Análise do Perceptron

Para entender como funciona um perceptron, vamos continuar considerando um perceptron com somente duas entradas, x_1 e x_2, e uma saída s.

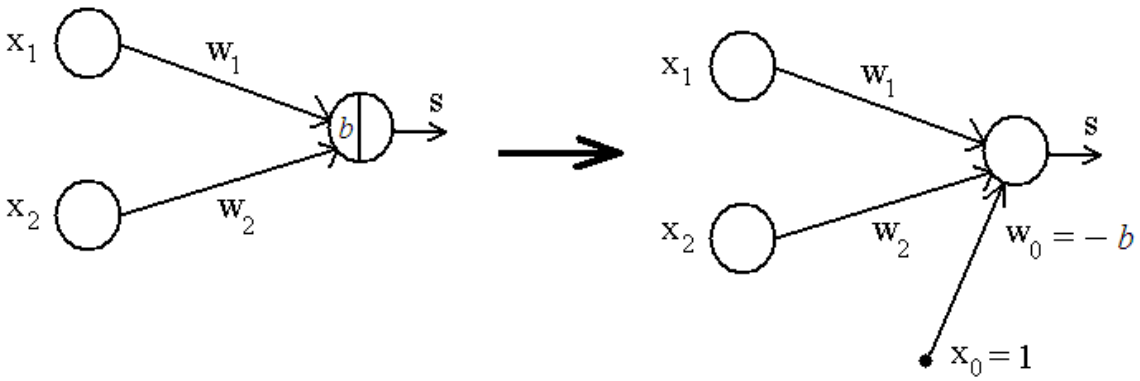
O neurônio de saída tem limiar b, de maneira que o seu nível de ativação é

$$u = \omega_1 x_1 + \omega_2 x_2 - b .$$

Podemos reescrever essa equação como

$$u = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 ,$$

onde definimos: $\omega_0 = -b$ e $x_0 = 1$ (veja a figura abaixo).

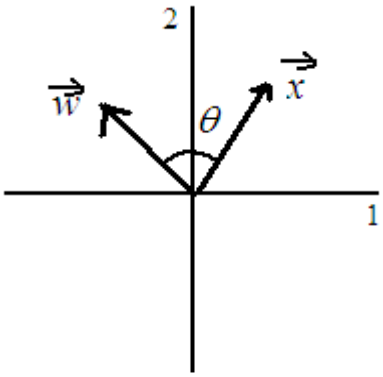


Portanto, podemos tratar o perceptron de duas entradas e limiar b como um perceptron de três entradas (uma delas com valor e peso fixos) e limiar zero.

Para facilitar a análise a ser feita, e sem perda de generalidade, podemos fazer o peso $\omega_0 = -b = 0$, de maneira que o perceptron fica somente com duas entradas.

Para um dado valor do vetor de pesos $\vec{\omega} = (\omega_1, \omega_2)$, o que ocorre quando um padrão de entrada $\vec{x} = (x_1, x_2)$ é fornecido ao perceptron?

Para responder a esta questão, consideremos a figura a seguir.



A figura mostra um vetor de pesos $\vec{\omega}$ e um padrão de entrada $\vec{x}$ arbitrários. Notem que há um ângulo θ entre os dois vetores. O nível de ativação do neurônio de saída é

$$u = \omega_1 x_1 + \omega_2 x_2 .$$

Matematicamente, o nível de ativação corresponde ao *produto interno* ou *escalar* entre os dois vetores (veja as notas de revisão sobre álgebra linear):

$$u = \vec{\omega}.\vec{x} = \left|\vec{\omega}\right|.\left|\vec{x}\right| \cos\,\theta \ .$$

Se o valor desse produto escalar for positivo ou nulo, a saída (resposta) do perceptron será +1; se o valor for negativo, a saída será 0.

Observem que a expressão para o produto escalar acima envolve o produto dos módulos dos vetores $\vec{\omega}$ e $\vec{x}$, que é sempre positivo, vezes o cosseno do ângulo θ entre eles, que pode ser positivo, negativo ou nulo. Logo, o que decide se o valor do produto escalar – e, portanto, do nível de ativação do neurônio de saída – é positivo, nulo ou negativo é o valor do ângulo θ.

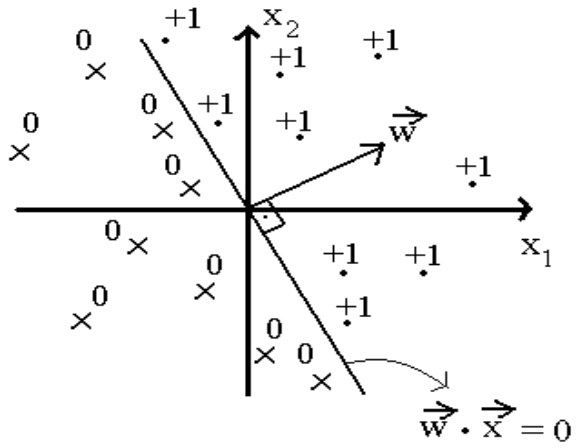
Dado um vetor de pesos $\vec{\omega}$, todos os vetores $\vec{x}$ cujos ângulos θ com $\vec{\omega}$ forem, ou menores ou iguais a 90°, ou maiores ou iguais a 270°, serão classificados pelo perceptron como pertencentes à classe designada por + 1. Já os vetores cujos ângulos estiverem entre 90° e 270° (excluindo estes valores) serão classificados como pertencentes à classe designada por 0.

A situação crítica (aquela que define a transição entre uma classe ou outra) ocorre para os vetores $\vec{x}$ que formam ângulos iguais a 90° ou 270° com $\vec{\omega}$. Esses são os vetores cujos produtos escalares com $\vec{\omega}$ são iguais a zero.

Visualizando geometricamente, $\vec{\omega} \cdot \vec{x} = 0$ define uma reta passando pela origem do espaço bi-dimensional (x_1, x_2):

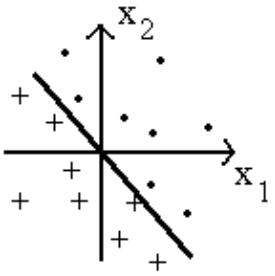
$$\omega_1 x_1 + \omega_2 x_2 = 0 \Rightarrow x_2 = -\frac{\omega_1}{\omega_2} x_1 .$$

O vetor $\vec{\omega}$ é perpendicular à reta (veja a figura abaixo).

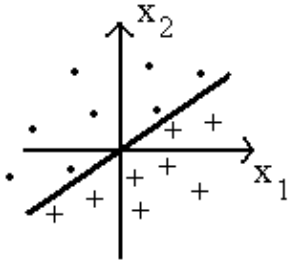


Os pontos à direita da reta são classificados como sendo da classe +1, pois os vetores correspondentes formam ângulos menores que 90° ou maiores que 270° com $\vec{\omega}$. Já os pontos à esquerda da reta são classificados como sendo da classe 0, pois os vetores correspondentes formar ângulos com $\vec{\omega}$ entre 90° e 270°.

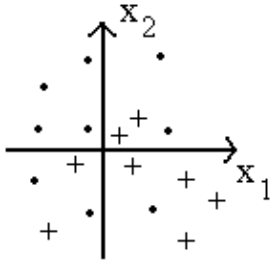
Quando houver um conjunto de pontos num espaço bi-dimensional pertencentes a duas classes distintas e for possível encontrar uma reta capaz de dividir o espaço em duas regiões, uma contendo os pontos de uma classe e a outra contendo os pontos da outra classe, diremos que o conjunto de pontos é *linearmente separável* (veja a figura a seguir).



Linearmente separável



Linearmente separável



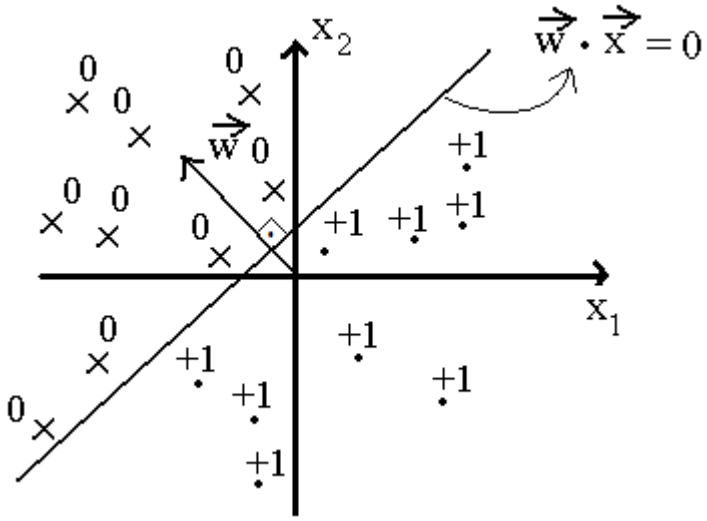
Não-linearmente separável

Um perceptron só consegue classificar padrões que sejam linearmente separáveis. Um problema não linearmente separável, como o do terceiro caso da figura acima, está além das capacidades computacionais do perceptron.

Se tivéssemos considerado o limiar *b* como não nulo, a análise feita acima não sofreria grandes alterações. A condição $\vec{\omega} \cdot \vec{x} = 0$ continuaria nos dando uma reta no plano (x_1, x_2), só que deslocada da origem por um fator constante:

$$x_2 = -\frac{\omega_1}{\omega_2} x_1 + \frac{b}{\omega_2} .$$

A figura abaixo ilustra este caso.



Assim como no caso sem o viés *b*, um conjunto de pontos pertencentes a duas classes distintas que não puder ter a sua separação nas duas classes feita por uma reta não poderá ser corretamente classificado por esse perceptron.

O conceito de separabilidade linear é extensível para mais dimensões.

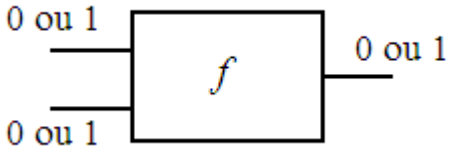
Definição: Dois conjuntos de pontos A e B em um espaço *n*-dimensional são ditos *linearmente separáveis* se existirem *n*+1 números reais ω_0, ω_1, ω_2, ... , ω_n tais que todo ponto $(x_1, x_2, \ldots, x_n) \in$ A satisfaça $\sum_{i=1}^{n} \omega_i x_i \geq \omega_0 x_0$ e todo ponto $(x_1, x_2, \ldots, x_n) \in$ B satisfaça $\sum_{i=1}^{n} \omega_i x_i < \omega_0 x_0$.

Podemos agora pensar num perceptron com 1 neurônio na camada de saída e com *N* neurônios na camada de entrada. A tarefa do perceptron de classificar um conjunto de *M* padrões *N*-dimensionais em duas classes distintas só é possível se os *M* padrões forem linearmente separáveis. Neste caso, a separação entre as duas classes não será mais feita por uma linha reta, mas por um *hiperplano* definido por $\vec{\omega} \cdot \vec{x} + b = 0$ (supondo $b \neq 0$).

O fato de que um perceptron só consegue resolver tarefas linearmente separáveis impõe severas limitações à sua aplicabilidade em situações práticas.

Consideremos novamente o caso do perceptron com duas entradas e vamos supor que os valores dessas entradas só podem ser 0 ou 1 (entradas binárias). Uma função de variáveis binárias que dê uma resposta binária é chamada de *função booleana* (em homenagem ao matemático inglês George Boole (1815-1864) que estudou as propriedades algébricas das funções binárias).

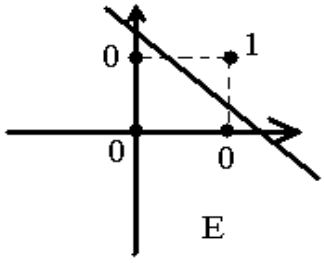
Em duas dimensões, uma função booleana genérica fornece um modelo matemático para uma porta lógica usada em circuitos (veja a figura a seguir):

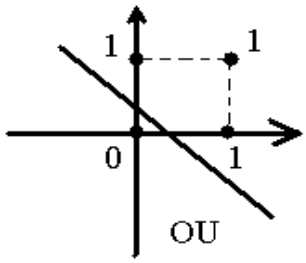


Nesse caso bi-dimensional, há 16 funções booleanas possíveis, dadas a seguir:

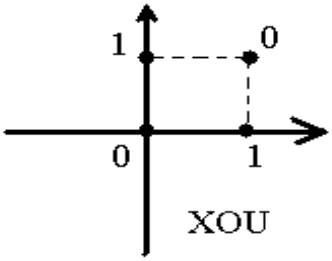
X_1	X_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

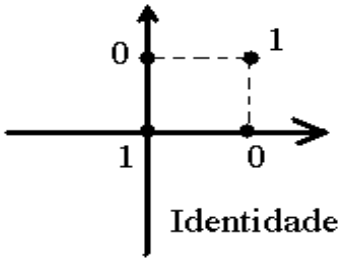
A função f_0 é a função 0, a função f_8 é a função lógica "E", a função f_{14} é a função lógica "OU", a função f_6 é a função lógica "XOU", etc. Destas 16 funções, apenas 14 são linearmente separáveis. Vamos dar o exemplo de duas, a "E" e a "OU", e deixar as outras 12 para serem feitas como exercício.





As duas funções que não são linearmente separáveis são a "XOU" e a identidade (f_6 e f_9):





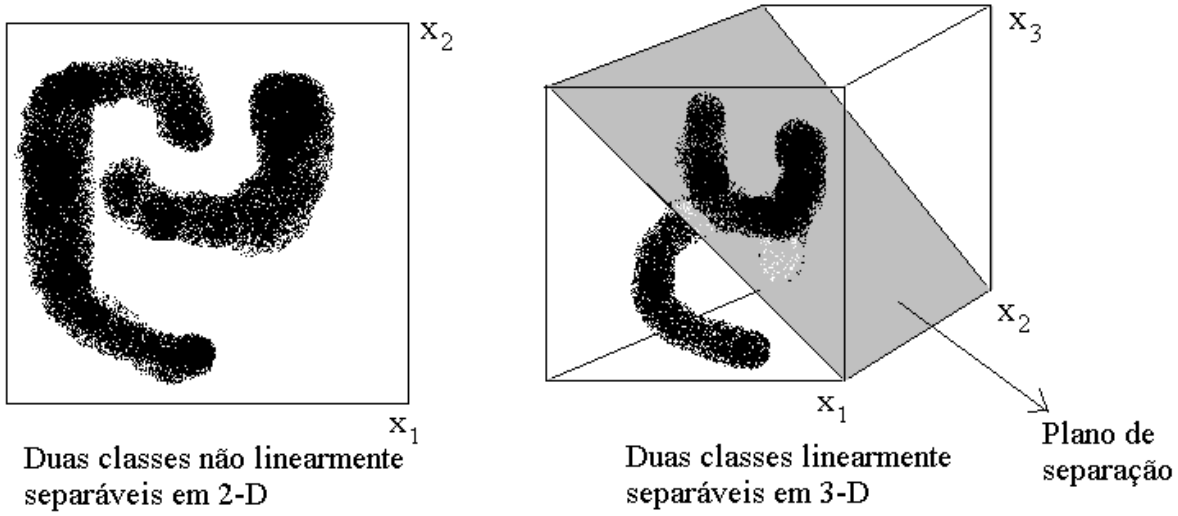
Um perceptron não consegue implementar a função lógica "XOU" bi-dimensional.

Para que o perceptron possa implementar a função lógica "XOU" é necessário acrescentar uma camada extra (oculta) entre a camada de entrada e a de saída (veja a solução que McCulloch e Pitts deram para o problema do ou-exclusivo na aula 2).

Uma questão interessante é saber quantas funções booleanas linearmente separáveis existem em n dimensões. Para $n = 2$, temos 14 das 16 possíveis. Para $n = 3$, temos 104 das 256 possíveis. Para $n = 4$, temos 1882 das 65.536 possíveis. Não existe uma fórmula para calcular este número para n genérico.

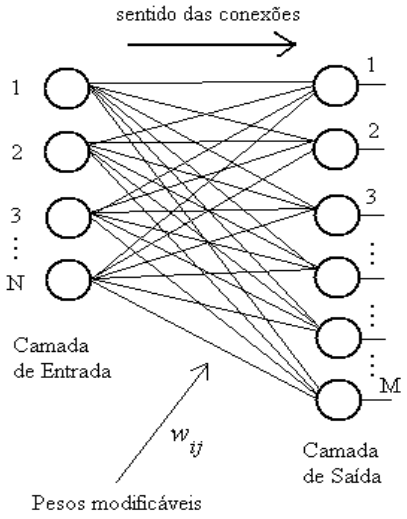
Às vezes, um problema pode não ser linearmente separável em n dimensões, mas pode sê-lo em $(n+1)$ dimensões. Neste caso, a questão seria encontrar uma nova característica dos padrões originais projetados na camada da retina do perceptron que possa, quando adicionada como uma dimensão a mais na representação dos padrões, propiciar a sua separação linear.

Como exemplo, sejam as figuras abaixo mostrando duas classes de padrões que não são linearmente separáveis em duas dimensões, mas o são em três, caso a terceira dimensão seja escolhida convenientemente.



No desenho acima, a figura da esquerda, em 2 dimensões, é a projeção no plano (x_1, x_2) das duas classes mostradas na figura da direita em 3 dimensões. Vemos que as duas classes são linearmente separáveis em 3 dimensões, mas não o são em 2 dimensões.

Dá-se abaixo um algoritmo para o treinamento de um perceptron com *N* unidades na camada de entrada e *M* unidades na camada de saída.



$$u_i = \sum_{j=1}^{N} \omega_{ij} x_j - b_i$$

$$S_i = f(u_i) = \begin{cases} 1 \text{ se } u_i \geq 0 \\ 0 \text{ se } u_i < 0 \end{cases}$$

i

Passo 0. Inicialize os pesos w_{ij} (em geral, inicializa-se os pesos de maneira que todos tenham o valor zero ou valores aleatórios pequenos). Inicialize uma variável para contar o número de épocas de treinamento. Leia os valores dos viéses, b_i , $i = 1, \ldots, M$, e da taxa de aprendizagem η (suposta a mesma para todos os neurônios). Leia o valor do número máximo de épocas de treinamento nº_max.

Passo 1. Até que o critério de parada seja satisfeito, repita os passos 1–8.

Passo 2. Para cada par composto por um padrão de treinamento $\vec{p}$ e pela respectiva saída desejada $\vec{a}$, repita os passos 3–7.

Passo 3. Leia a ativação de cada unidade de entrada, $j = 1, \ldots, N$:

$x_j = p_j$.

Passo 4. Calcule a ativação de cada unidade de saída, $i = 1, \ldots, M$:

$$u_i = \sum_{j=1}^{N} \omega_{ij} x_j - b_i \ .$$

Passo 5. Calcule a saída de cada unidade de saída, $i = 1, \ldots, M$:

$$S_i = \begin{cases} 1 \text{ se } u_i \geq 0 \\ 0 \text{ se } u_i < 0 \end{cases}$$

Passo 6. Modificação dos pesos ω_{ij}, $i = 1, \ldots, M$ e $j = 1, \ldots, N$:

Se $S_i \neq a_i$, *Então*

Erro = $a_i - S_i$

Para $j = 1, \ldots, N$ *Faça* $\omega_{ij}(\text{novo}) = \omega_{ij}(\text{velho}) + \eta.\text{Erro}.\ x_j$

Caso Contrário não mude os pesos: $\omega_{ij}(\text{novo}) = \omega_{ij}(\text{velho})$

Passo 7. Incremente o número de épocas de treinamento de 1 unidade.

Passo 8. Critério de parada: *Se* nenhum peso mudou no *Passo 6*, *Pare;*

Caso Contrário

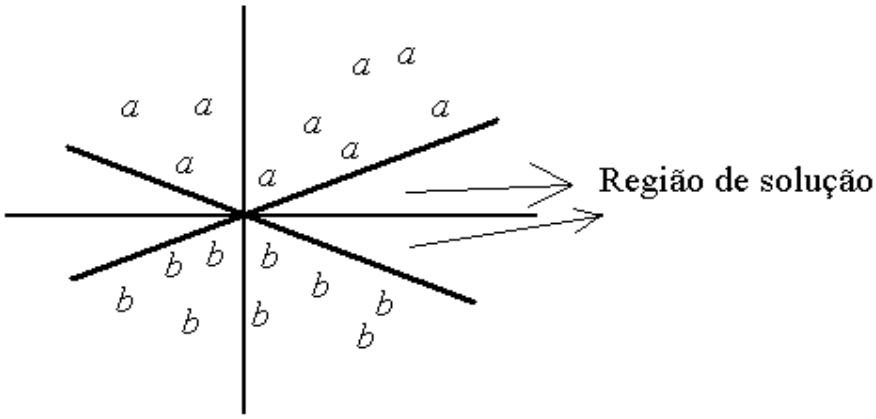
Se número de épocas = nº_max, *Pare;*

Caso Contrário, Continue

Note que, pelo critério de parada, o aprendizado só termina quando o perceptron encontra vetores de peso (um para cada unidade de saída) capazes de classificar corretamente todos os padrões $\vec{p}$ do conjunto de treinamento (caso isto não ocorra dentro de um número máximo de épocas de treinamento, o programa é parado para não continuar indefinidamente).

Em geral, caso os padrões de entrada sejam linearmente separáveis, existem inúmeros vetores de pesos capazes de classificar corretamente os padrões de entrada. O algoritmo de aprendizagem do perceptron não se preocupa com qual deles foi o encontrado.

Em duas dimensões (veja a figura abaixo), caso haja um conjunto de pontos linearmente separável em duas classes, "a" e "b", existe uma região de solução tal que qualquer reta passando por ela divide os pontos em duas regiões.



Note que basta que a reta fornecida como resposta pelo perceptron após uma dada época de treinamento caia dentro da região de solução (mesmo que seja por uma quantia infinitesimal) para que o treinamento seja interrompido. Isto não faz do perceptron uma "máquina" muito robusta, pois alguma pequena flutuação nos valores dos pesos finais pode fazer com que a reta obtida saia da região de solução.

Uma das causas do sucesso dos perceptrons na década de 1960 foi o fato de que se pode provar um teorema garantindo que, caso um problema de classificação tenha solução por um perceptron, isto é, se ele for linearmente separável, a regra de aprendizado do perceptron faz com que o vetor de pesos $\vec{\omega}$ convirja para um vetor que dê uma solução para o problema em um número *finito* de passos.

Este é o chamado *Teorema da Convergência do Perceptron*, que foi provado por Rosenblatt em 1962. Vamos apresentar aqui uma versão (não a mais rigorosa) da prova deste teorema (vocês podem pular essa prova se quiserem).

Teorema da Convergência do Perceptron:
Seja o problema de classificação solúvel com pesos apropriados w_{ij}, conectando as unidades da camada A (indexadas por j) às unidades da camada R (indexados por i), obtidos por meio da regra de aprendizagem do perceptron. Suponhamos que os vetores de entrada da camada R (os vetores das características), denotados por $\boldsymbol{x}$, sejam todos limitados, isto é, existe uma constante M tal que a desigualdade $|\boldsymbol{x}| < M$ seja sempre satisfeita. Então, com a escolha do parâmetro de taxa de aprendizagem $\eta_i = 1/|\boldsymbol{x}|$ para o i-ésimo neurônio da camada R, o algoritmo de aprendizagem do perceptron irá sempre encontrar uma solução após um número finito de passos adaptativos t para os pesos w_{ij} (somente os passos em que houver um erro, $e_i = S_i^{\text{desejada}} - S_i \neq 0$, são contados).

Prova:
Como cada elemento da camada R opera independentemente dos outros, para provar o teorema basta considerar apenas uma unidade de saída na camada R, ou seja, podemos desconsiderar o índice i usado no enunciado do teorema. Vamos denotar por $\boldsymbol{w}^* = (w^*_1, w^*_2, ..., w^*_N)$ o vetor $\boldsymbol{w}$ para o qual o perceptron resolve o problema de classificação corretamente (a existência de um tal vetor é requerida pelo enunciado do teorema).

Então, existe uma constante $\delta > 0$ tal que,

$$\boldsymbol{w}^* \cdot \boldsymbol{x} > \delta \text{ se } S^{\text{desejada}}(\boldsymbol{x}) = 1$$

e

$$\boldsymbol{w}^* \cdot \boldsymbol{x} < -\delta \text{ se } S^{\text{desejada}}(\boldsymbol{x}) = 0,$$

onde $S^{\text{desejada}}(\boldsymbol{x})$ representa a saída desejada (correta) da unidade quando o padrão $\boldsymbol{x}$ é apresentado na entrada.

Seja $\boldsymbol{w}(t)$ o vetor de pesos do perceptron obtido após t passos de modificação a partir de algum valor inicial arbitrário. A próxima modificação em $\boldsymbol{w}$ ocorrerá quando houver um erro, $e = S^{\text{desejada}} - S = \pm 1$. Essa modificação fará com que $\boldsymbol{w}(t + 1) = \boldsymbol{w}(t) + \eta e \boldsymbol{x}$. Portanto,

$$\boldsymbol{w}(t + 1) \cdot \boldsymbol{w}(t + 1) = (\boldsymbol{w}(t) + \eta e \boldsymbol{x}) \cdot (\boldsymbol{w}(t) + \eta e \boldsymbol{x}) =$$

$$= \boldsymbol{w}(t) \cdot \boldsymbol{w}(t) + 2\eta e \boldsymbol{w}(t) \cdot \boldsymbol{x} + \eta^2 e^2 \boldsymbol{x} \cdot \boldsymbol{x}.$$

Notemos que a saída da unidade é $S = \text{sinal}(\boldsymbol{w}(t) \cdot \boldsymbol{x})$, de maneira que,

$$e \boldsymbol{w}(t) \cdot \boldsymbol{x} = (S^{\text{desejada}} - S) \boldsymbol{w}(t) \cdot \boldsymbol{x} \leq 0.$$

Então (como $\eta > 0$),

$$\boldsymbol{w}(t) \cdot \boldsymbol{w}(t) + 2\eta e \boldsymbol{w}(t) \cdot \boldsymbol{x} + \eta^2 e^2 \boldsymbol{x} \cdot \boldsymbol{x} \leq \boldsymbol{w}(t) \cdot \boldsymbol{w}(t) + \eta^2 e^2 \boldsymbol{x} \cdot \boldsymbol{x}.$$

Assumindo (veja o enunciado do teorema) que $\eta = 1/|\boldsymbol{x}|$, temos que $\eta^2 e^2 = 1/|\boldsymbol{x}|^2 = 1/(\boldsymbol{x} \cdot \boldsymbol{x})$. Combinando o que obtivemos até agora,

$$\boldsymbol{w}(t + 1) \cdot \boldsymbol{w}(t + 1) \leq \boldsymbol{w}(t) \cdot \boldsymbol{w}(t) + (\boldsymbol{x} \cdot \boldsymbol{x})/(\boldsymbol{x} \cdot \boldsymbol{x}) \Rightarrow$$

$$\Rightarrow |\boldsymbol{w}(t + 1)|^2 \leq |\boldsymbol{w}(t)|^2 + 1.$$

A aplicação recursiva desta fórmula nos dá,

$$|\boldsymbol{w}(t + n)|^2 \leq |\boldsymbol{w}(t)|^2 + n,$$

ou

$$|\boldsymbol{w}(t)|^2 \leq |\boldsymbol{w}(0)|^2 + t. \qquad (1)$$

Por outro lado, a cada passo de modificação temos,

$$\boldsymbol{w}(t + 1) \cdot \boldsymbol{w}^* = (\boldsymbol{w}(t) + \eta e \boldsymbol{x}) \cdot \boldsymbol{w}^* = \boldsymbol{w}(t) \cdot \boldsymbol{w}^* + \eta e \boldsymbol{x} \cdot \boldsymbol{w}^*.$$

O valor de e pode ser +1 se a saída desejada for +1 e a unidade estiver dando como saída 0, ou pode ser −1 se a saída desejada for 0 e a unidade de saída estiver dando como saída +1. No primeiro caso, $\boldsymbol{x} \cdot \boldsymbol{w}^* > \delta$, e, no segundo caso, $\boldsymbol{x} \cdot \boldsymbol{w}^* < -\delta$. Tanto em um caso como no outro o produto de e por $\boldsymbol{x} \cdot \boldsymbol{w}^*$ nos dá a desigualdade,

$$e\boldsymbol{x} \cdot \boldsymbol{w}^* > \delta.$$

Então,

$$\boldsymbol{w}(t+1) \cdot \boldsymbol{w}^* > \boldsymbol{w}(t) \cdot \boldsymbol{w}^* + \eta\delta.$$

A aplicação recursiva desta fórmula nos dá,

$$\boldsymbol{w}(t+n) \cdot \boldsymbol{w}^* > \boldsymbol{w}(t) \cdot \boldsymbol{w}^* + n\eta\delta,$$

ou

$$\boldsymbol{w}(t) \cdot \boldsymbol{w}^* > \boldsymbol{w}(0) \cdot \boldsymbol{w}^* + t\eta\delta > \boldsymbol{w}(0) \cdot \boldsymbol{w}^* + t\delta/M, \qquad (2)$$

onde se usou o dado do enunciado, $\eta = 1/|\boldsymbol{x}| > 1/M$.

Segundo a desigualdade (2), a projeção do vetor de pesos $\boldsymbol{w}(t)$ sobre o vetor fixo $\boldsymbol{w}^*$ deve crescer em função de t de uma maneira mais rápida que a linear. No entanto, o resultado da desigualdade (1) mostra que o módulo de $\boldsymbol{w}(t)$ não pode crescer mais rapidamente do que $\sqrt{t}$. Estes dois resultados estão em contradição e a única maneira de compatibilizá-los é o número de passos de modificação t ser limitado. Se o número de passos em que $\boldsymbol{w}$ sofre modificação for limitado, o treinamento do perceptron deve terminar após um número finito de modificações. Isto prova o teorema da convergência do perceptron.

■

O fato de que existe um teorema provando a convergência do algoritmo de treinamento do perceptron para uma solução, caso ela exista, deu muita popularidade ao perceptron na década de 1960. Devido à sua fácil implementação, esperava-se que o perceptron viesse a ter inúmeras aplicações práticas.

Em particular, construiu-se um perceptron, denominado Mark I, no Laboratório Aeronáutico de Cornell, nos Estados Unidos, que tinha uma retina com 40×40 unidades conectada a uma câmera de vídeo, 512 unidades na camada A e 8 unidades na camada R. Os pesos sinápticos eram modificados por potenciômetros controlados por motores. Porém, esta, assim como outras tentativas de implementação do perceptron em hardware, não obteve muito sucesso em aplicações práticas.

No fim da década de 1960, após muitos fracassos e muito dinheiro do governo e indústrias norte-americanas gasto, o entusiasmo pelos perceptrons arrefeceu. Isso levou a uma grande redução no interesse e no financiamento de novas pesquisas sobre redes neurais. O descrédito pelas redes neurais em função do fracasso dos perceptrons durou mais de uma década.

Um problema prático com os perceptrons (mesmo sabendo que existe uma solução) é que chegar a uma solução que classifique corretamente um dado conjunto de padrões de treinamento pode ser uma tarefa extremamente *vagarosa*. Isso ocorre em parte pela maneira como o aprendizado é feito, pois só há correção nos pesos quando a resposta é errada. Isto quer dizer que à medida que o sistema vai aprendendo e cometendo menos erros o aprendizado fica mais lento.

Outro problema prático, extremamente comum em implementações em hardware, é a dificuldade dos perceptrons em trabalhar com padrões ruidosos. É muito comum que os padrões apresentados à Retina estejam corrompidos por ruído (alguns pixels que deveriam ser brancos estão cinza ou pretos, ou vice-versa). Nesses casos, os pesos do perceptron podem nunca se estabilizar, mas ficam mudando para sempre. Isto ocorre porque o algoritmo de treinamento do perceptron é feito para provocar mudanças nos pesos sempre que algum erro é encontrado, mesmo que ele seja mínimo.

O ataque definitivo aos perceptrons foi feito por dois pesquisadores do MIT (*Massachussets Institute of Technology*), Marvin Minsky (1927 –) e Seymour Papert (1928 –). Eles publicaram, em 1969, um livro intitulado *Perceptrons* em que era feita análise rigorosa dos perceptrons e suas limitações.

Minsky foi um dos líderes da chamada *abordagem simbólica* ao problema da inteligência artificial. Segundo essa abordagem, a inteligência envolve primariamente operações lógicas ou manipulações de símbolos baseadas em regras. Um trecho de um relatório interno do Laboratório de Inteligência Artificial do MIT, escrito por Minsky em 1972, resume essa visão:

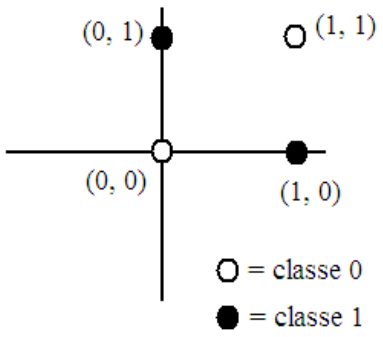
O pensamento é baseado no uso de descrições simbólicas e processos manipuladores de descrições para representar uma variedade de tipos de conhecimento — sobre fatos, processos, soluções de problemas, sobre a própria computação, em maneiras que estão sujeitas a estruturas de controle heterárquicas — sistemas nos quais o controle dos programas de solução de problemas é afetado por heurísticas que dependem do significado dos eventos. Esta habilidade de resolver novos problemas requer, em última análise, que o agente inteligente conceba, depure e execute novos procedimentos. Tal agente deve saber, em maior ou menor grau, como planejar, produzir, testar, modificar e adaptar os procedimentos; resumindo, ele deve saber muito sobre processos computacionais. Não estamos dizendo que uma máquina inteligente, ou pessoa, deve ter tal conhecimento disponível ao nível de afirmações públicas ou da consciência, mas sustentamos que o equivalente a tal conhecimento deve estar representado de uma maneira efetiva em algum lugar do sistema.

Um perceptron não usa símbolos, pelo menos não de maneira explícita, e não possui muitas estruturas de controle complexas. Qualquer estrutura de alto-nível (no sentido cognitivo) que apareça como uma de suas respostas é resultado do aprendizado a partir de exemplos, representados de maneira distribuída por um conjunto de unidades que interagem entre si de maneira bastante simples.

Isto é verdadeiro não só para os perceptrons, mas para todas as redes neurais em geral. Portanto, Minsky e Papert não se opunham somente aos perceptrons, mas a toda a chamada *abordagem conexionista* para a inteligência artificial. O ataque deles se deu contra os perceptrons porque, no fim da década de 1960, os perceptrons eram as redes neurais mais conhecidas.

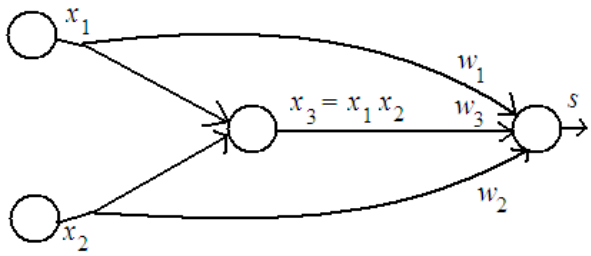
Alguns dos problemas apontados por Minsky e Papert para o perceptron foram resolvidos por modelos posteriores de redes neurais, mas alguns dos problemas ainda persistem, especialmente os relacionados com processos de alto nível cognitivo, como o aprendizado de regras da linguagem, por exemplo.

Uma das fraquezas dos perceptrons apontada por Minsky e Papert foi a sua limitação de resolver apenas problemas linearmente separáveis. Um exemplo dessa limitação foi visto no estudo da função booleana XOU. A disposição dos quatro vetores de entrada com as suas respectivas classes (0 ou 1) mostra que este não é um problema linearmente separável (veja abaixo).

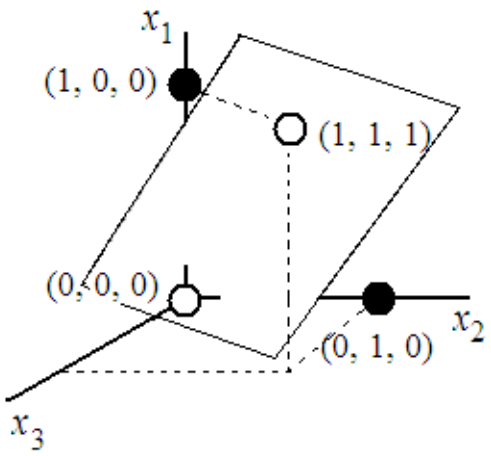


Quando estudamos o modelo de Mculloch e Pitts, vimos que eles propuseram uma rede neural para implementar esta função lógica e que ela tinha uma camada intermediária (oculta ou escondida) entre a entrada e a saída.

A mesma idéia pode ser usada para "adaptar" o perceptron para que ele resolva o problema do XOU. Por exemplo, se adicionarmos uma unidade escondida que receba as entradas x_1 e x_2 e forneça na saída o produto $x_1.x_2$ que é enviado conjuntamente com x_1 e x_2 para a unidade de saída, teremos a rede ilustrada a seguir.



Essa rede tem o potencial de resolver o problema do XOU (depende da escolha adequada dos pesos e do viés da unidade de saída). Podemos ver isto fazendo o gráfico dos quatro padrões e das suas respectivas classes, como feito na página anterior, só que agora em três dimensões.



Porém, o que ocorre neste caso é que, efetivamente, adicionou-se uma *camada extra* de neurônios ao perceptron. Considerando que a camada de entrada não é contada, o que temos neste caso é que o perceptron passou a ter *duas* camadas, uma oculta e a de saída. Portanto, ele deixou de ser o que se chama de um perceptron simples e tornou-se um *perceptron multicamadas*.

Segundo Minsky e Papert, o problema com os perceptrons multicamadas é que agora não existe mais um teorema de convergência provando que o algoritmo de aprendizagem conduz a uma solução.

Mais grave ainda, não existe um algoritmo de mudança dos pesos que leve a uma solução.

Note que o algoritmo de Rosenblatt diz que os pesos chegando à unidade de saída devem ser mudados de maneira proporcional ao erro cometido pela unidade. Este erro é fácil de ser definido porque o erro da unidade de saída é o próprio erro da rede. Mas e os erros que a(s) unidade(s) oculta(s) comete(m)? Quando a rede comete um erro na sua saída, não fica claro *qual* a unidade que está errando na camada oculta, nem *como* é que ela está errando.

Minsky e Papert achavam que o estudo de algoritmos de aprendizagem para perceptrons multicamadas era um campo sem interesse. Nas suas próprias palavras:

O problema da extensão (para mais de uma camada) não é meramente técnico. É também estratégico. O perceptron revelou-se digno de estudo apesar (ou mesmo por causa!) das suas limitações. Ele tem muitas propriedades atraentes: sua linearidade; seu teorema de aprendizado intrigante; sua simplicidade paradigmática como um exemplo de computação paralela. Não há razão para supor que qualquer dessas virtudes continue a existir na sua versão multicamadas. No entanto, consideramos como um importante problema de pesquisa a elucidação (ou a rejeição) do nosso julgamento intuitivo de que tal extensão é estéril.

Uma posição como esta é muito forte e, devido à influência de Minsky, Papert e do MIT sobre a comunidade científica e as agências financiadoras de pesquisa, o apoio à pesquisa em redes neurais foi fortemente reduzido no período entre meados da década de 1960 e meados da década de 1980.

A maior parte dos recursos destinados à inteligência artificial nesse período foi direcionada para pesquisas envolvendo a abordagem simbólica.

Apesar disso, alguns pesquisadores continuaram a propor e estudar modelos de redes neurais durante a década de 1970.

Alguns dos mais importantes são James Anderson e Stephen Grossberg nos Estados Unidos, Geoffrey Hinton no Canadá, David Willshaw na Grã Bretanha, Christian von der Malsburg na Alemanha, Teuvo Kohonen na Finlândia e Sun-Ichi Amari no Japão. Na década de 1970 os seus trabalhos não atraíram muita atenção, mas foram valorizados após o início da nova onda das redes neurais na segunda metade da década de 1980.

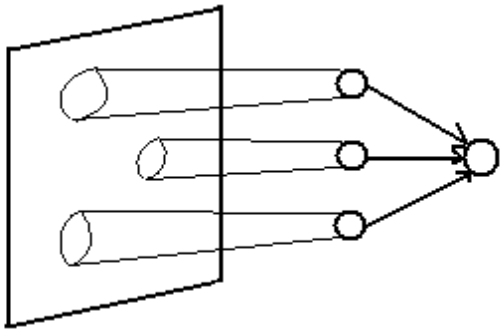
O livro Perceptrons teve um efeito drástico sobre as pesquisas em redes neurais entre a comunidade interessada em aplicações tecnológicas, composta basicamente por engenheiros e cientistas da computação. Porém, entre a comunidade de psicólogos, cientistas cognitivos e neurocientistas o efeito foi bem menor. Talvez isso tenha ocorrido por que algumas das limitações apontadas por Minsky e Papert também existem nos sistemas biológicos.

Por exemplo, um dos objetivos do perceptron era resolver problemas de percepção visual. Em seu livro, Minsky e Papert analisaram a capacidade de um perceptron em computar um predicado geométrico simples, a *conectividade* de uma figura. Em figuras simples, a conectividade pode ser identificada facilmente (veja os exemplos a seguir).

Figuras conexas

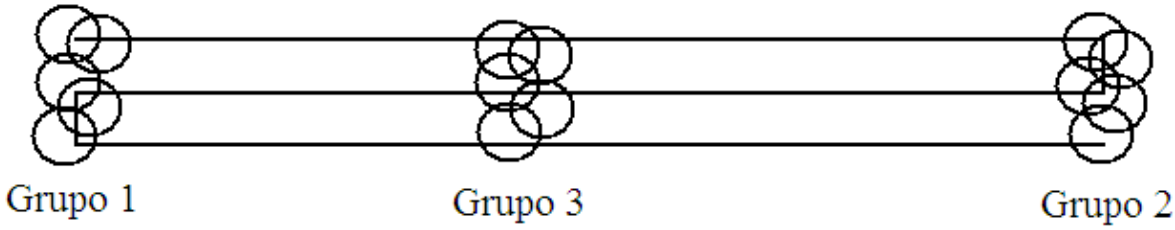
Figuras desconexas

As unidades da camada A de um perceptron têm campos receptivos que não abrangem toda a retina, mas partes dela (vejam a figura abaixo). O mesmo ocorre com os neurônios dos córtices visuais primários dos mamíferos.



Minsky e Papert mostraram em seu livro que um perceptron desse tipo não consegue decidir se uma figura é conexa ou desconexa.

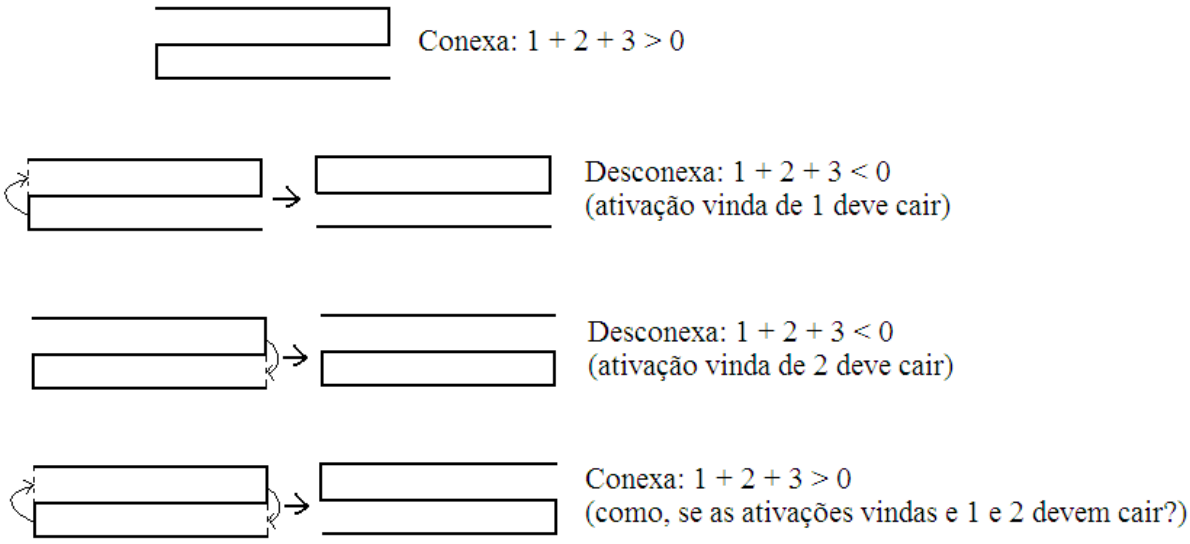
Considere uma figura como a do desenho abaixo, longa e fina. Há três tipos de unidades na camada A: as do grupo 1 “vêem” somente o lado esquerdo da figura; as do grupo 2 “vêem” somente o lado direito; a as do grupo 3 “vêem” somente o centro da figura.



Cada grupo de unidades computa funções *locais e simples* das suas entradas. Isto quer dizer que o nível de ativação calculado pela unidade da camada de saída do perceptron, cuja função é determinar se a figura é conexa ou não, tem três partes, cada uma devida a um dos três grupos. Vamos supor que a unidade da camada de saída tenha que dar a saída +1 se a figura for conexa e a saída 0 se a figura for desconexa.

Então, se uma figura conexa como a dada acima for apresentada à retina do perceptron, sua saída deverá ser +1. Para que isto ocorra, as ativações vindas dos três grupos de unidades da camada A devem ter soma maior ou igual a zero:

$$u = u_1 + u_2 + u_3 \geq 0.$$



A partir da figura conexa deste exemplo, podemos obter uma figura desconexa. Basta fazer como na primeira parte do desenho acima, deslocando a linha vertical da parte de baixo do lado esquerdo para a parte de cima. Se quisermos que o perceptron acerte este caso também, a sua unidade de saída vai ter que dar o valor 0. Para que isto aconteça o seu nível de ativação deverá ser negativo,

$$u = u_1 + u_2 + u_3 < 0.$$

Como a única coisa que mudou na figura foi uma propriedade do seu lado esquerdo, apenas as unidades do grupo 1 devem ter suas atividades alteradas neste caso. Isto implica que a mudança na barra do lado esquerdo deve ser suficiente para reduzir a contribuição u_1 a um ponto tal que a sua soma com as contribuições u_2 e u_3 torne-se negativa.

Pelo mesmo raciocínio, quando formamos uma figura desconexa como a da segunda parte do desenho acima a contribuição vinda das unidades do grupo 2, u_2, deve ser suficientemente reduzida para fazer que a sua soma com u_1 e u_3 torne-se negativa.

Por outro lado, quando fazemos as duas mudanças nas duas pontas da figura ao mesmo tempo, gerando uma nova figura conexa, a resposta do perceptron deverá ser +1. Porém, a mudança em uma das pontas só é notada pelas unidades que têm campo receptivo na ponta, de maneira que a mudança simultânea nas duas pontas deve ter um efeito sobre $u_1 + u_2$ igual ao que a mudança na ponta esquerda tem sobre u_1 isoladamente mais o que a mudança na ponta direita tem sobre u_2 isoladamente.

Isto implica que a mudança simultânea nas duas pontas deve reduzir $u_1 + u_2$, fazendo com que o nível de ativação da unidade de saída do perceptron fique negativo. Mas como isso é possível se a sua saída deve ser positiva?

Esta contradição mostra que se o perceptron classifica a primeira figura como conexa ele não consegue classificar a última também como conexa. Portanto, ele não é capaz de reconhecer padrões conexos do tipo do desenho, finos e compridos.

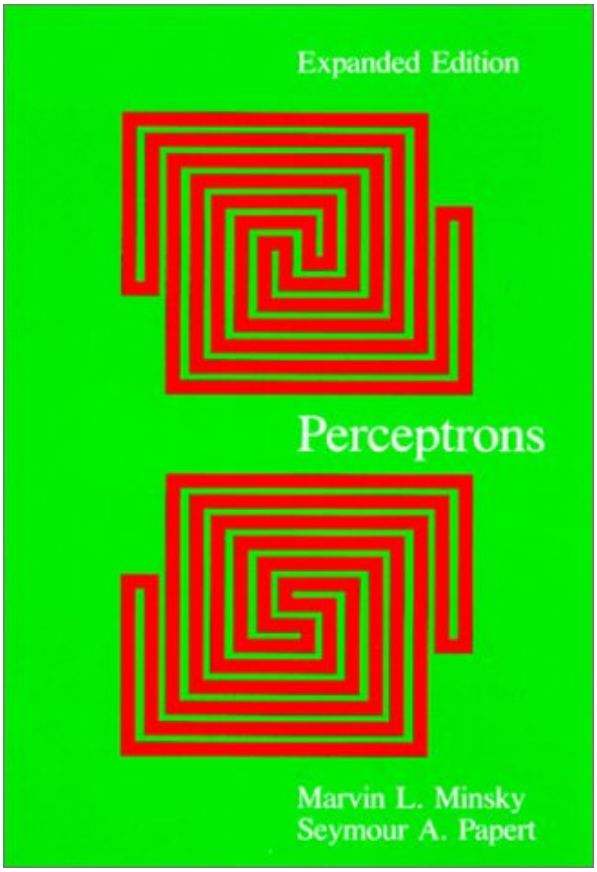
O problema da conectividade é análogo ao da função lógica XOU, pois não pode ser resolvido pelo perceptron. Porém, as pessoas e os animais também têm dificuldades em aprender a lógica do XOU e em reconhecer figuras conexas.

Quando se tenta ensinar a ratos uma tarefa de discriminação que é uma realização do XOU lógico, o que acontece é que eles aprendem o OU (função lógica ou-inclusivo) quase imediatamente. Portanto, eles acertam apenas 75% dos casos do XOU. Apenas após muitos treinos e com dificuldade é que os ratos elevam o seu percentual de acerto.

A identificação da conectividade de uma figura também apresenta dificuldades. Para figuras simples como as mostradas anteriormente, a conectividade é reconhecida facilmente.

Porém, para figuras mais complexas como as mostradas abaixo (cópia da capa da 3ª edição do livro *Perceptrons* de Minsky e Papert, 1988), uma das quais é conexa e a outra não, não é fácil perceber imediatamente qual é qual.

É necessário traçar os contornos mentalmente para verificar qual é a conexa e qual é a desconexa. Ou seja, é preciso que nos transformemos em uma "máquina serial" para resolver o problema.



Qual das duas figuras acima é a conexa?

Talvez os perceptrons não consigam determinar a conectividade de uma figura qualquer, mas os seres humanos também não, pelo menos nos estágios iniciais de percepção.

Isso mostra que os perceptrons, e as redes neurais em geral, podem possuir certas limitações que acabam por torná-los úteis como modelos para a cognição humana.

Uma diferença entre o tipo de aprendizado que ocorre nos seres vivos e o do perceptron é que o dos seres vivos nunca para, sempre podendo ser melhorado a partir de um dado ponto. Para os perceptrons, ou o aprendizado se encerra quando todos os padrões são classificados perfeitamente, ou nunca se atinge a resposta correta.

Para os seres vivos, dada uma tarefa de classificação, o tempo de resposta decresce continuamente mesmo depois que a resposta passa a ser a correta.