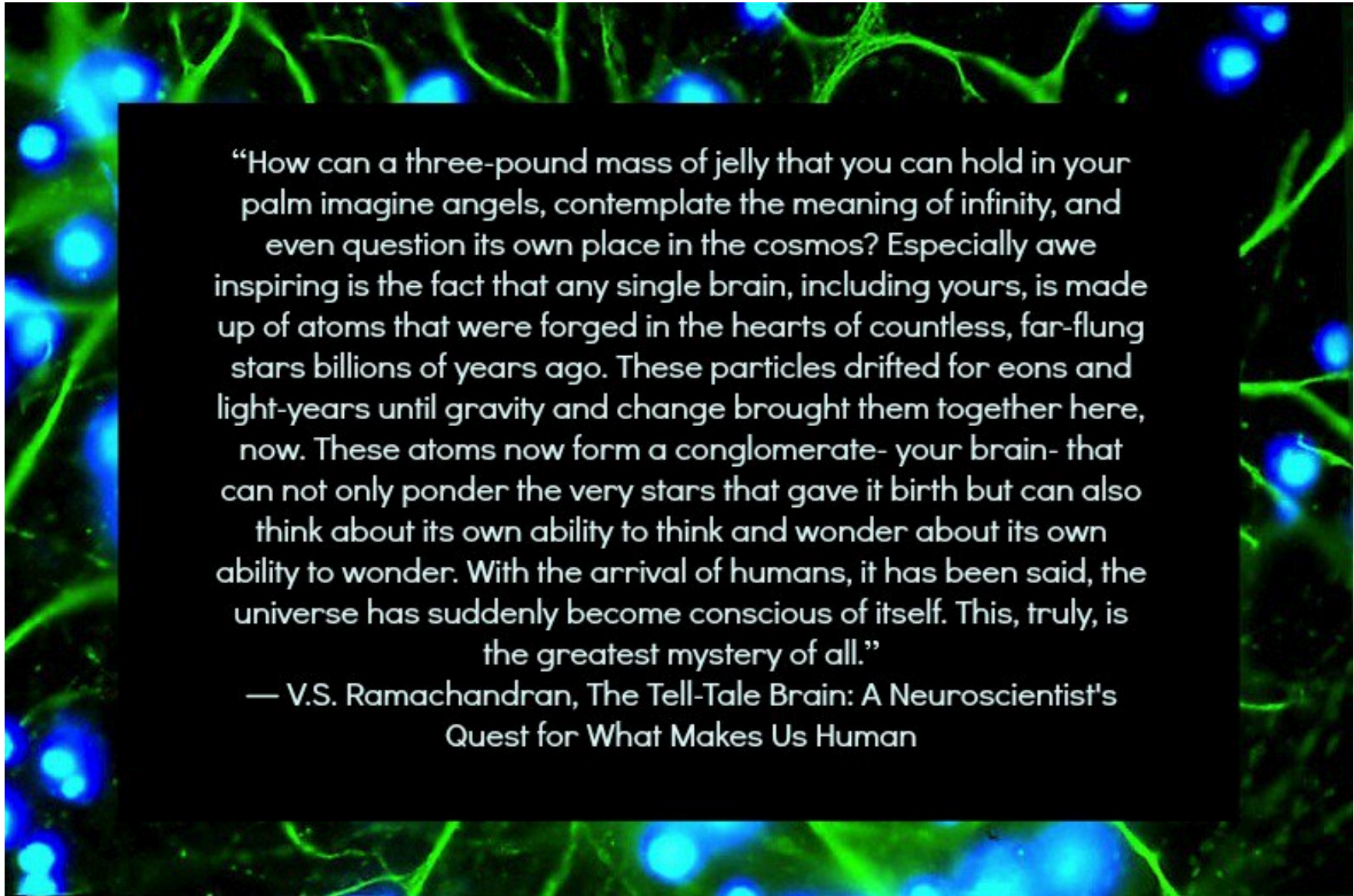


Redes Neurais Artificiais



“How can a three-pound mass of jelly that you can hold in your palm imagine angels, contemplate the meaning of infinity, and even question its own place in the cosmos? Especially awe inspiring is the fact that any single brain, including yours, is made up of atoms that were forged in the hearts of countless, far-flung stars billions of years ago. These particles drifted for eons and light-years until gravity and chance brought them together here, now. These atoms now form a conglomerate- your brain- that can not only ponder the very stars that gave it birth but can also think about its own ability to think and wonder about its own ability to wonder. With the arrival of humans, it has been said, the universe has suddenly become conscious of itself. This, truly, is the greatest mystery of all.”

— V.S. Ramachandran, *The Tell-Tale Brain: A Neuroscientist's Quest for What Makes Us Human*

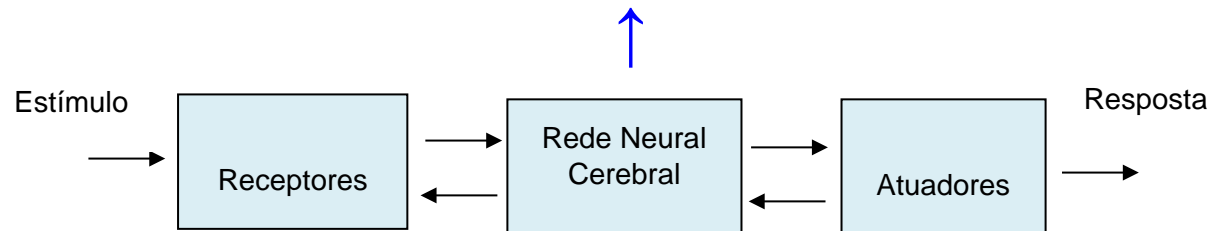
Analogia Neurobiológica

RNA – rede neural artificial



Estrutura computacional projetada para mimetizar a maneira pela qual o **cérebro** → opera de uma forma altamente complexa, não-linear e paralela desempenha uma particular tarefa de seu interesse

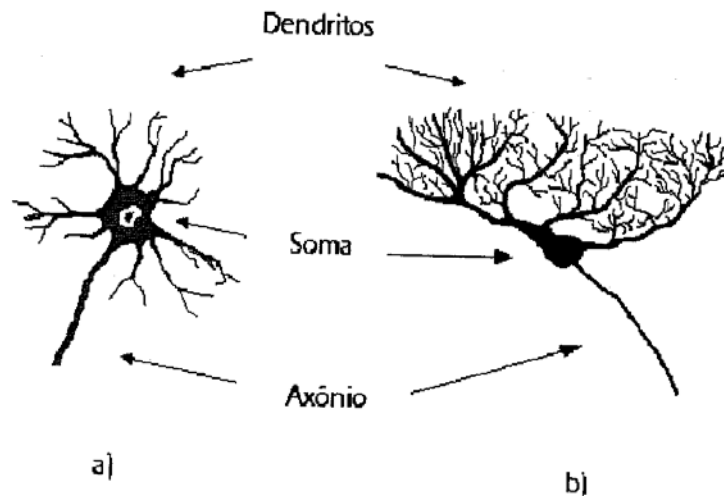
Recebe informações, as percebe e toma decisões apropriadas.



Representação do sistema nervoso em diagrama de blocos.

Neurônios

- Estruturas que constituem o cérebro (Ramón y Cajál, em 1911);
- Organizados de forma a desempenhar operações tais como reconhecimento de padrões, controle de movimento, etc... muitas vezes mais rápido do que o mais rápido computador digital existente;
- **5 a 6 ordens de grandeza mais lentos do que as portas lógicas de silício;**
 - eventos em um chip de silício acontecem na ordem de 10^{-9} s, enquanto que
 - eventos neurais acontecem na ordem de 10^{-3} s
- **Taxa de operação (relativamente lenta) de um neurônio é compensada por um inacreditavelmente grande número de neurônios, com densas interconexões entre eles** (≈ 10 bilhões de neurônios no córtex humano e 60 trilhões de sinapses ou conexões!)



Neurônios do sistema nervoso central dos vertebrados
a) cérebro anterior b) cérebro posterior

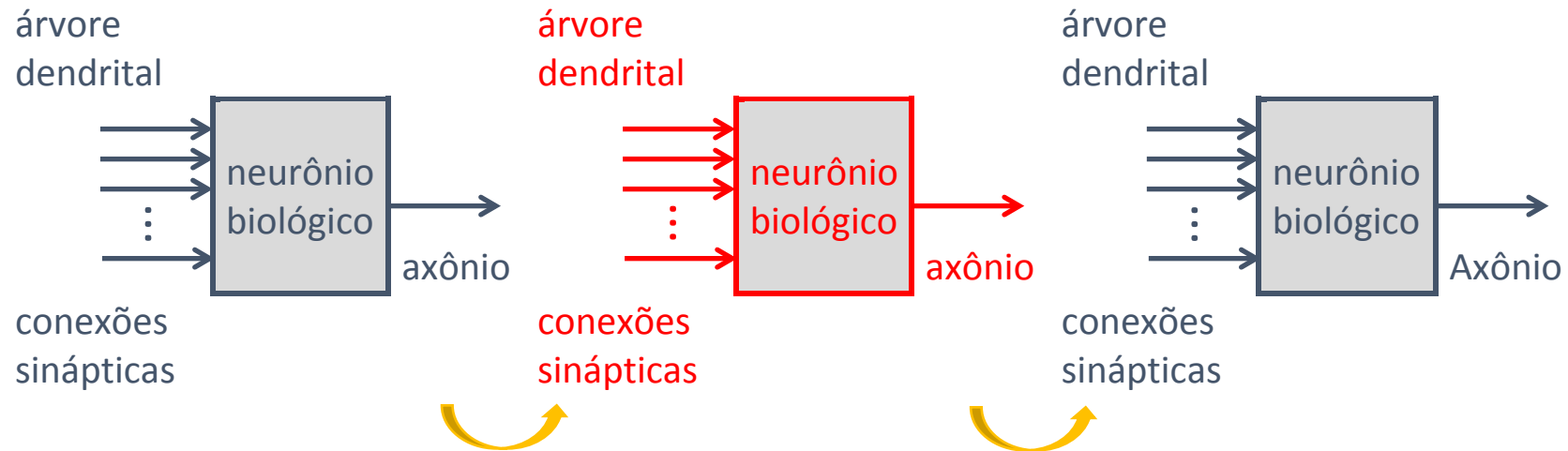
Delimitado por uma fina membrana celular que possui propriedades essenciais para o funcionamento elétrico da célula nervosa.

A partir do corpo celular (ou soma), que é o centro dos processos metabólicos da célula nervosa, projetam-se extensões filamentosas, que são os dendritos, e o axônio.

Dendritos são linhas de transmissão para os sinais de entrada de um neurônio e cobrem um volume muitas vezes maior do que o próprio corpo celular, formando uma árvore dendrital.

O **axônio** é uma linha de transmissão para o sinal de saída do neurônio e serve para conectar a célula nervosa a outras células do sistema nervoso.

Mecanismo computacional elementar do sistema nervoso

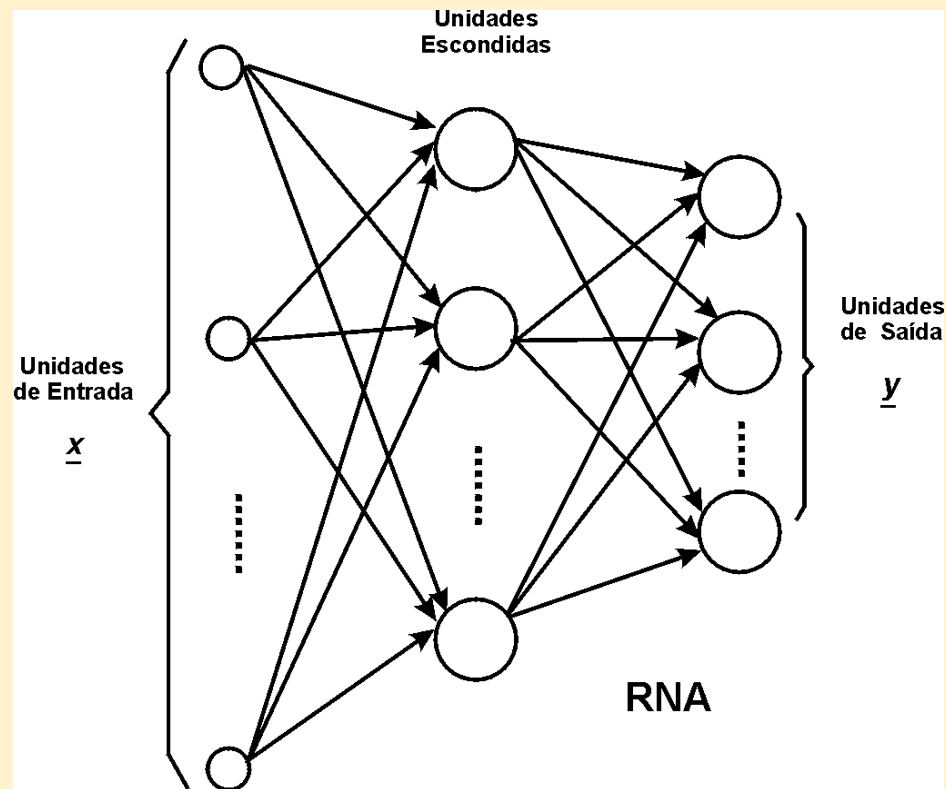


Pulsos elétricos conhecidos como **impulsos nervosos** ou **potenciais de ativação** (constituem a informação que o neurônio processará para produzir como saída um impulso nervoso no seu axônio)

Estes breves pulsos de tensão originam-se no próprio corpo celular do neurônio (ou próximo a ele) e então se propagam através dos neurônios individuais à velocidade e amplitude constantes

Neurotransmissores -> substâncias através das quais o estímulo nervoso que chega à sinapse é transferido à membrana dendritral

Arquitetura típica de uma Rede Neural Artificial

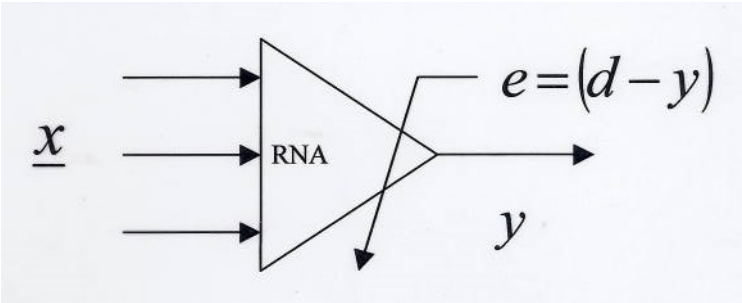


A exemplo das "redes neurais naturais",

- as redes artificiais consistem da interconexão de um grande número de unidades de processamento chamadas neurônios,
- as conexões entre as unidades computacionais (ou neurônios) são chamadas sinapses ou pesos sinápticos.

Processo de aprendizagem

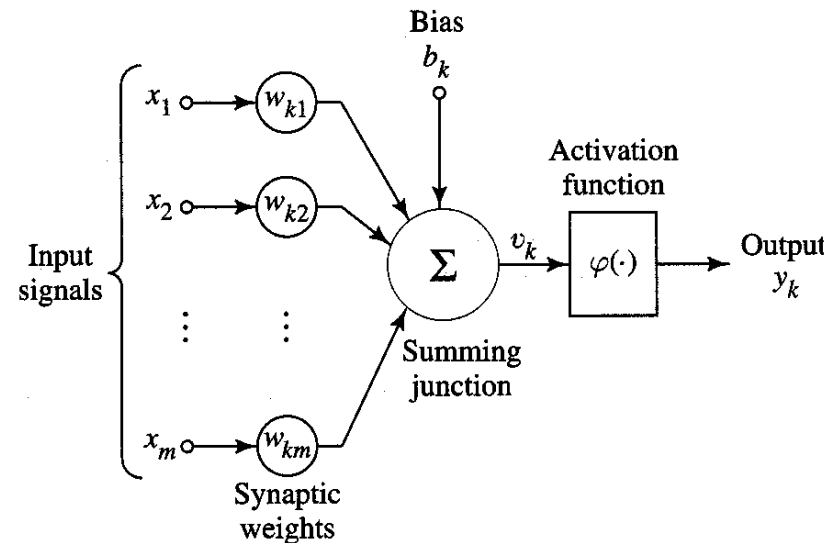
Processo de Aprendizagem	→ modo pelo qual as RNAs conseguem obter conhecimento a partir do ambiente.
Parâmetros Livres da Rede	→ provêm um mecanismo para armazenar o conteúdo de informação subjacente presente nos dados que são apresentados à rede na fase de treinamento. 1. Pesos sinápticos 2. Parâmetros que definem a função das unidades computacionais ou neurônios.
Algoritmo de Aprendizagem	→ procedimento utilizado para o processo de aprendizagem, o qual modifica de forma adaptativa os parâmetros livres da rede para atingir um objetivo desejado.



O diagrama mostra um bloco triangular rotulado 'RNA'. Três setas horizontais apontam para o lado esquerdo do bloco, com o símbolo \underline{x} à esquerda. Uma seta horizontal aponta para o lado direito do bloco, rotulada com y . Uma seta diagonal aponta para cima e para a direita, rotulada com a equação $e = (d - y)$.

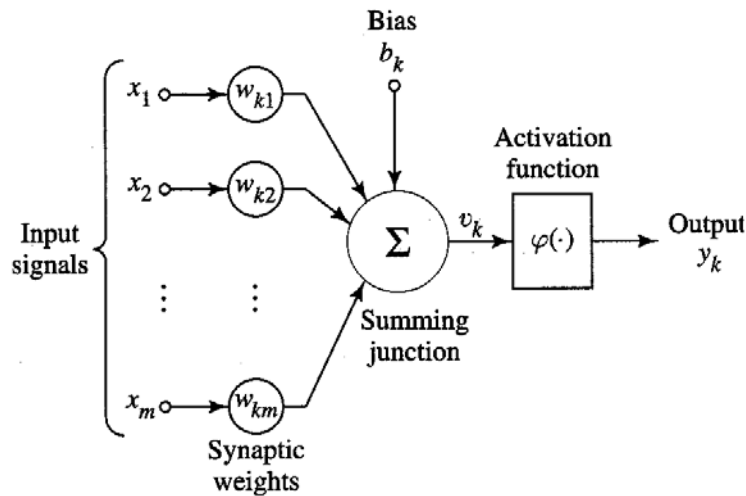
- Da mesma forma que em um filtro linear adaptativo convencional, as redes neurais artificiais têm a capacidade de, através da informação de uma resposta desejada, aproximar um sinal alvo durante o processo de aprendizagem.
- Esta aproximação é obtida através do ajuste, de forma sistemática, de um conjunto de parâmetros livres, característico de cada rede neural.

Modelo não-linear de um neurônio



- um conjunto de sinapses, cada uma delas caracterizada por um peso (=transmitância) característico;
- um combinador linear para somar os sinais de entrada, ponderados pela respectiva sinapse do neurônio;
- uma função de ativação para limitar a amplitude da saída do neurônio a algum valor finito. Tipicamente, a excursão da amplitude normalizada da saída de um neurônio é restrita ao intervalo unitário fechado $[0,1]$ ou, alternativamente $[-1,1]$;
- uma polarização externa (*bias*), denotada por b_k . A polarização b_k tem o efeito de aumentar ou diminuir o argumento da função de ativação, caso seja positivo ou negativo, respectivamente.

Descrição analítica do k -ésimo neurônio de uma rede de neurônios artificiais



$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (1) \quad \text{e} \quad y_k = \varphi(u_k + b_k) \quad (2)$$

onde:

x_1, x_2, \dots, x_m são os sinais de entrada;

$w_{k1}, w_{k2}, \dots, w_{km}$ são os pesos sinápticos do neurônio k ;

u_k é a saída do combinador linear devida aos sinais de entrada;

b_k é a polarização ou *bias*;

$\varphi(\cdot)$ é a função de ativação e

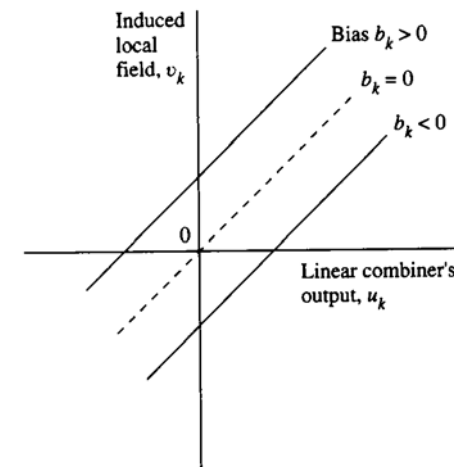
y_k é o sinal de saída do neurônio.

bias \rightarrow aplica uma transformação à saída u_k do combinador linear,

$$v_k = u_k + b_k \quad (3)$$

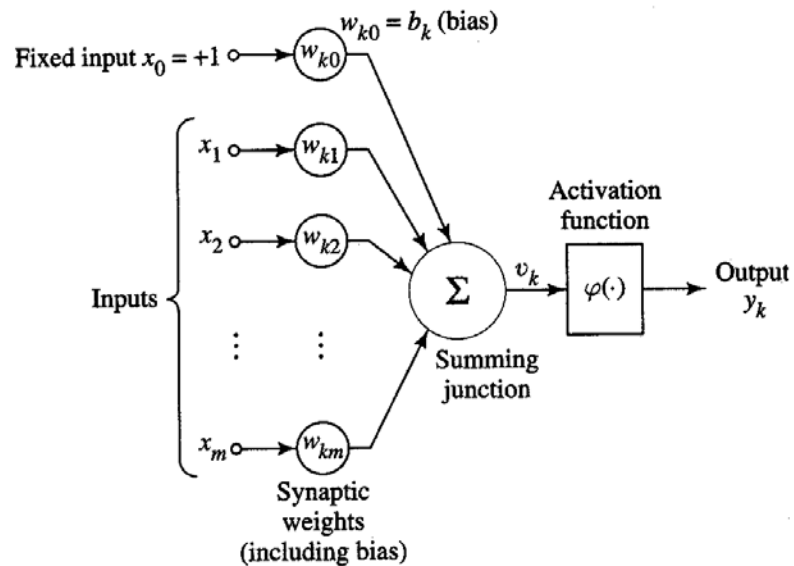
dependendo de $b_k > 0$ ou $b_k < 0$, a relação entre o potencial de ativação v_k do neurônio k e a saída do combinador linear u_k é conforme mostrada na figura ao lado.

Observe que, como resultado da transformação, o gráfico de $v_k \times u_k$ não passa mais pela origem.



\leftarrow Transformação produzida pela polarização ou *bias*
 ($v_k = b_k$
 para
 $u_k = 0$)

Modelo matematicamente equivalente



A polarização (*bias*) é um parâmetro externo do neurônio artificial k .

Outra forma de expressar a presença de *bias* é

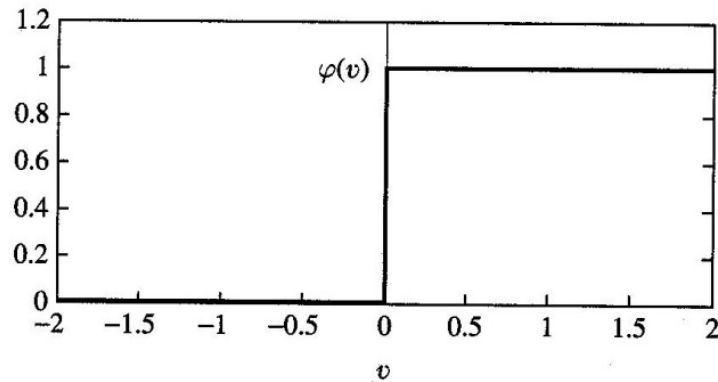
$$v_k = \sum_{j=0}^m w_{kj} x_j \quad (4) \quad \text{e} \quad y_k = \varphi(v_k) \quad (5)$$

Nova sinapse na eq. (4) $\rightarrow x_0 = +1$ e peso é $w_{k0} = b_k$.

A transmitância da **função de ativação** $\varphi(v)$ define a saída do neurônio em termos do potencial de ativação v .

Tipos de funções de ativação

1. Função Threshold (Limiar)



Modelo de McCulloch-Pitts

$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \quad [0,1]$$

$$y_k = \begin{cases} 1 & \text{se } v_k \geq 0 \\ 0 & \text{se } v_k < 0 \end{cases} \quad (\text{saída do neurônio } k)$$

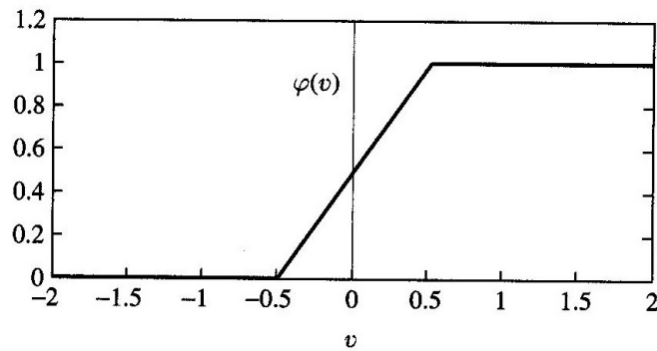
$$v_k = \sum_{j=1}^m w_{kj} x_j + b_k \quad (\text{potencial de ativação de } k)$$

Função Signum:

Threshold quando definida em $[-1,1]$

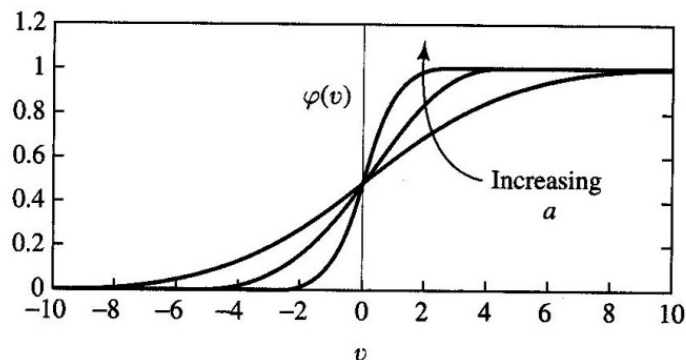
$$\varphi(v) = \begin{cases} 1 & \text{se } v > 0 \\ 0 & \text{se } v = 0 \\ -1 & \text{se } v < 0 \end{cases}$$

2. Função Piecewise-linear (Linear por partes)



$$\varphi(v) = \begin{cases} 1 & \text{se } v \geq +\frac{1}{2} \\ v + \frac{1}{2} & \text{se } +\frac{1}{2} > v > -\frac{1}{2} \\ 0 & \text{se } v \leq -\frac{1}{2} \end{cases} \quad [0,1]$$

3. Função Sigmóide



Função Logística:

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \quad [0,1]$$

Função Tangente Hiperbólica:

Sigmoidal quando definida em $[-1,1]$

$$\varphi(v) = \tanh(v)$$

RNAs vistas como Grafos de Fluxo de Sinal

- Grafo de fluxo de sinal: rede de ramos orientados que são interconectados a certos pontos chamados nós
- Um nó típico j tem um sinal de nó associado x_j
- Um típico ramo direcionado se origina no nó j , termina no nó k , e tem uma função de transferência associada (ou transmitância) que especifica a maneira pela qual o sinal y_k no nó k depende do sinal x_j no nó j .

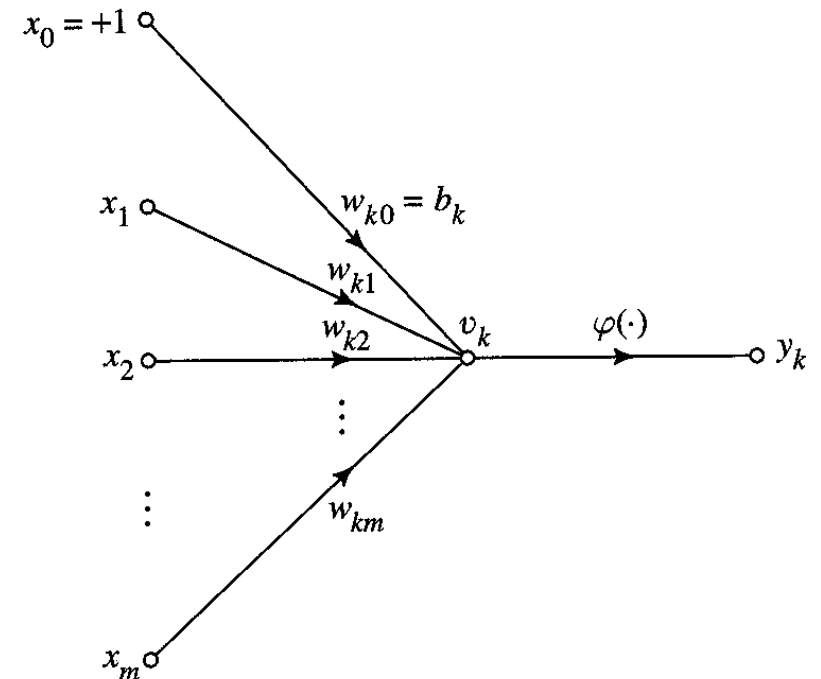
O fluxo de sinais nas várias partes do grafo é regido por três regras básicas:

1. Um sinal flui ao longo de um ramo somente na direção definida pela seta.

Ramos sinápticos: aqueles governados por uma relação linear entrada/saída, em que o sinal no nó x_j é multiplicado pelo peso sináptico w_{kj} para produzir o sinal no nó y_k e

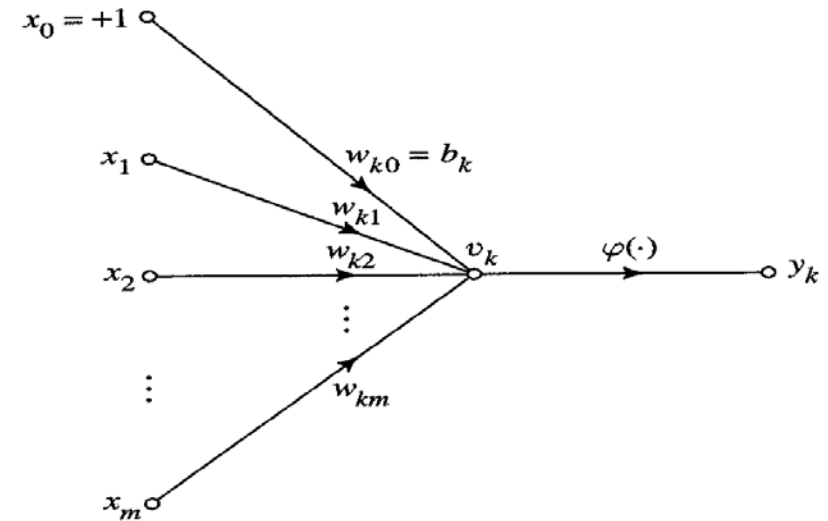
Ramos de ativação: aqueles governados, em geral, por uma relação não-linear entrada-saída.

2. Um sinal em um nó é igual à soma algébrica de todos os sinais que entram no nó, através dos ramos que chegam ao nó.
3. O sinal no nó é transmitido para cada ramo de saída originado no nó, com a transmissão sendo inteiramente independente das funções de transferência dos ramos que saem do nó.



Uma **rede neural** é um **grafo de fluxo de sinal orientado**, consistindo de **nós com interconexões sinápticas** e **ramos de ativação**, caracterizado por quatro propriedades:

1. Cada neurônio é representado por um conjunto de ramos sinápticos lineares, uma polarização externa aplicada, e um possível ramo não-linear de ativação.
2. A polarização é representada por um ramo sináptico conectado a uma entrada fixa em +1.
3. Os ramos sinápticos de um neurônio ponderam seus respectivos sinais de entrada.
4. A soma ponderada dos sinais de entrada define o potencial de ativação do neurônio em questão.
5. O ramo de ativação limita o potencial de ativação do neurônio para produzir uma saída.

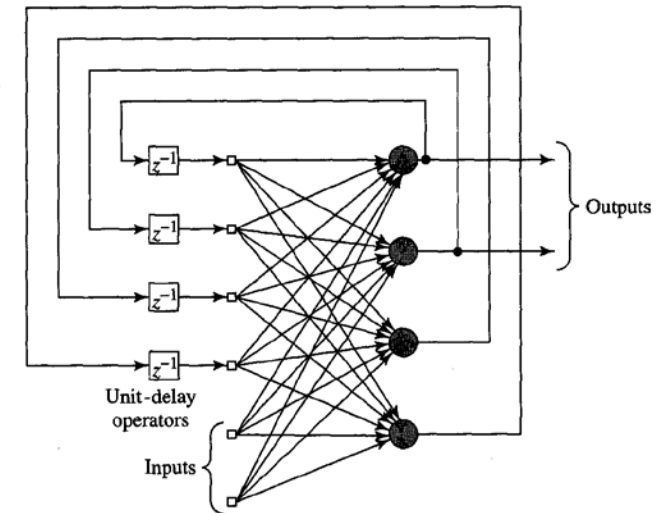
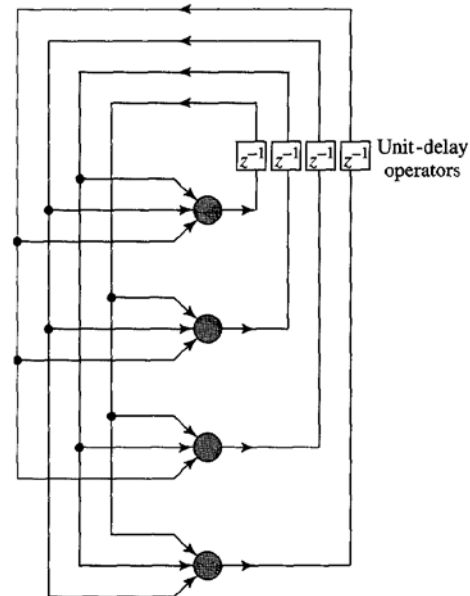
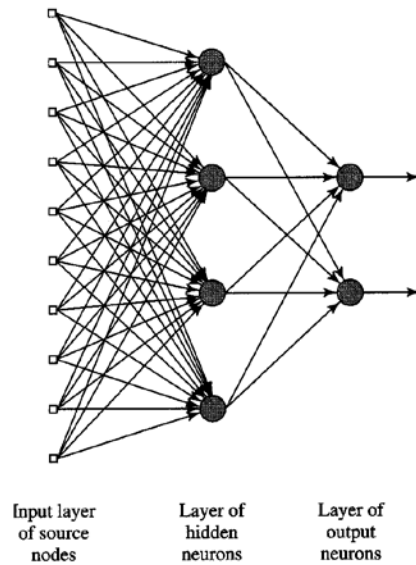
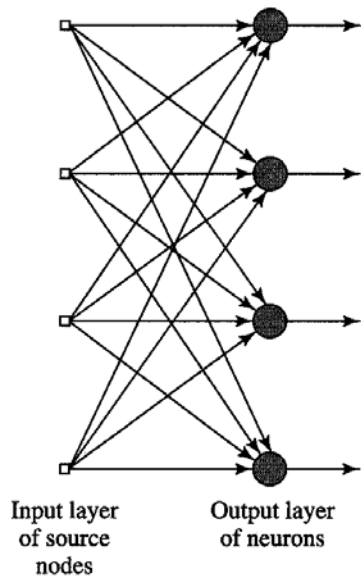


Arquiteturas de RNAs

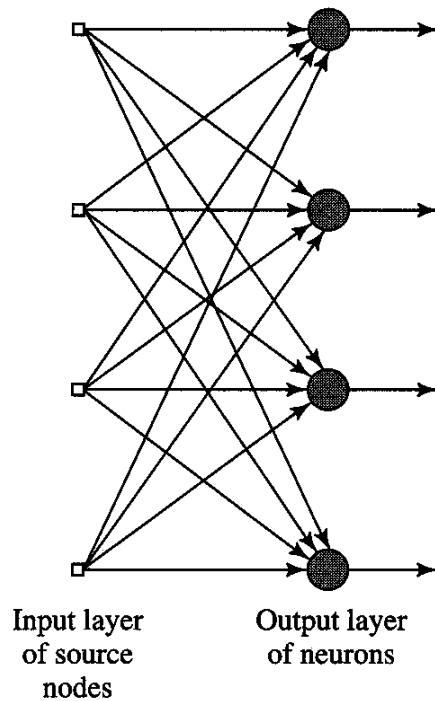
A forma pela qual os neurônios da rede são estruturados está intimamente relacionada ao algoritmo de aprendizagem usado para treinar a rede.

Classes fundamentais de arquiteturas de redes:

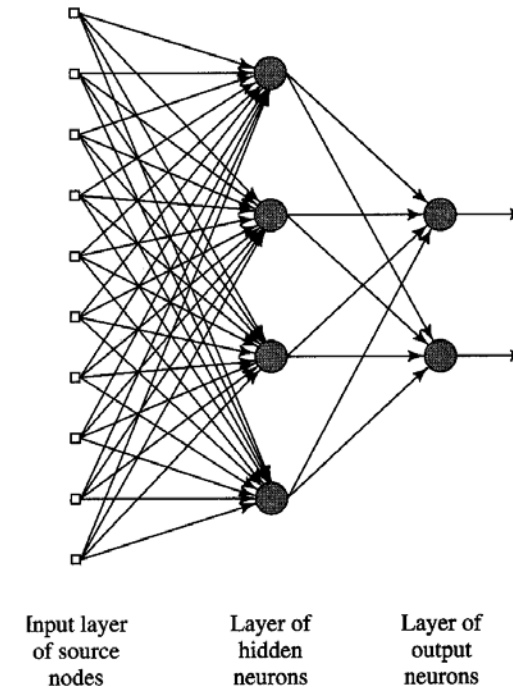
- Redes *Single-Layer Feedforward*
- Redes *Multilayer Feedforward*
- Redes Recorrentes



Redes Progressivas

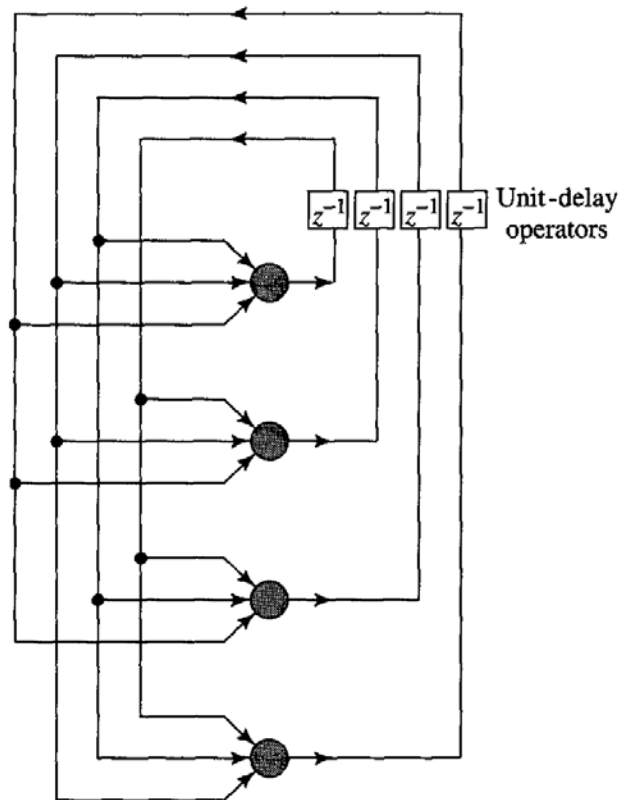


Rede progressiva (Single-Layer Feedforward) formada por uma única camada de neurônios (representada com quatro nós na camada de entrada e quatro neurônios na camada de saída).

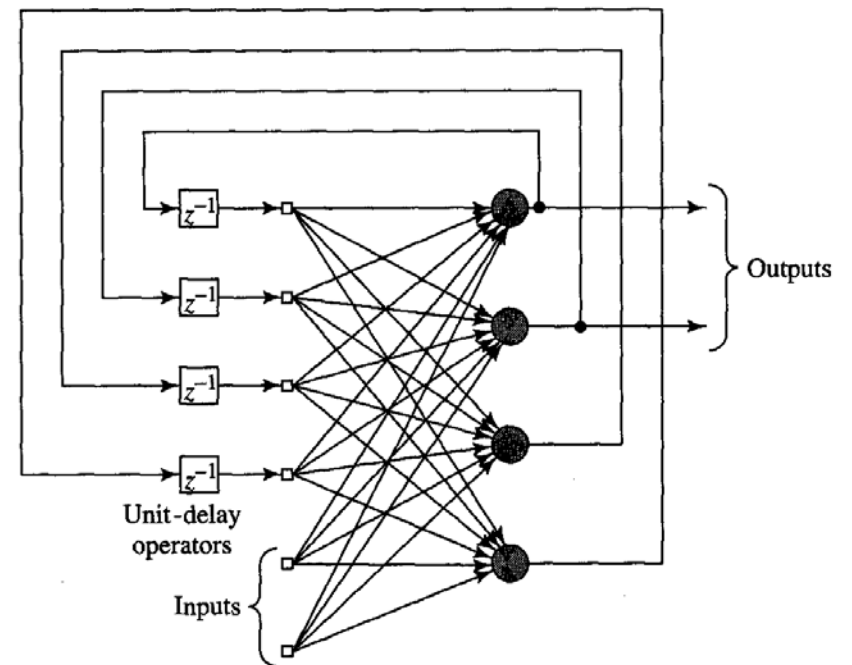


Rede progressiva multicamadas (Multilayer Feedforward) completamente conectada, formada por uma única camada escondida de neurônios e uma única camada de saída (representada com 10 nós fonte na camada de entrada, 4 neurônios escondidos e 2 neurônios na camada de saída).

Redes Recorrentes



Rede recorrente em que não há *loops* auto-realimentados (saída de cada neurônio conecta-se apenas às entradas dos demais), nem neurônios escondidos.



Rede recorrente com neurônios escondidos e *loops* auto-realimentados.

Algoritmos de aprendizagem

- Aprendizagem é um processo pelo qual os parâmetros livres de uma RNA são adaptados através da estimulação do ambiente no qual a rede está inserida.
- O tipo de aprendizagem é determinado pela forma através da qual é efetuada a mudança nos parâmetros.

S. Haykin

Esta definição implica em eventos sequenciais:

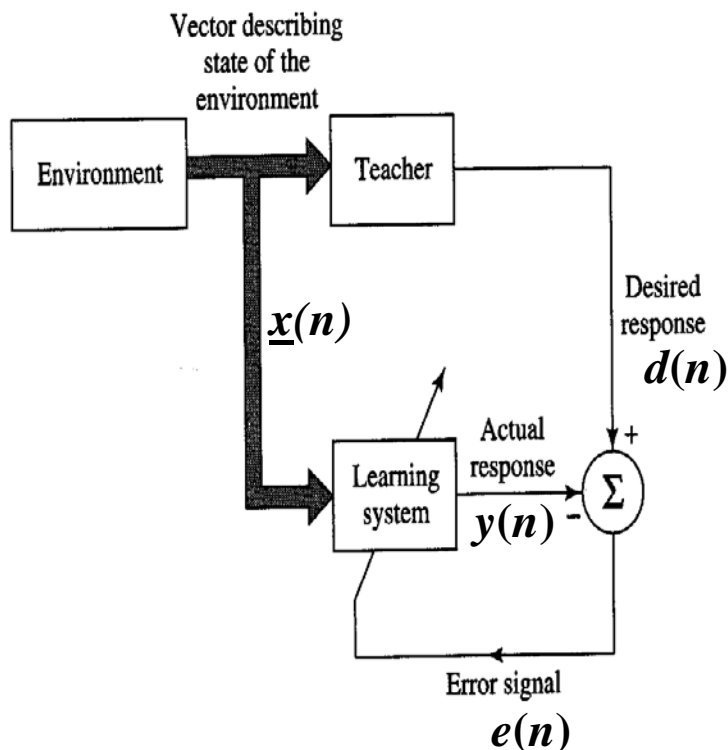
1. A rede é estimulada pelo ambiente;
2. A rede sofre mudanças nos seus parâmetros livres como resultado deste estímulo;
3. A rede responde de uma forma nova ao ambiente devido às mudanças que ocorreram em sua estrutura interna.

Um algoritmo de aprendizagem é um conjunto de regras definidas para a solução do problema de aprendizado.
(\neq características \neq vantagens)

Formas básicas de aprendizagem - paradigmas de aprendizagem

- através de um tutor (aprendizado supervisionado)
- sem um tutor (aprendizado não-supervisionado)
- com um crítico (ou juiz) (aprendizado por reforço)

Aprendizado Supervisionado



Conhecimento do Ambiente

Tutor: detém
RNA: desconhece

Representação do Conhecimento
{exemplos entrada/saída}

Construção do Conhecimento
(Transferido para a RNA,
durante o treinamento.)

RNA: exposta a vetor de treino extraído do ambiente

Tutor: provê à rede uma resposta desejada $d(n)$ para este específico vetor de treino $\underline{x}(n)$ a cada iteração n .

A parcela do conhecimento do ambiente disponível ao tutor é transferida para a RNA, durante o treinamento. Quando esta condição é atingida, o tutor é dispensado e a rede passa a lidar com o ambiente por si só.

Ajuste iterativo nos parâmetros da rede, através da influência combinada de $\underline{x}(n)$ e $e(n)$;

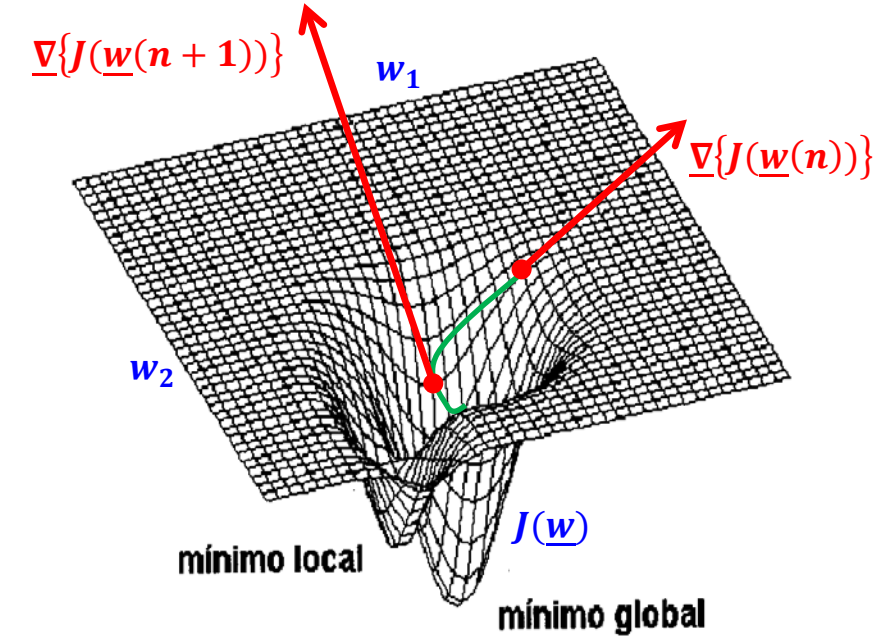
$$e(n) = d(n) - y(n)$$

$d(n)$: resultado ótimo que a rede deveria apresentar p/ $\underline{x}(n)$

Qualquer função analítica que expresse o quão distante da saída desejada $d(n)$ está a saída $y(n)$ determinada pelo processo de aprendizado é denominada de **função de custo** $J(\underline{w})$, cujo domínio é o vetor $\underline{w} = [w_1 \ \cdots \ w_m]^T$ que contém os m parâmetros livres do sistema. Uma função de custo usual é, por exemplo, o erro quadrático $e^2(n)$.

Função de Custo - minimização pela Regra Delta

Função de custo (=superfície de erro) $J(\underline{w})$: Mede o erro do processo de aprendizado em função dos parâmetros livres (no caso deste exemplo, $m = 2$ parâmetros: w_1 e w_2)



$\underline{w} = [w_1 \ w_2]^T$: parâmetros livres w_j (= coordenadas)

- A função de custo ou superfície de erro $J(\underline{w})$ é uma função dos m parâmetros livres w_j , $1 \leq j \leq m$, e é utilizada para avaliar o desempenho do algoritmo de aprendizagem a cada iteração n .

- A cada iteração n do processo de aprendizagem, o grau de “desconhecimento” sobre o mapeamento que está sendo aprendido sob a supervisão do tutor é representada como um ponto sobre a superfície de erro $J(\underline{w}(n))$ na coordenada $\underline{w}(n) = [w_1(n) \ w_2(n)]^T$. O objetivo do processo de aprendizado é ajustar iterativamente, a cada iteração n , as coordenadas $\underline{w}(n) = [w_1(n) \ w_2(n)]^T$ em direção ao ponto de mínimo da superfície de erro $J(\underline{w}(n))$.

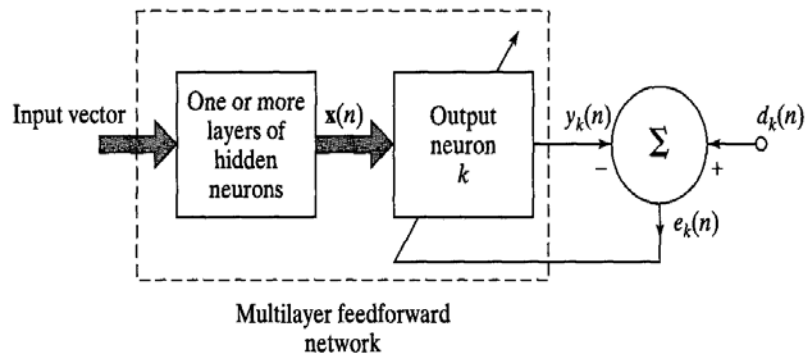
- Para qualquer iteração, o gradiente $\nabla\{J(\underline{w})\}$ da superfície de erro $J(\underline{w})$ calculado em uma dada coordenada $\underline{w} = [w_1 \ w_2]^T$ é o vetor que, partindo desta coordenada, aponta na direção de subida mais íngreme (vide figura ao lado).

- O algoritmo de aprendizado supervisionado executa iterativamente, a cada iteração n , o processo de descida em direção ao ponto de mínimo (*steepest descent*) da superfície $J(\underline{w})$ ajustando $\underline{w}(n) = [w_1(n) \ w_2(n)]^T$ na direção contrária do gradiente $\nabla\{J(\underline{w}(n))\}$, de acordo com a **Regra Delta**, conforme abaixo.

$$\underline{w}(n+1) = \underline{w}(n) - \eta \nabla\{J(n)\} \Rightarrow \text{Regra Delta}$$

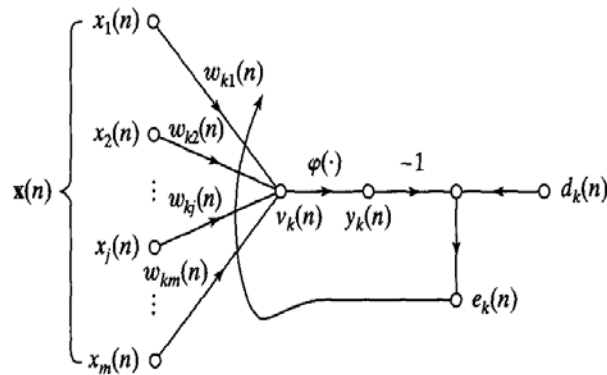
onde $0 < \eta < 1.0$ é a razão de aprendizagem e $\nabla\{J\} = \frac{\partial J}{\partial \underline{w}}$ é o gradiente da superfície de erro (=função de custo).

(= aprendizado por correção de erro)



Multilayer feedforward network

(a) Block diagram of a neural network, highlighting the only neuron in the output layer



(b) Signal-flow graph of output neuron

A minimização da função de custo quadrática

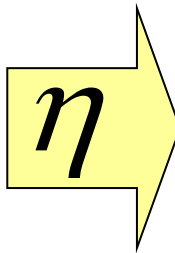
$$J(n) = \frac{1}{2} e_k^2(n) \text{ com } e_k(n) = d_k(n) - y_k(n)$$

através da Regra Delta conduz à Regra de Widrow-Hoff:

"O ajuste feito a um peso sináptico de um neurônio linear ($\varphi(\cdot) = 1.0$) é proporcional ao produto do sinal de erro pelo sinal de entrada da sinapse em questão."

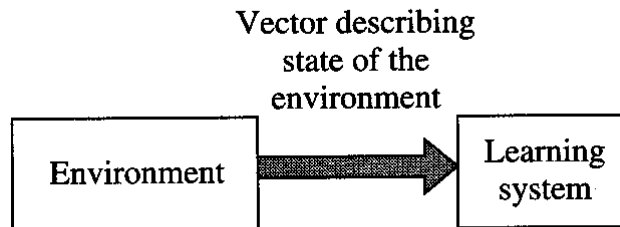
$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n)$$

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n)$$



- Razão de aprendizagem – expressa o tamanho do passo dado à cada iteração n à medida que o evolui o processo de minimização de $J(n)$ (processo de aprendizagem);
- Constante $\eta > 0$;
- Seleção cuidadosa de η necessária para a estabilidade ou convergência do processo de aprendizagem iterativo.

Aprendizado Não-Supervisionado ou Auto-organizado



- Não há um tutor externo ou crítico para supervisionar o processo de aprendizado.
- Não requer conhecimento de saídas desejadas.
- Não são utilizados exemplos “entradas”/”saída desejada” a serem aprendidos pela rede.

1. Padrões de entrada são apresentados à RNA até que se torne "sintonizada" às regularidades estatísticas dos dados de entrada.
2. Desenvolve a habilidade de formar representações internas para codificar características da entrada (por exemplo, adapta os pesos de suas conexões para representar os padrões de entrada).
3. Agrupa os padrões de entrada em grupos com características similares ou cria novos grupos automaticamente.

O aprendizado Hebbiano (um dos mais importantes aprendizados não-supervisionados)

Quando um axônio de uma célula A está próximo o suficiente para excitar uma célula B e repetidamente ou persistentemente participa de sua ativação, algum processo de crescimento ou alteração metabólica acontece em uma ou ambas as células, tal que a eficiência de A como uma das células que ativa B é aumentada.

Postulado de Hebb (neurobiologista)

Expandindo o postulado de Hebb em duas regras:

1. Se dois neurônios em cada um dos lados da sinapse (conexão) são ativados simultaneamente (i. é, sincronamente), então a transmitância daquela sinapse é seletivamente aumentada.
2. Se dois neurônios em cada um dos lados da sinapse são ativados de forma assíncrona, então a transmitância daquela sinapse é seletivamente enfraquecida ou eliminada.

Formulação Matemática do Aprendizado Hebbiano

Consideremos a transmitância w_{kj} da sinapse j do neurônio k (= peso sináptico w_{kj}) com sinais pré-sinápticos e pós-sinápticos denotados por x_j e y_k , respectivamente.

O ajuste aplicado ao peso sináptico w_{kj} no intervalo de tempo n é expresso na forma geral

$$\Delta w_{kj}(n) = F(y_k(n), x_j(n)) \quad (1)$$

onde $F(\cdot, \cdot)$ é uma função de ambos sinais pré-sinápticos e pós-sinápticos.

Há duas formulações Hebbianas básicas para a expressão (1):

A primeira delas é baseada na **hipótese de Hebb** e a segunda, na **hipótese da covariância**.

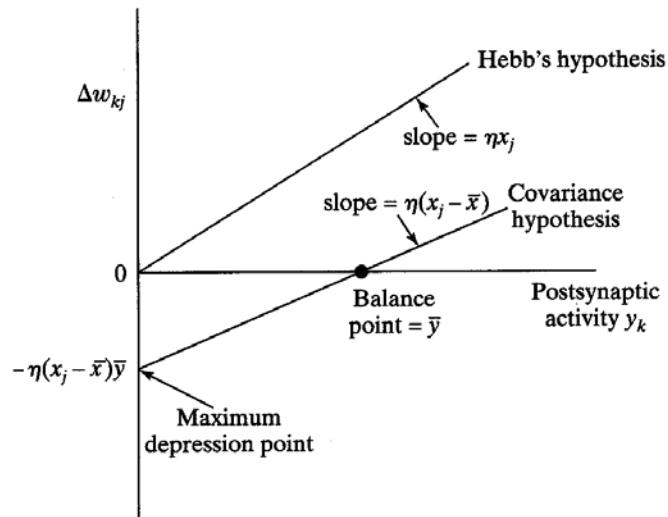
a) Hipótese de Hebb

A forma mais simples de aprendizado Hebbiano é descrita por

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n) \quad (2)$$

onde o parâmetro η é uma constante positiva que determina a razão de aprendizado.

A equação (2) enfatiza a natureza **correlacional** da sinapse Hebbiana.



A curva superior da figura ao lado mostra uma representação gráfica da Equação (2) com o ajuste no peso $\Delta w_{kj}(n)$ plotado versus o sinal de saída (atividade pós-sináptica) y_k .

Pode ser observado que a aplicação repetida do sinal de entrada (atividade pré-sináptica) x_j leva a um aumento em y_k e, portanto, a um crescimento exponencial que leva a transmitância da sinapse à saturação, ponto em que nenhuma informação será armazenada na sinapse e a capacidade de discriminar (filtrar) a informação de entrada é perdida.

Ilustração da Hipótese de Hebb e da Hipótese da covariância.

b) Hipótese da Covariância

Sejam \bar{x} e \bar{y} os valores médios sobre um determinado intervalo de tempo dos sinais pré e pós-sinápticos x_j e y_k , respectivamente. De acordo com a hipótese da covariância, o ajuste aplicado à sinapse w_{kj} é definido por

$$\Delta w_{kj} = \eta(x_j - \bar{x})(y_k - \bar{y}) \quad (3)$$

onde η é o parâmetro razão de aprendizado. Os valores médios \bar{x} e \bar{y} constituem limiares pré e pós-sinápticos, os quais determinam o sinal do ajuste aplicado à sinapse (ver figura).

Note da Equação (3) que:

1. A transmitância da sinapse w_{kj} é aumentada se há níveis suficientes de atividade pré-sináptica e pós-sináptica; ou seja, as condições $x_j > \bar{x}$ e $y_k > \bar{y}$ são ambas satisfeitas.
2. A transmitância da sinapse w_{kj} é diminuída se ocorre ativação pré-sináptica (isto é, $x_j > \bar{x}$) e simultaneamente ocorre insuficiente ativação pós-sináptica (isto é, $y_k < \bar{y}$). Alternativamente, a transmitância da sinapse w_{kj} é diminuída se ocorre ativação pós-sináptica (i. é, $y_k > \bar{y}$) e simultaneamente ocorre insuficiente ativação pré-sináptica (i. é, $x_j < \bar{x}$).

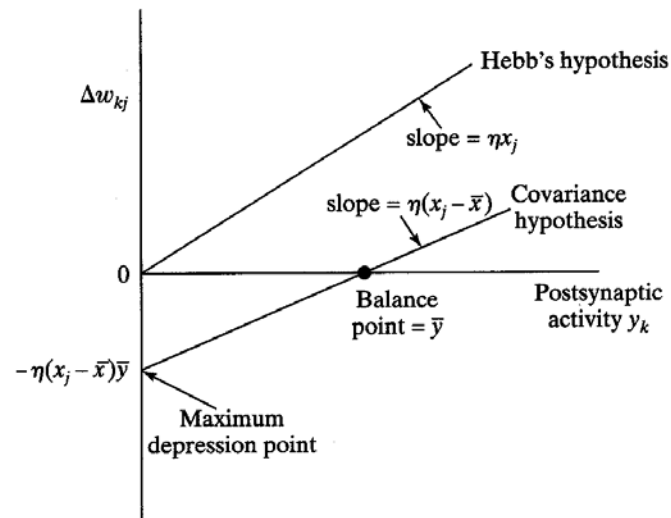
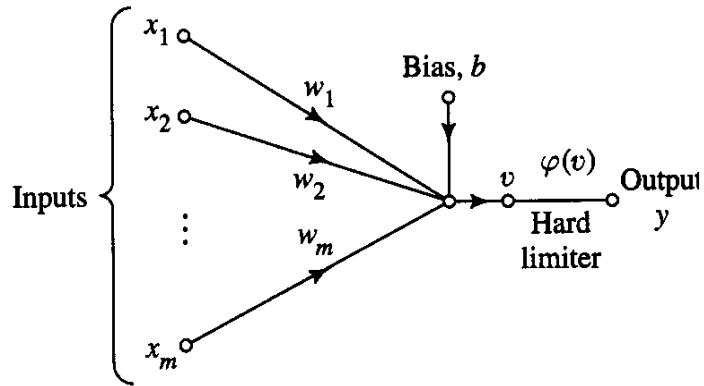


Ilustração da Hipótese de Hebb e da Hipótese da covariância.

O Perceptron – um classificador baseado em aprendizado supervisionado

Perceptron \Rightarrow neurônio não-linear (modelo de McCulloch-Pitts).



← Grafo de fluxo de sinal do Perceptron

- Pesos sinápticos do Perceptron : w_1, w_2, \dots, w_m
- Entradas aplicadas ao Perceptron: x_1, x_2, \dots, x_m
- Polarização ou *bias* (aplicada externamente): b

• Nó somador
(combinador linear)
 \Downarrow

{ entradas aplicadas às sinapses +
pesos sinápticos associados +
polarização externamente aplicada.

Soma resultante \Rightarrow Potencial de Ativação \Rightarrow aplicado a um hard limiter $\varphi(v)$.

$$v = \sum_{i=1}^m w_i x_i + b$$

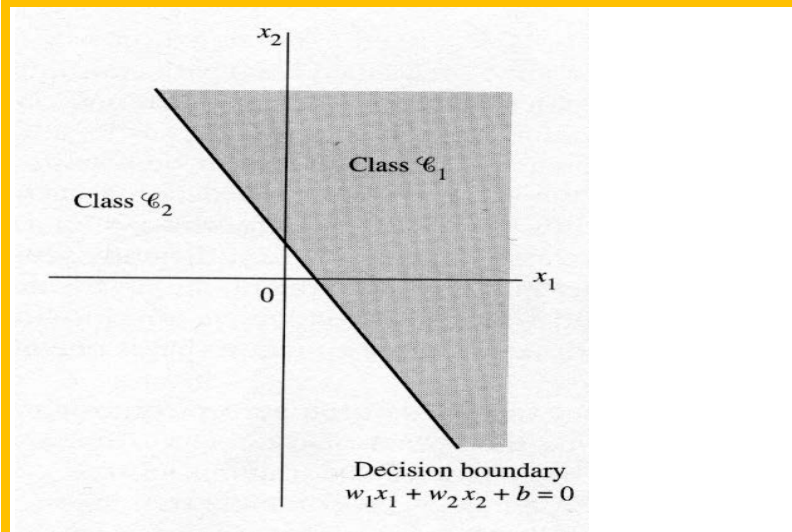
• Limitador $\varphi(v) \Rightarrow$ implementa a função signum \Downarrow

Neurônio produz
uma saída igual a: { (+1) se a entrada do limitador é positiva
(-1) se é negativa

Objetivo do Perceptron:	Classificar corretamente o conjunto de estímulos externos aplicados x_1, x_2, \dots, x_m em uma de duas classes, C_1 ou C_2 .
Regra de decisão:	Atribuir o ponto representado pelas entradas x_1, x_2, \dots, x_m à classe $\begin{cases} C_1 & \text{se a saída do Perceptron for } +1 \\ C_2 & \text{se a saída do Perceptron for } -1 \end{cases}$

Na forma mais simples do Perceptron há duas **regiões de decisão separadas por um hiperplano definido**

por $\sum_{i=1}^m w_i x_i + b = 0$ conforme ilustrado na figura abaixo, para o caso de duas variáveis de entrada x_1 e x_2 , para as quais o limite de decisão assume a forma de uma linha reta.



- Mapa das regiões de decisão no espaço de sinal m -dimensional gerado pelas m variáveis de entrada x_1, x_2, \dots, x_m .
- Ilustração do hiperplano (neste caso, uma linha reta) como limite de decisão para um problema de classificação de padrões de duas classes (bi-dimensional).

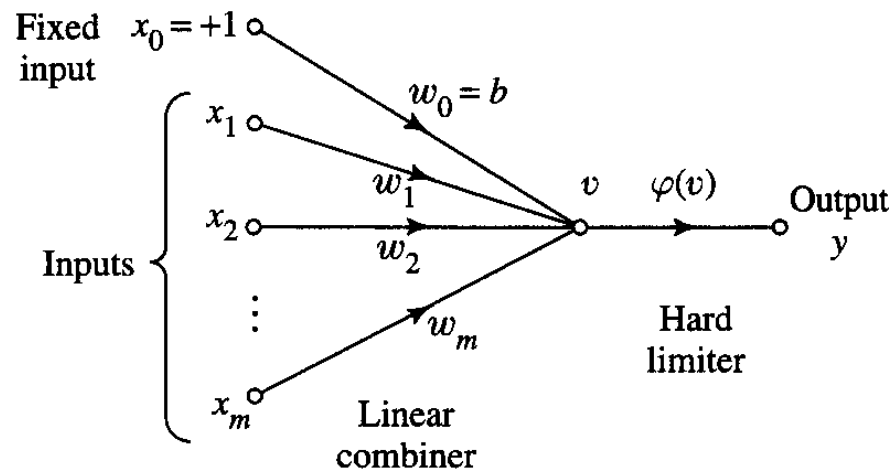
Ponto (x_1, x_2) :
 acima da linha limítrofe é atribuído à classe C_1 ; abaixo da linha limítrofe é atribuído à classe C_2 .

O processo de aprendizagem do Perceptron

Os pesos sinápticos w_1, w_2, \dots, w_m do Perceptron são adaptados de iteração a iteração n .

Para a adaptação, adota-se a regra de correção de erro conhecida como algoritmo de convergência do Perceptron, baseado no modelo do grafo de fluxo de sinal modificado mostrado na figura abaixo.

Neste modelo, o *bias* $b(n)$ é tratado como um peso sináptico cuja entrada é fixa em +1.

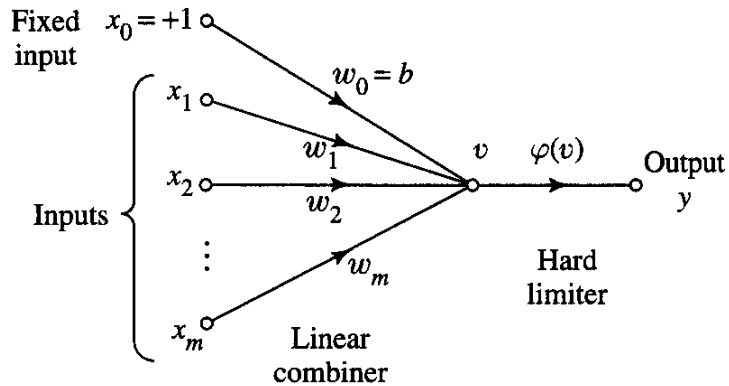


Grafo de fluxo de sinal equivalente do Perceptron (a dependência do tempo foi omitida por questões de clareza).

Note que a saída $v(n) = \sum_{i=0}^m w_i(n) x_i(n) = \underline{w}^T(n) \underline{x}(n)$ é a projeção (produto escalar) do vetor

$\underline{x}(n) = [+1 \ x_1(n) \ x_2(n) \ \dots \ x_m(n)]^T$ sobre o vetor de pesos sinápticos $\underline{w}(n) = [b(n) \ w_1(n) \ w_2(n) \ \dots \ w_m(n)]^T$, e, portanto, expressa o quão o vetor dos estímulos de entrada $\underline{x}(n)$ correlaciona-se com o vetor do conjunto de sinapses $\underline{w}(n)$.

Como $v(n)$ expressa o **nível de correlação** entre a entrada $\underline{x}(n)$ e o conjunto de sinapses $\underline{w}(n)$, então, ao submeter $v(n)$ à transmitância tipo *hard limiter* da função de ativação $\varphi(v)$, **que mimetiza a função de ativação de um neurônio biológico**, infere-se que a saída $y(n) = \varphi(v(n))$ expressa uma decisão sobre este nível de correlação na iteração n .



- **Vetor de entrada** $[(m+1) \times 1]$ - dimensional:

$$\underline{x}(n) = [+1 \ x_1(n) \ x_2(n) \ \dots \ x_m(n)]^T$$

- **Vetor de pesos** $[(m+1) \times 1]$ -dimensional :

$$\underline{w}(n) = [b(n) \ w_1(n) \ w_2(n) \ \dots \ w_m(n)]^T$$

- **Saída do combinador linear:**

$$v(n) = \sum_{i=0}^m w_i(n) x_i(n) = \underline{w}^T(n) \underline{x}(n)$$

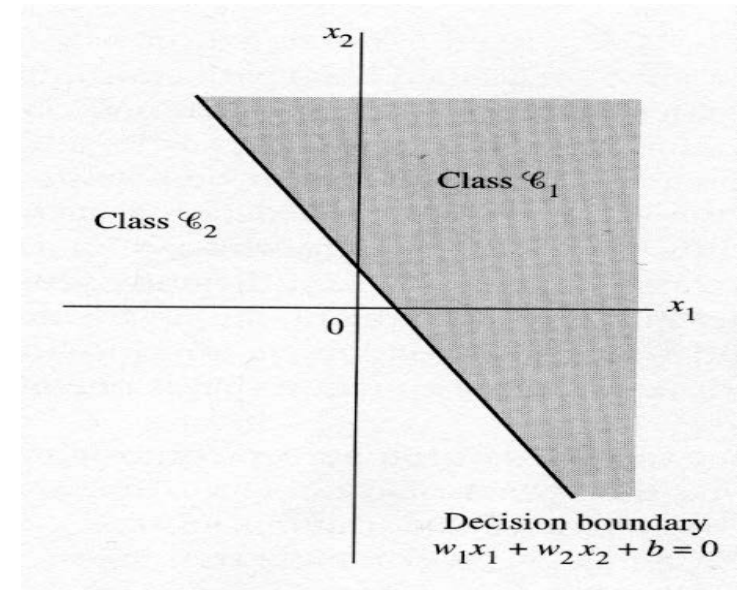
E como $\underline{w}(n)$ são parâmetros livres ajustáveis e afins à $\underline{x}(n)$ – **afins porque são correlacionáveis entre si** – e como $\underline{w}(n)$ afeta a saída $y(n) = \varphi(v(n))$, infere-se que um tutor do tipo

$$d(n) = \begin{cases} +1 & \text{se } \underline{x}(n) \text{ pertence à classe } C_1 \\ -1 & \text{se } \underline{x}(n) \text{ pertence à classe } C_2 \end{cases} \quad \text{pode ser usado para ajustar}$$

$\underline{w}(n)$ através do erro instantâneo $e(n) = d(n) - y(n)$, que, em função de $\underline{w}(n)$ e $\underline{x}(n)$ serem afins, sugere a o ajuste

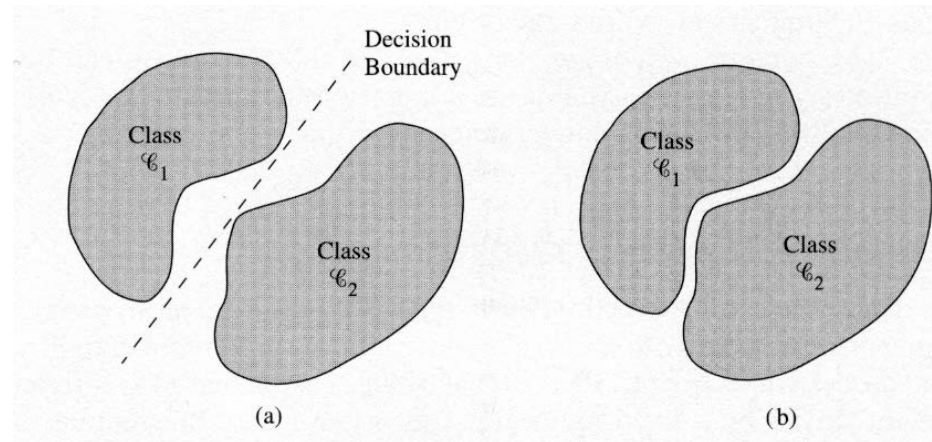
$$\Delta \underline{w}(n) = \eta e(n) \underline{x}(n) = \eta [d(n) - y(n)] \underline{x}(n)$$

onde $0.0 < \eta < 1.0$ é a razão de aprendizagem ou passo de adaptação.



Nota: Para que o Perceptron funcione adequadamente, as duas classes C_1 e C_2 precisam ser linearmente separáveis, o que significa dizer que os padrões a serem classificados devem ser suficientemente separados uns dos outros para garantir que a superfície de decisão consista de um hiperplano.

- (a) Um par de padrões linearmente separáveis.
- (b) Um par de padrões não-linearmente separáveis.



Na Figura (a), as duas classes C_1 e C_2 são suficientemente separáveis uma da outra, de tal forma que é possível desenhar um hiperplano (neste caso uma linha reta) como limite de decisão.

Se, entretanto, as duas classes C_1 e C_2 tivessem se aproximado tanto uma da outra (como mostrado na Figura (b)) teriam se tornado não-linearmente separáveis, uma situação que está além da capacidade computacional do Perceptron.

Algoritmo de Convergência do Perceptron

Variáveis e Parâmetros:

Vetor de entrada $\underline{x}(n)$ de dimensão $[(m+1) \times 1]$; $\underline{x}(n) = [+1 \ x_1(n) \ x_2(n) \ \cdots \ x_m(n)]^T$

Vetor de pesos sinápticos $\underline{w}(n)$ de dimensão $[(m+1) \times 1]$; $\underline{w}(n) = [b(n) \ w_1(n) \ w_2(n) \ \cdots \ w_m(n)]^T$

Bias = $b(n)$

Resposta desejada = $d(n)$

Resposta atual (quantizada) = $y(n)$

Razão de aprendizado (constante positiva < 1) = η

1. Inicialização: Fazer $\underline{w}(0) = \underline{0}$. Executar as etapas seguintes do algoritmo para os instantes de tempo $n = 1, 2, \dots$

2. Ativação: No instante de tempo n ativar o Perceptron aplicando o vetor de entrada $\underline{x}(n)$ e a resposta desejada $d(n)$.

3. Cômputo da Resposta Atual: Computar a resposta atual do Perceptron através de

$$y(n) = \text{sgn}(\underline{w}^T(n) \underline{x}(n)), \text{ onde } \text{sgn}(\cdot) \text{ é a função signum; } \text{sgn}(x) = \begin{cases} +1 & \text{se } x \geq 0 \\ -1 & \text{se } x < 0 \end{cases}$$

4. Adaptação do Vetor de Pesos Sinápticos: Atualizar as sinapses do Perceptron através de

$$\underline{w}(n+1) = \underline{w}(n) + \eta [d(n) - y(n)] \underline{x}(n) \text{ onde } d(n) = \begin{cases} +1 & \text{se } \underline{x}(n) \text{ pertence à classe } C_1 \\ -1 & \text{se } \underline{x}(n) \text{ pertence à classe } C_2 \end{cases}$$

5. Continuação: Fazer $n = n + 1$ e voltar à etapa 2.

Note que se o erro $e(n) = d(n) - y(n)$ é zero o vetor $\underline{w}(n)$ não é atualizado:

$\underline{w}(n+1) = \underline{w}(n) + \eta[d(n) - y(n)]\underline{x}(n)$			
$d(n) = \begin{cases} +1 & \text{se } \underline{x}(n) \in C_1 \\ -1 & \text{se } \underline{x}(n) \in C_2 \end{cases}$		$y(n) = \text{sgn}(\underline{w}^T(n)\underline{x}(n)); \quad \text{sgn}(x) = \begin{cases} +1 & \text{se } x \geq 0 \\ -1 & \text{se } x < 0 \end{cases}$	
ACERTO		ERRO	
$d(n) - y(n)$		$d(n) - y(n)$	
Classe C1	$+1 - (+1) = 0$	Classe C1	$+1 - (-1) \neq 0$
Classe C2	$-1 - (-1) = 0$	Classe C2	$-1 - (+1) \neq 0$

Portanto, se $\underline{w}(n)$ não se altera para todos os vetores de treino $\underline{x}(n)$, então isto indica que o algoritmo convergiu.

A resposta quantizada $y(n)$ do Perceptron pode ser, então, expressa na forma compacta:

$$y(n) = \text{sgn}(\underline{w}^T(n)\underline{x}(n))$$

Notas adicionais:

- O vetor de entrada $\underline{x}(n)$ é um vetor $[(m+1) \times 1]$, cujo primeiro elemento é fixo em +1 (*bias*) ao longo de todo o processo iterativo.
- O vetor de pesos $\underline{w}(n)$ é um vetor $[(m+1) \times 1]$, cujo primeiro elemento é a transmitância associada ao *bias* $b(n)$.
- A resposta desejada quantizada $d(n)$ é definida por

$$d(n) = \begin{cases} +1 & \text{se } \underline{x}(n) \text{ pertence à classe } C_1 \\ -1 & \text{se } \underline{x}(n) \text{ pertence à classe } C_2 \end{cases}$$

Então, a adaptação do vetor de pesos $\underline{w}(n)$ pode ser sumariada na forma da regra de aprendizado por correção de erro:

$$\underline{w}(n+1) = \underline{w}(n) + \eta [d(n) - y(n)] \underline{x}(n)$$

onde η é o parâmetro razão de aprendizado, e a diferença $d(n) - y(n)$ representa o sinal de erro.

Notas adicionais:

O parâmetro razão de aprendizado é uma constante positiva limitada ao intervalo $0 < \eta \leq 1$.

Na escolha de um valor para η , dentro deste intervalo, é preciso considerar dois requisitos conflitantes:

- Manter a estabilidade da trajetória (estimativas estáveis para os pesos) \rightarrow
requer valores pequenos para η ;
- Adaptar-se de forma rápida com respeito às mudanças reais nas distribuições subjacentes do processo responsável pela geração do vetor de entrada $\underline{x} \rightarrow$
requer valores grandes para η .

Perceptron de Roseblatt como classificador - Exemplo:

Seja o conjunto X com NumVect vetores de dimensão DimVect e o vetor D com as respectivas classificações $\{-1, 1\}$ para cada um dos NumVect vetores de X conforme abaixo, onde o primeiro elemento de cada vetor é o *bias* de valor $+1.0$:

$$X := \begin{pmatrix} 1.0 & 0.3 & -0.5 & -0.7 \\ 1.0 & 0.25 & -0.8 & -0.6 \\ 1.0 & -0.8 & 0.9 & 0.5 \\ 1.0 & -0.7 & 0.6 & 0.4 \\ 1.0 & 9.3 & -0.45 & -0.35 \end{pmatrix} \quad D := \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$\text{NumVect} := \text{rows}(X) = 5$
 $\text{DimVect} := \text{cols}(X) = 4$

Utilizando a regra de aprendizado do Perceptron com uma razão de aprendizado $\eta := 0.5$, pede-se:

- a) Determine o vetor W dos pesos sinápticos do Perceptron, desde a iteração inicial $n = 0$ até a convergência, quando $W_{n+1} = W_n$ para todos os $\text{NumVect} = 5$ vetores de X .
- b) Verifique a consistência do Perceptron determinado em a) como classificador dos vetores do conjunto X , tendo com referência o vetor D com as respectivas classificações $\{-1, 1\}$ para cada um dos $\text{NumVect} = 5$ vetores de X .
- c) Compare os valores de entrada e de saída do *hard limiter* do Perceptron após a sua convergência para cada um dos $\text{NumVect} = 5$ vetores de X .

Solução:

A regra de aprendizado do Perceptron é conforme o procedimento abaixo, onde $NIter$ é o número de iterações desejadas:

```
Perceptron(X, D,  $\eta$ , NIter) :=
```

	$DimVect \leftarrow cols(X)$
	$NumVect \leftarrow rows(X)$
	for $k \in 0 .. DimVect - 1$
	$W_k \leftarrow 0$
	for $n \in 0 .. NIter - 1$
	$y \leftarrow \text{signum} \left[\sum_{k=0}^{DimVect-1} (W_k \cdot X_{\text{mod}(n, NumVect), k}) \right]$
	$W \leftarrow W + \eta \cdot (D_{\text{mod}(n, NumVect)} - y) \cdot (X^T)^{\langle \text{mod}(n, NumVect) \rangle}$
	$Historico^{\langle n \rangle} \leftarrow W$
	Historico

a) $NIter := 16$ $\eta = 0.5$ $\underline{W} := \text{Perceptron}(X, D, \eta, NIter)$

$$W^T =$$

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	-1	0.8	-0.9	-0.5
3	-1	0.8	-0.9	-0.5
4	-2	-8.5	-0.45	-0.15
5	-1	-8.2	-0.95	-0.85
6	0	-7.95	-1.75	-1.45
7	-1	-7.15	-2.65	-1.95
8	-2	-6.45	-3.25	-2.35
9	-2	-6.45	-3.25	-2.35
10	-1	-6.15	-3.75	-3.05
11	-1	-6.15	-3.75	-3.05
12	-1	-6.15	-3.75	-3.05
13	-1	-6.15	-3.75	-3.05
14	-1	-6.15	-3.75	-3.05
15	-1	-6.15	-3.75	-3.05
16				
17				

Vetor W após a convergência:

$$W^{(NIter-1)} = \begin{pmatrix} -1 \\ -6.15 \\ -3.75 \\ -3.05 \end{pmatrix}$$

← ocorreu a convergência nesta iteração porque observa-se que $W_{n+1} = W_n$ para todos os $\text{NumVect} = 5$ vetores de X.

b)

$$D = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \quad \text{signum}(X \cdot W^{\langle NIter-1 \rangle}) = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$$\text{ErroDeClassificacao} := \text{signum}(X \cdot W^{\langle NIter-1 \rangle}) - D = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

c)

entrada do *hard limiter* :

saída do *hard limiter* :

$$X \cdot W^{\langle NIter-1 \rangle} = \begin{pmatrix} 1.165 \\ 2.293 \\ -0.98 \\ -0.165 \\ -55.44 \end{pmatrix}$$

$$\text{signum}(X \cdot W^{\langle NIter-1 \rangle}) = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

Homework:

Seja o conjunto X com NumVect vetores de dimensão DimVect e o vetor D com as respectivas classificações $\{-1, 1\}$ para cada um dos NumVect vetores de X conforme abaixo (não usar *bias* neste exemplo):

$$X := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$D := \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$$\text{NumVect} := \text{rows}(X) = 5$$

$$\text{DimVect} := \text{cols}(X) = 4$$

Utilizando a regra de aprendizado do Perceptron com uma razão de aprendizado $\eta := 0.5$, pede-se:

- Determine o vetor W dos pesos sinápticos do Perceptron, desde a iteração inicial $n = 0$ até a convergência, quando $W_{n+1} = W_n$ para todos os $\text{NumVect} = 5$ vetores de X .
- Verifique a consistência do Perceptron determinado em a) como classificador dos vetores do conjunto X , tendo com referência o vetor D com as respectivas classificações $\{-1, 1\}$ para cada um dos $\text{NumVect} = 5$ vetores de X .
- Compare os valores de entrada e de saída do *hard limiter* do Perceptron após a sua convergência para cada um dos $\text{NumVect} = 5$ vetores de X .

Homework - respostas:

A convergência é observada após 8 iterações.

a) Vetor W após a convergência: $W^{\langle NIter-1 \rangle} = \begin{pmatrix} 0 \\ 0 \\ -1 \\ -1 \end{pmatrix}$

b) $D = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$ $\text{signum}(X \cdot W^{\langle NIter-1 \rangle}) = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$

ErroDeClassificacao := $\text{signum}(X \cdot W^{\langle NIter-1 \rangle}) - D = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

c) entrada do *hard limiter* : saída do *hard limiter* :

$X \cdot W^{\langle NIter-1 \rangle} = \begin{pmatrix} 0 \\ 0 \\ -2 \\ -2 \\ -2 \end{pmatrix}$ $\text{signum}(X \cdot W^{\langle NIter-1 \rangle}) = \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$