rede são aplicadas em todos os nós. Por isso, a rede opera em ciclos, passando por vários estados sucessivos até quando convergir para um estado estável. Este estado é alcançado quando o valor dos nós não se alteram.

A saída da rede será o valor de todos os nós quando um estado estável for alcançado.

A operação da rede é diferente do sistema de perceptrons, pois neste sistema, as entradas produzem uma saída qualquer, enquanto que na rede Hopfield, uma entrada implicará em uma saída, que por sua vez será uma entrada para uma outra saída e assim sucessivamente até que não ocorra mais nenhuma mudança de ciclo para ciclo.

5.2. Aplicações da Rede Hopfield

Uma aplicação seria calcular a menor energia de uma determinada amostra. Começaríamos calculando a energia de um sistema numa dada configuração (E1), atavés da configurações de seus respectivos spins, atibuíriamos valores para os spins. Por exemplo, spins verticais orientados para cima teriam valor +0,5, enquanto spins orientados para baixo teriam valor -0,5. Para calcular a energia de uma determinada amostra, faríamos o somatório das iterações de um spin com os seus primeiros vizinhos. Este modelo é conhecido como Modelo de Ising.

Depois mudaríamos a configuração aleatoriamente e calcularíamos novamente a energia (E2), se $E1 > E2$, então $E1 = E2$, caso $E1<E2$, então sortearíamos um número aleatório entre zero e um ($0 \leq r \leq 1$). Se $r < e^{-(E2-E1)}$, então E1=E2, caso contrário, E1 permanece com o seu valor, e haverá uma nova mudança no sistema e um novo cálculo de E2.

Se acharmos sempre o mesmo valor em sucessivas iterações, então encontramos estado estável, que é a menor energia do sistema. Esse algoritmo, que calcula a menor energia de um sistema, é conhecido como Algoritmo de Metropolis.

Podemos usar a Rede de Hopfield com o mesmo objetivo, pois esta também realiza vários ciclos até encontrar um estado estável.

Referências

[1] - Simon Catterall - smc@npac.syr.edu

[2] - "An Introduction to the Modeling of Neural Networks", P. Peretto, Cambridge University Press, ISBN 0 521 42487 9, 1992.

[3] - "Neural Networks Computers with Intuition", S. Brunak and B. Lautrup, World Scientific, ISBN 9971-50-939-3, 1990.

[4] - "Neural Computing: An Introduction", R. Beale and T. Jackson, IOP Publishing Ltda, ISBN 0-85274-262-2, 1994.

[5] - "Neural Networks - A Comprehensive Foundation", S. Haykin, Macmillan College Publishing Company, ISBN 0-02-352761-7, 1994.

[6] - "Neural Networks: Algorithms, Applications and Programming Techniques", J. A. Freeman and D. M. Skapura, Addison-Wesley Publishing Company, ISBN 0-201-51376-5, 1992.

[7] - "http://suhep.phy.syr.edu/courses/modules/MM/index.html"

OBS: A maioria das figuras desta Nota Técnica foram obtidas das referências acima relacionadas. A baixa qualidade de algumas destas figuras é devido aos equipamentos utilizados.